

# Arction Gauges

## User's Manual



## About this document

This document is a brief User's Manual, quick introduction to **Arction Gauges software components**. Only essential key features are explained. Hundreds of classes, properties or methods are not described in this document. Run the provided demo applications for getting a quick preview of some Gauge features. For code examples, take a look at the source code of included demo applications.

All code examples in this document are written in C# language and XAML for User Interface programming. Majority of demo applications provide code preview in C# and XAML, as well.

**And remember, don't hesitate to contact support ([support@arction.com](mailto:support@arction.com)) if you have any question!**

**Applies to Gauge edition, v.1.0**



Copyright Arction Ltd 2016

All rights reserved.

[www.arction.com](http://www.arction.com)

[www.lightningchart.com](http://www.lightningchart.com)

## Table of contents

1. Overview.....	5
1.1 Components .....	5
2. Installation.....	5
2.1 Before installing.....	5
2.2 Adding Arcion components manually to Visual Studio Toolbox.....	6
2.3 Selecting target framework.....	6
3. Introduction.....	8
3.1 Components .....	8
4. Gauge component .....	9
4.1 Adding from toolbox into WPF and Silverlight project.....	9
4.2 Adding into Blend WPF project .....	9
4.2.1 Properties .....	10
4.2.2 Event handlers.....	10
4.3 Object model .....	11
5. Gauge.....	12
5.1 Gauge Customization.....	14
5.1.1 Shape customization .....	14
5.1.2 Colour Customization .....	16
6. Scale.....	19
6.1 Dial.....	23
6.1.1 Needles .....	23
6.2 Tick Lines.....	26
6.2.1 Properties .....	26
7. Troubleshooting .....	29
7.1 Web support.....	29

# 1. Overview

Arction Gauges is an add-on to Microsoft Visual Studio, having a radial gauge control for Windows Presentation Foundation (WPF) and Silverlight .NET platforms.

## 1.1 Components

The main component **for WPF** is

- Arction.WPF.Gauges

and for **Silverlight**

- Arction.Silverlight.Gauges

# 2. Installation

## 2.1 Before installing

Check your computer configuration meets the requirements

### ***System requirements for WPF***

- Windows Vista, 7, 8 or 10, as 32 bit or 64 bit
- Visual Studio 2010-2015
- .NET framework v. 4.0 or newer installed

### ***System requirements for Silverlight***

- Vista, 7, 8 or 10, Windows Server 2008 (R2 SP1,SP2)/2012(R2), as 32 bit or 64 bit
- Visual Studio 2010-2015
- .NET framework v. 4.0 or newer installed
- Silverlight-capable browser, and Silverlight plug-in installed

Check the requirements in Microsoft website.

## 2.2 Adding Arction components manually to Visual Studio Toolbox

### WPF

1. Open Visual studio. Create a new **WPF** project. Right-click on Toolbox, select **Add Tab** and give name "Arction Gauges"
2. Right-click on Arction Gauges tab and select **Choose items...**
3. In **Choose Toolbox items** window, Select **WPF Components** page. Click **Browse...**  
Browse **Arction.WPF.Gauges.dll**, from the package you downloaded, and click open. The components can be now found in the toolbox.

### Silverlight

1. Open Visual studio. Create a new **Silverlight** project. Right-click on Toolbox, select **Add Tab** and give name "Arction Gauges"
2. Right-click on Arction Gauges tab and select **Choose items...**
3. In **Choose Toolbox items** window, Select **Silverlight Components** page. Click **Browse...**  
Browse **Arction.Silverlight.Gauges.dll**, from the package you downloaded, and click open. The components can be now found in the toolbox.

## 2.3 Selecting target framework

In C# project, the framework selection can be made in **Project -> Properties -> Application -> Target framework**.

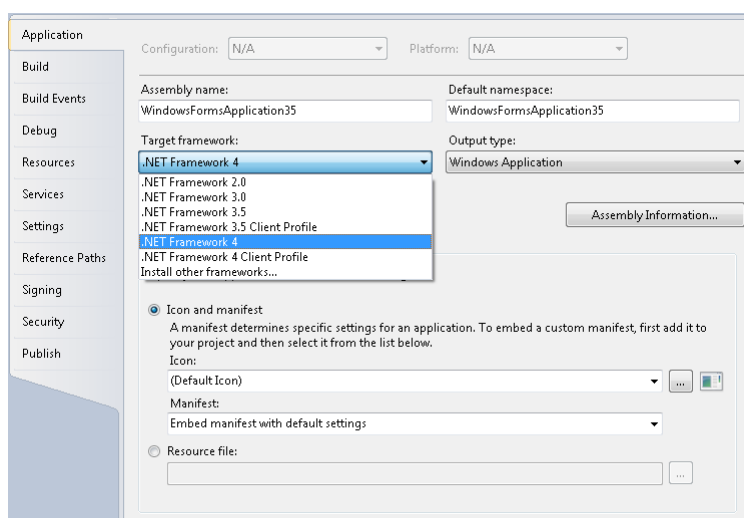


Figure 2-1. Selecting target framework in C# project.

In Visual Basic project, the framework can be selected in **Project -> Options -> Compile -> Advanced compile options -> Target framework**.

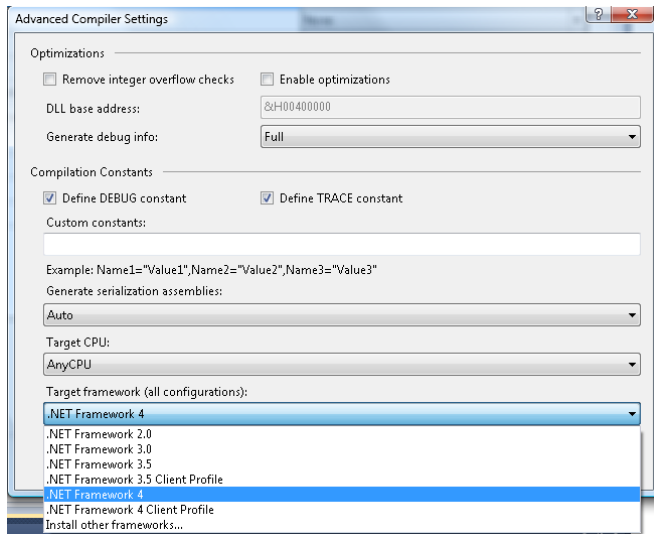


Figure 2-2. Selecting target framework in Visual Basic project.

## 3. Introduction

### 3.1 Components

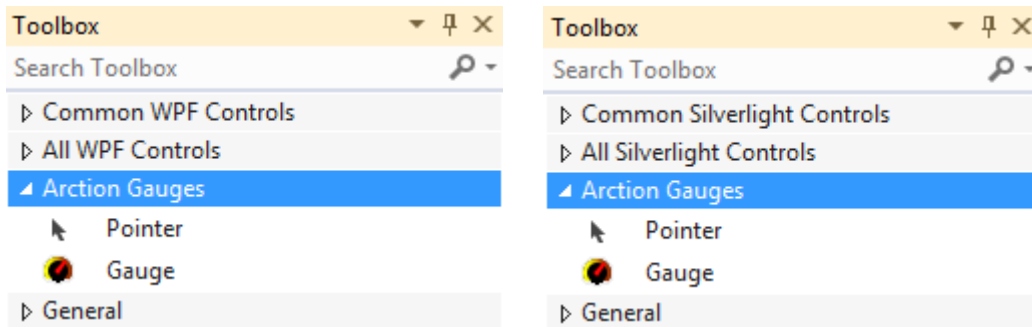


Figure 3-1. WPF and Silverlight Toolbox components.

Components in **Arction.WPF.Gauges.dll** and **Arction.Silverlight.Gauges.dll** have an identical meaning.

- **Gauge.** The measuring instrument, which displays certain information, e.g. Time, Voltage, Pressure, etc.
- **Scale.** One of the main components of a gauge defined on a set of points.
- **Dial.** The component of the gauge, where numerical labels are arranged in a circular layout so that a pointed needle end shows the measured data by rotating around a center.
- **MajorTicksLine.** Small tick lines on a scale show major results.
- **MinorTicksLine.** Small tick lines on a scale show intermediate results.
- **TertiaryTickLine.** Small tick line on a scale shows major or intermediate results according to the scale settings.
- **CustomTicksLine.** Gives an opportunity to create fully customized lines on a gauge scale.



## 4. Gauge component

### 4.1 Adding from toolbox into WPF and Silverlight project

Add **Gauge** (*Arction.WPF.Gauges*) control from your toolbox to your Window or other container. The gauge appears into the designer and properties are shown in **Properties window**. XAML editor shows the contents and modifications to the gauge default properties.

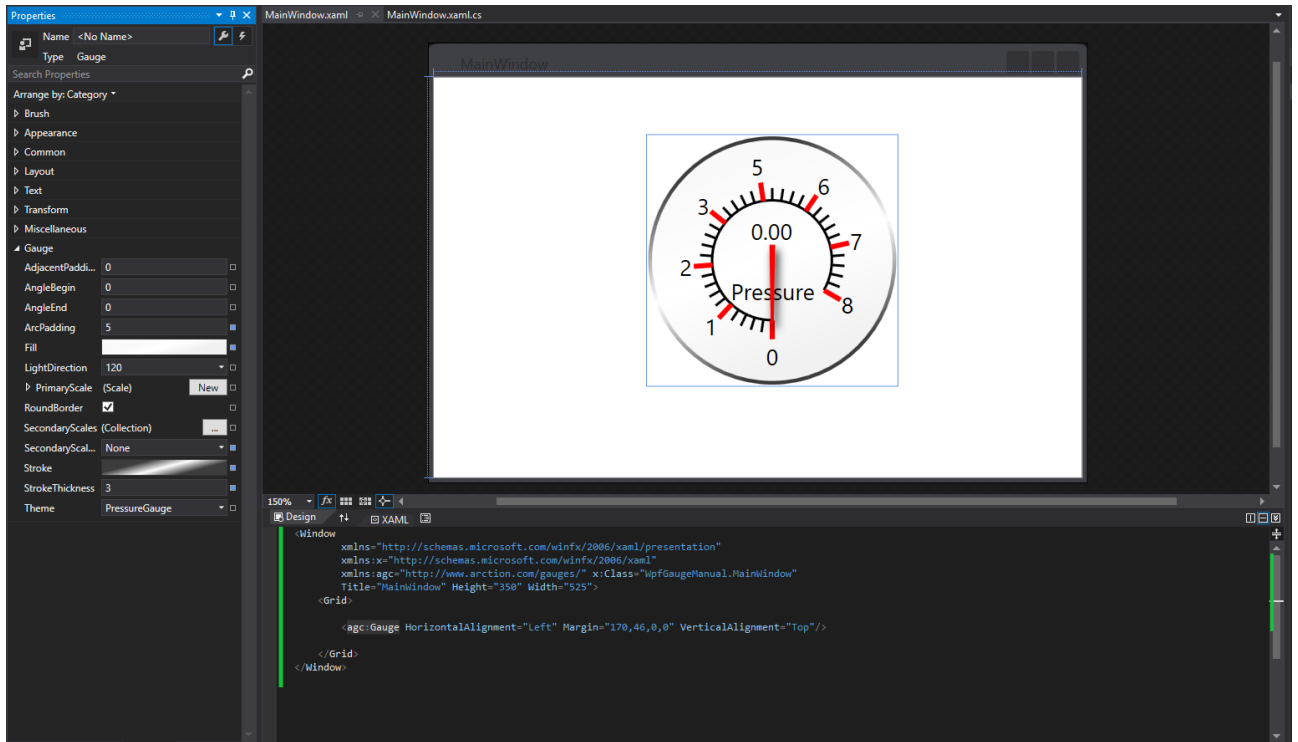


Figure 4-1. Gauge control added into WPF designer.

### 4.2 Adding into Blend WPF project

In **Projects** tab, go to **References**. Right-click and select **Add reference...** Browse *Arction.WPF.Gauges.dll* from the downloaded package.

Go to **Assets** tab. Write "Gauge" in the Search box. It lists **Gauge** row in the search results. Drag-drop the object into your WPF window.

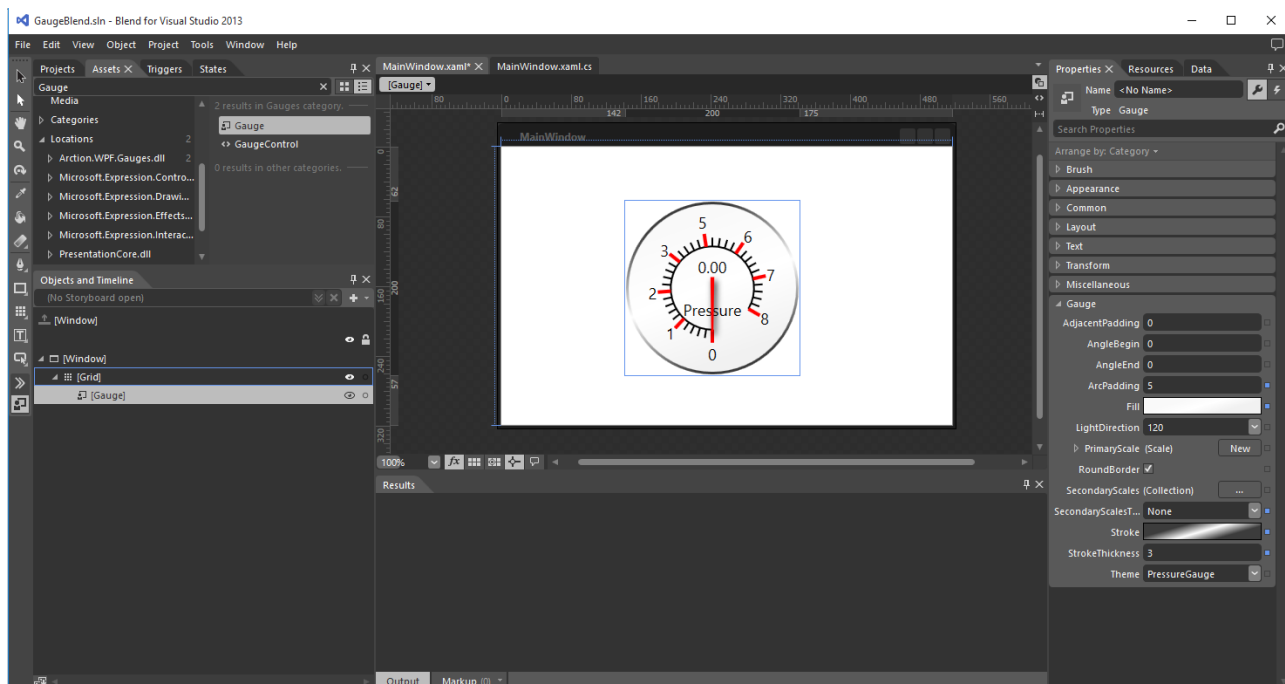


Figure 4-2. Gauge control added into Blend For Visual Studio 2013 designer.

## 4.2.1 Properties

Most of the properties can be edited in Properties window and XAML editor, and are bindable in MVVM way.

## 4.2.2 Event handlers

Event handlers of the gauge main level can be assigned with the property grid directly. For objects that you have added to the collections, events can be assigned *in code* or by navigating to object's row in XAML editor and by defining the event handler *in property grid*.

When adding same event handler in code by defining `object.SomethingChanged += [TAB][TAB]`, it creates the event handler correctly.

## 4.3 Object model

The object model of Gauge can be learned quickly by using *Properties editor* of Visual Studio.

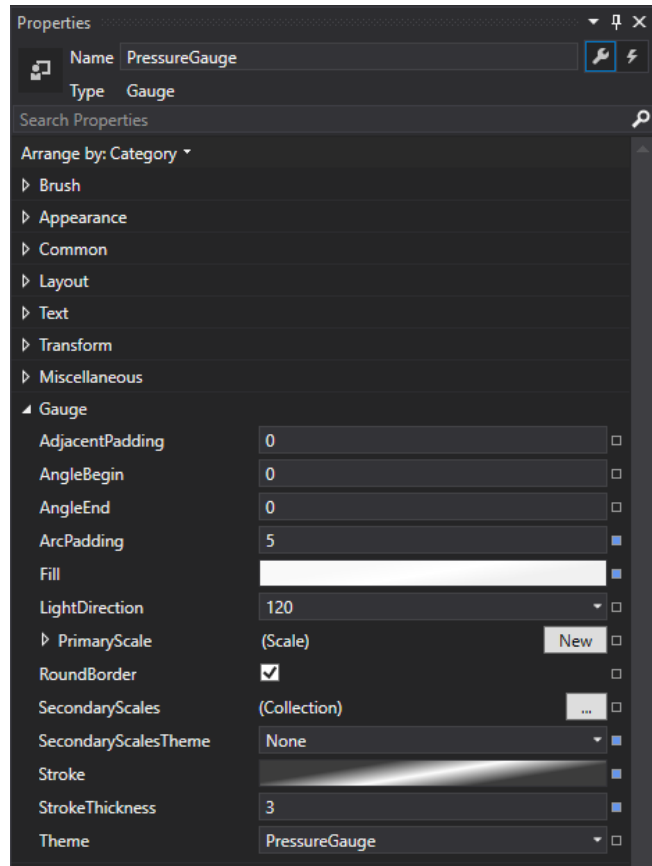


Figure 4-4. Gauge specific properties can be found under *Gauge* category in both Silverlight and WPF Properties window. By expanding the nodes, or in WPF creating new objects, you will notice a huge set of properties exist in overall.

## 5. Gauge

Gauge allows presenting a wide variety of data. As any other measuring instrument, it has Scales, it is possible to have more than one scale. Thus, users are able to have several data readings at the same time.

Gauges can be fully customized by shape (AdjacentPadding, AngleBegin, AngleEnd, ArcPadding, RoundBorder, StrokeThickness), colour (Fill, Stroke), and scale properties. WPF and Silverlight gauges have a set of themes. Each of it has its own default design, which can be customized too. A certain theme can be selected from a list of themes, see Figure below.

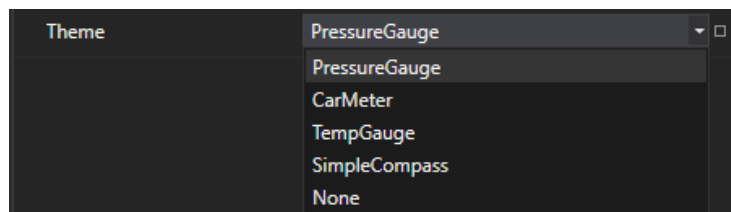


Figure 5-1. Gauge theme selection property

Although the default gauge has a primary scale, all the properties for PrimaryScale and SecondaryScales Collection are invisible. To make them visible, click on a button **New** and the properties for Primary scale will appear, and relevant XAML block will be added to the designer.

Primary Scale appending:

```
<agc:Gauge>
  <agc:Gauge.PrimaryScale>
    <agc:Scale/>
  </agc:Gauge.PrimaryScale>
</agc:Gauge>
```

Secondary Scales appending:

```
<agc:Gauge>
  <!-- Secondary Scales -->
  <agc:Gauge.SecondaryScales>

    <!-- SecondaryScale 1 -->
    <agc:Scale/>

    <!-- SecondaryScale 2 -->
    <agc:Scale/>

    <!-- SecondaryScale 3 -->
    <agc:Scale/>

  </agc:Gauge.SecondaryScales>

  <!-- Primary Scale -->
  <agc:Gauge.PrimaryScale>
    <agc:Scale/>
  </agc:Gauge.PrimaryScale>
</agc:Gauge>
```

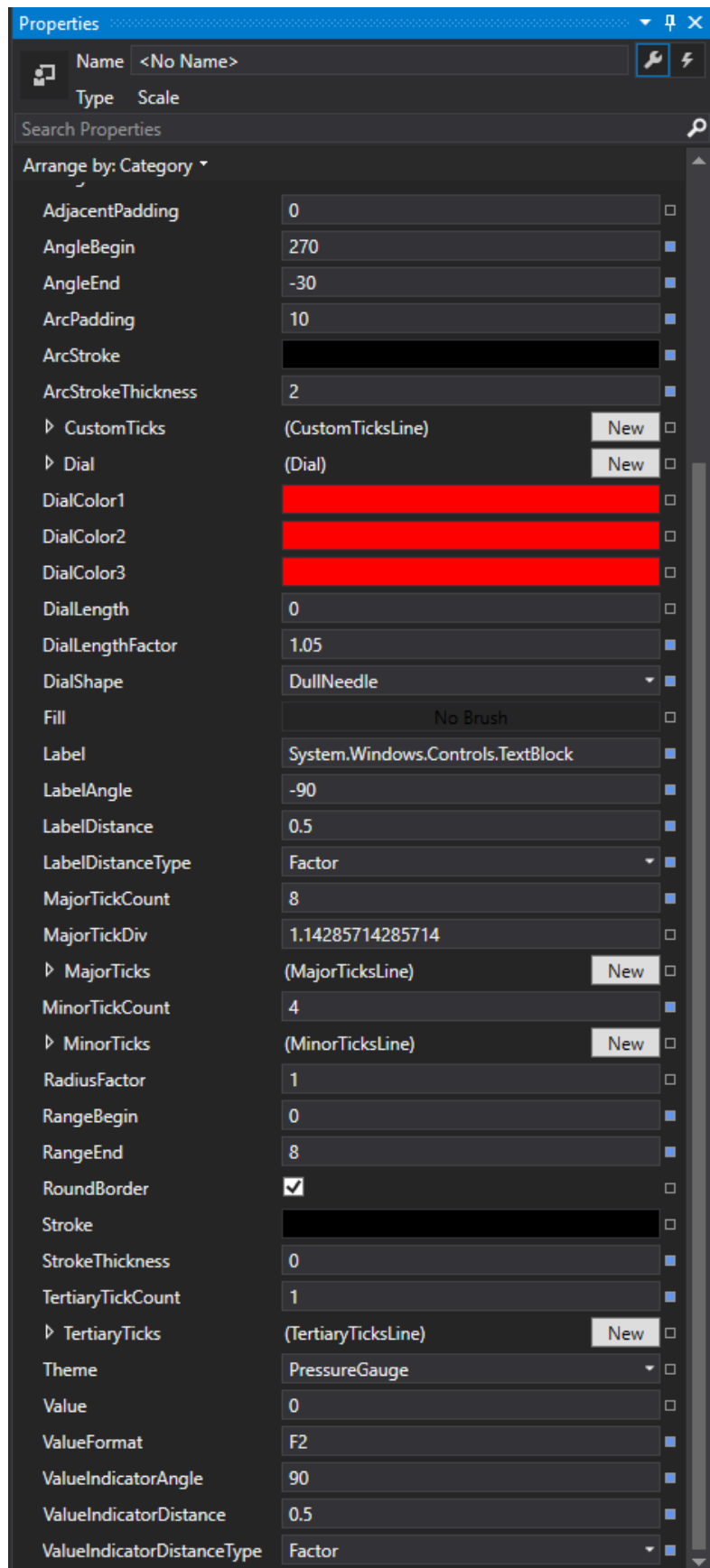


Figure 5-2. Scale list of properties

## 5.1 Gauge Customization

### 5.1.1 Shape customization

**AdjacentPadding.** Changing this property, a user can see how circular border of a gauge will take various angular shapes. This property affects only on the gauge, scales should be modified separately.



Figure 5-3. Shape modification after setting `AngleBegin="138"`, `AngleEnd="0"` with `AdjacentPadding="44"`

**AngleBegin** and **AngleEnd.** These two properties determine two points of gauge's border (start and end), which can be different – the result is an arc; otherwise the gauge's border is a full circle.



Figure 5-4. Shape after setting `AngleBegin="138"`, `AngleEnd="0"` without `AdjacentPadding`

**ArcPadding.** This property defines a size of empty space between scale's stroke and gauge's border.



Figure 5-5. The gauge with increased `ArcPadding="20"`

**RoundBorder.** Boolean property determines a roundness of gauge's border. If we set the property to false and change an `AdjacentPadding` and `AngleBegin/AngleEnd` to some value, e.g. `AdjacentPadding = "114"`, `AngleBegin = "180"`, the result is shown below.



Figure 5-6. The gauge takes a new shape after varying of these three properties

**StrokeThickness.** This property defines a thickness of a gauge's border. (**BorderThickness**)



Figure 5-7. The **BorderThickness** originally was equal to 3, now it is equal to 8

### 5.1.2 Colour Customization

**Fill** – a user can fill a gauge in five different ways:

- Transparent, means no filling. A gauge will embrace the colour of a container background where it is placed.





- Solid Brush. A gauge will embrace the picked colour according to ARGB (Alfa, Red, Green, and Blue) channels.



Figure 5-8. Solid colour filling

- Gradient Brush. A gauge will embrace a progression between selected colours. A user can add as many colours as it needed by typing in XAML the code inside the <[Linear-/Radial-]GradientBrush/> block:

```
<GradientStop Color="[COLOR]" Offset="[1-100]">
```

or click on the <[Linear-/Radial-]GradientBrush/> block and find in the properties GradientStops (Collection). In addition, user can change the spreading of a gradient colour, the point coordinates on a gauge where the gradient starts and ends spreading.

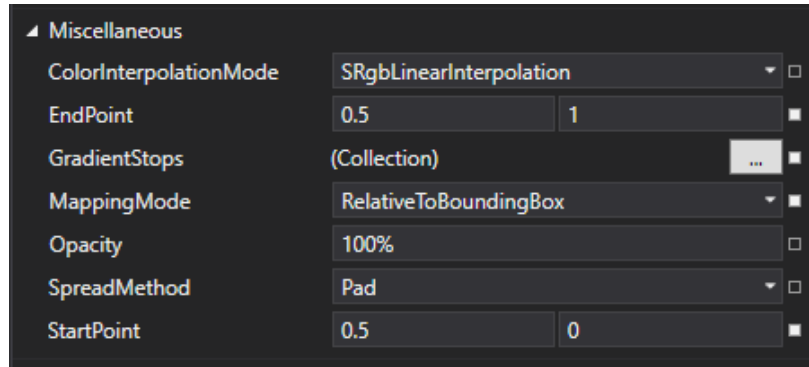


Figure 5-9. The LinearGradient settings

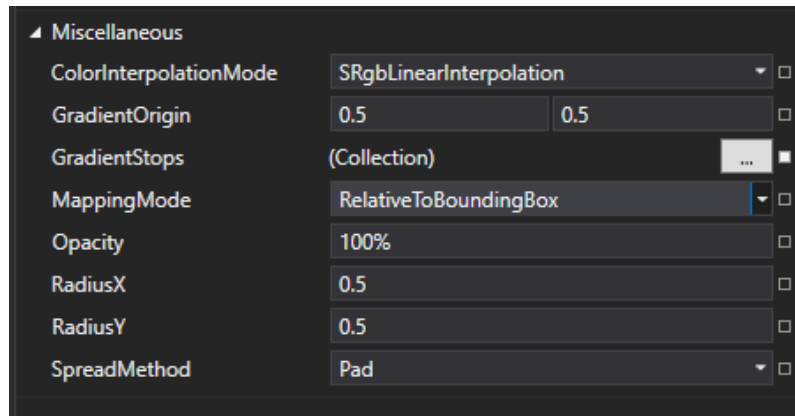


Figure 5-10. The RadialGradient settings

- Linear Gradient.



Figure 5-11. Linear gradient filling using two colours (two gradient stops differently coloured)

- Radial Gradient.



Figure 5-12. Radial gradient filling using two colours (two gradient stops differently coloured)

- Tile Brush. A gauge will not be coloured by a brush, a picture will be used as a gauge background.



Figure 5-13. Apply a texture picture as a filling for a gauge

- System Brush Resources. A gauge will embrace the system colour.

**Stroke** (Gauge Border Colour). A border colouring has the same colour properties as a gauge filling, which were mentioned before.

## 6. Scale

A gauge has the major scale named **PrimaryScale**, which can be only one, and minor scale – **SecondaryScale**, which can be infinite amount. Although they have a different priority, both have the same type Scale, thus, they have same properties, as it was mentioned in section 5. The properties see in the Figure 6-1.

Each scale has a circle shape by default with border and background, alike gauge inside another gauge. However, it might also take the shape and colour according to the properties such as, `AdjacentPadding`,

AngleBegin/-End, ArcPadding, Stroke (Border Colour), StrokeThickness, Fill, and RoundBorder, were mentioned in section 5.1.1 and 5.1.2. These properties will have effect only for a Scale.

**ArcStroke (Scale Stroke colour).** This property defines the colour that is applied to a scale stroke.

**ArcStrokeThickness (Scale Stroke Thickness).** This property defines a thickness of scale stroke.

**RadiusFactor.** This property is an index, which defines a coefficient of ratio between scale radius and gauge radius.

**RangeBegin.** This property means the minimum value on a scale.

**RangeEnd.** This property means the maximum value on a scale.

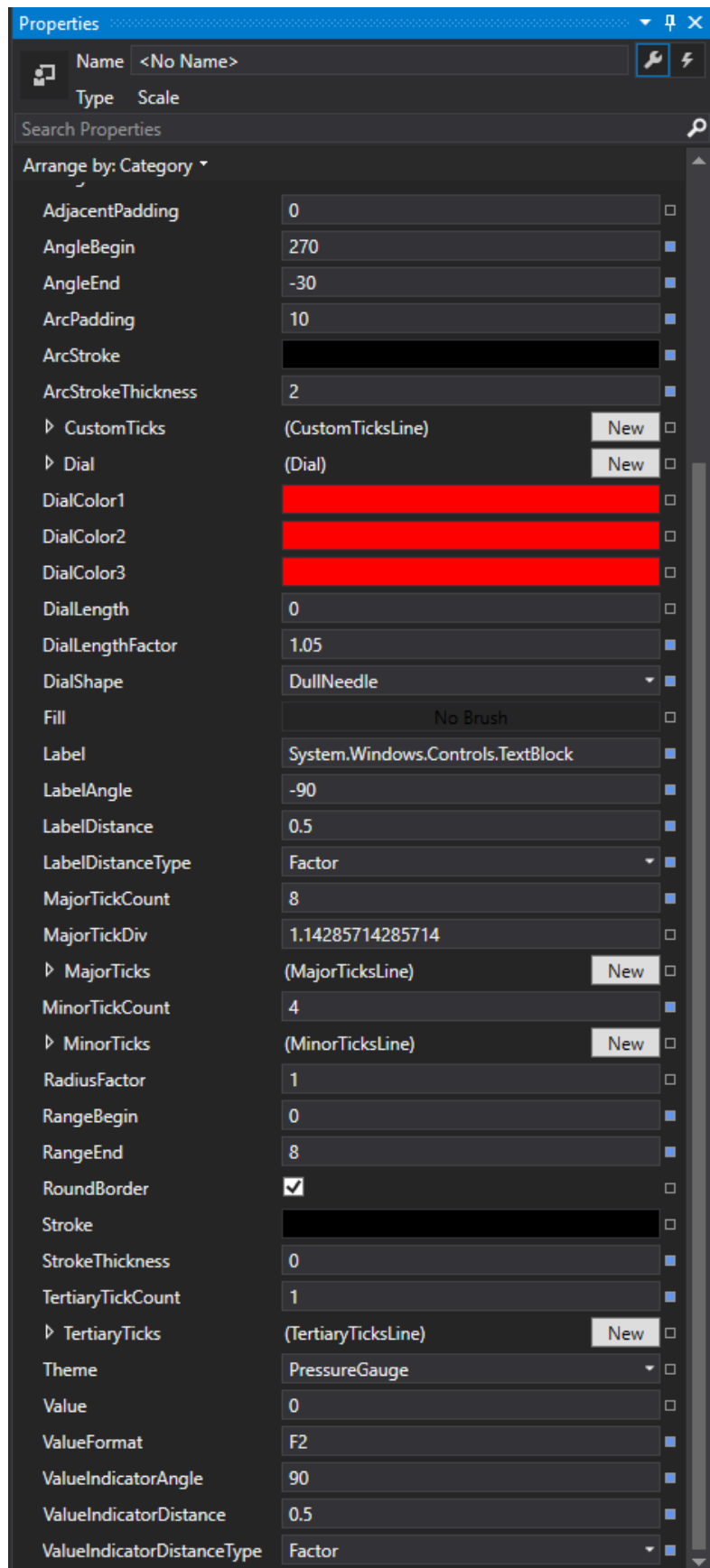


Figure 6-1. Object properties type of Scale

**Dial.** See section 6.1.

**CustomTicks.** A set of ticks and labels placed on a line inside a gauge. The sharp end of a needle has additional value indicator that shows the current value, this indicator can be customized. These ticks can be fully customized. See section 6.2.

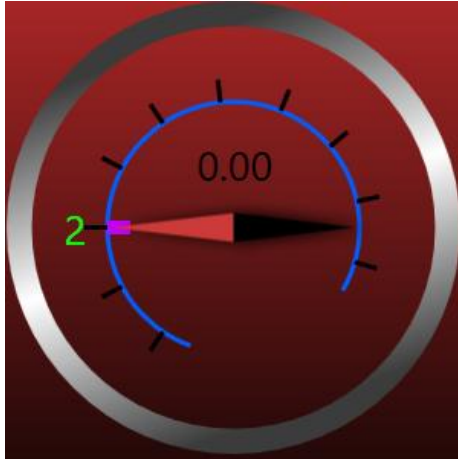


Figure 6-2. CustomTickLine Example

**MajorTicks, MinorTicks, TertiaryTicks.** A set of ticks and labels placed on a line inside a gauge. The sharp end of a needle shows the current value. See section 6.2.

**Value.** This is the value, which will be presented on a gauge.

**ValueFormat.** Represents the format in which a value indicator will show the data (e.g. F1 means one digit after floating point).

**ValueIndicatorAngle.** This property rotates a value indicator around the centre of gauge.

**ValueIndicatorDistance.** This property defines a positioning of the value indicator on the Y-axis at a distance from the centre of gauge according to the **ValueIndicatorDistanceType** property.

**ValueIndicatorDistanceType.** This property defines a factor or absolute distance in pixels for the value indicator on the Y-axis from the centre point.

**Label.** This is the main title of gauge. Can be written as a metric unit (Bar, C°, F°, km/h, MPH, etc.) or as the gauge's name.

**LabelAngle.** This property rotates a label around the centre of gauge.

**LabelDistance.** This property defines a positioning of the value indicator on the Y-axis at a distance from the centre of gauge according to the **LabelDistanceType** property.

**LabelDistanceType.** This property defines a factor or absolute distance in pixels for the label on the Y-axis from the centre point.

## 6.1 Dial

The dial property tree has settings for needle customization. Each scale has its own needle, but when a user adds new secondary scale, the needle is hidden (**DialShape** property is “None”). To make it visible, set the **DialShape** property to value differ from “none”. A primary scale has visible needle with opened properties for it, see Figure 6-3. By changing this properties user may change the colour of needle, length and shape. Arction Gauges have five different shapes included into WPF and Silverlight libraries.

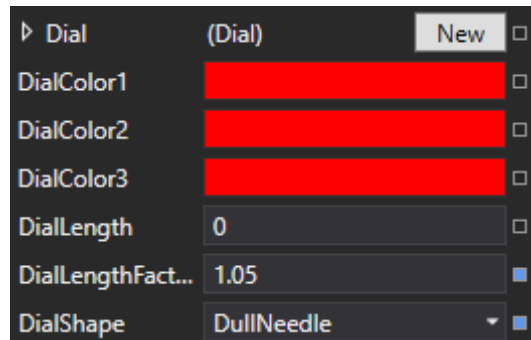


Figure 6-3. Basic needle properties

### 6.1.1 Needles

**DullNeedle** or **DefaultNeedle**. The colour of this needle is regulated by **DialColor1** solid brush colour property.

**Line**. The default colour is “Black” and cannot be changed.

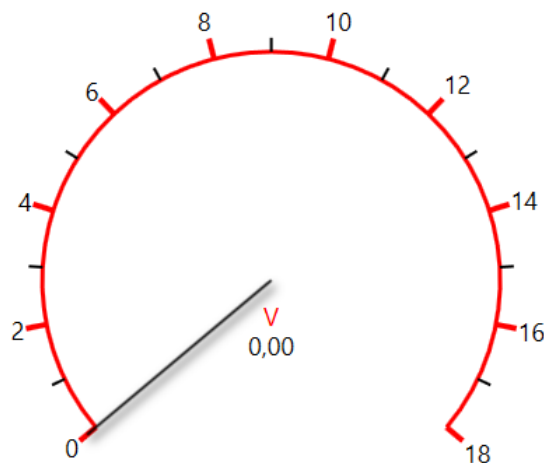


Figure 6-4. Needle has a “Line” shape

**WideNeedle.** This shape has three colour setters DialColor1, DialColor2, DialColor3 to create an interesting effect. See Figure 6-5.

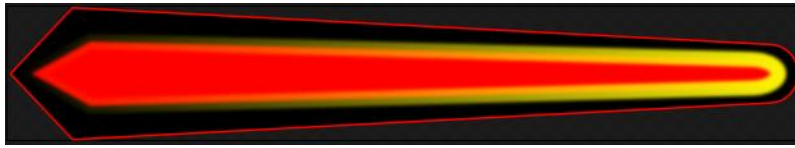


Figure 6-5. WideNeedle shape

**SimpleCompassNeedle.** This shape has close to real compass needle; it has two colours, configured by DialColor1 and DialColor2.

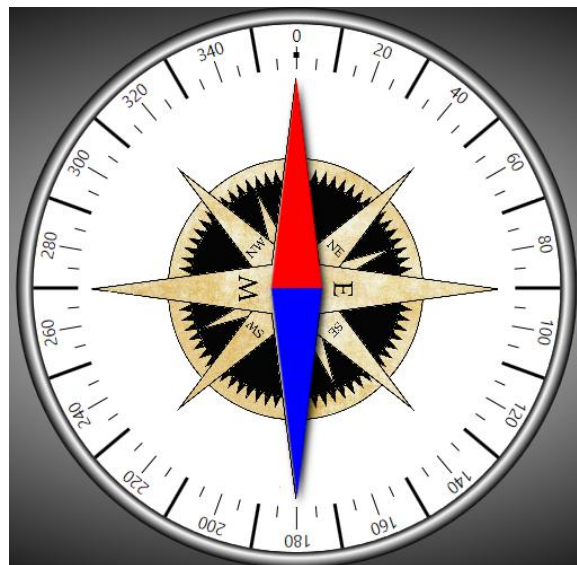


Figure 6-6. SimpleCompassNeedle shape

All the needles have the same advanced properties, which become visible if a user clicks on button “New”.

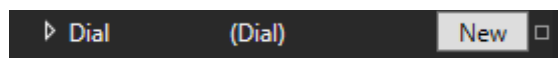


Figure 6-7. Add advanced properties for a needle

Using this advanced settings we can change the thickness of certain needle by defining **AspectRatio**. If the needle needs to be not in the center of gauge, user can easily move the rotation point by changing **Margin** and **Center** values. Set the length in pixels or in ratio using **Length** and **LengthFactor** properties. Even a user may follow the values in some properties, which can be changed during the gauging, for example Angle of needle – get the current angle of needle and use it further.



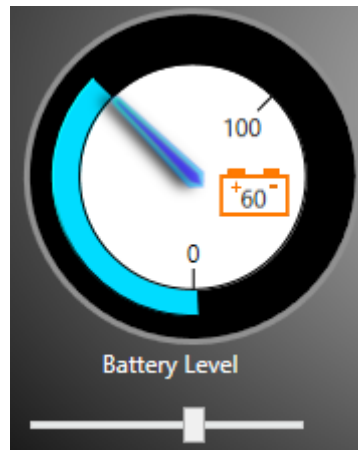


Figure 6-8. The light blue line follows the needle by binding an angle of the needle

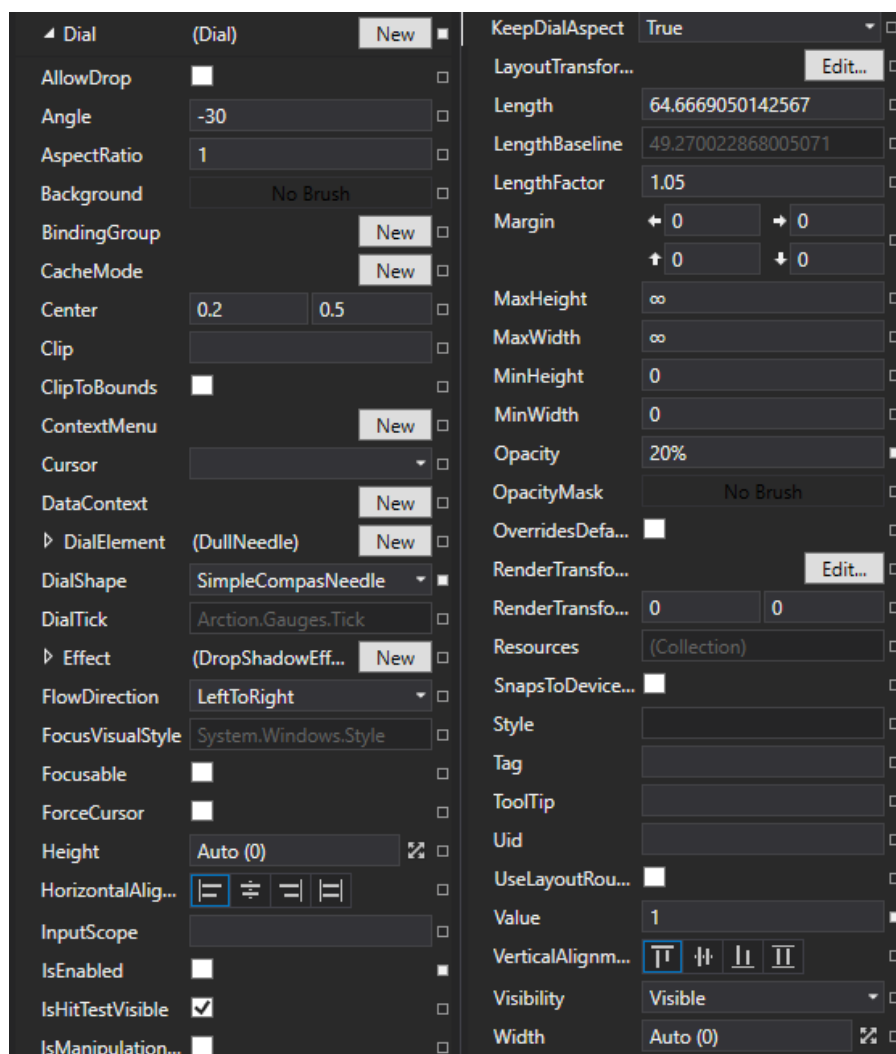


Figure 6-9. Dial advanced property tree

## 6.2 Tick Lines

Each scale has three types of tick lines (Major, Minor and Tertiary), which are widely used in the gauges. According to the position of needle in a front of certain tick, a user can see a current value. In addition, they can be fully customized to make a gauge more attractive.

All the tick lines have identical properties, which will be explained in further section.

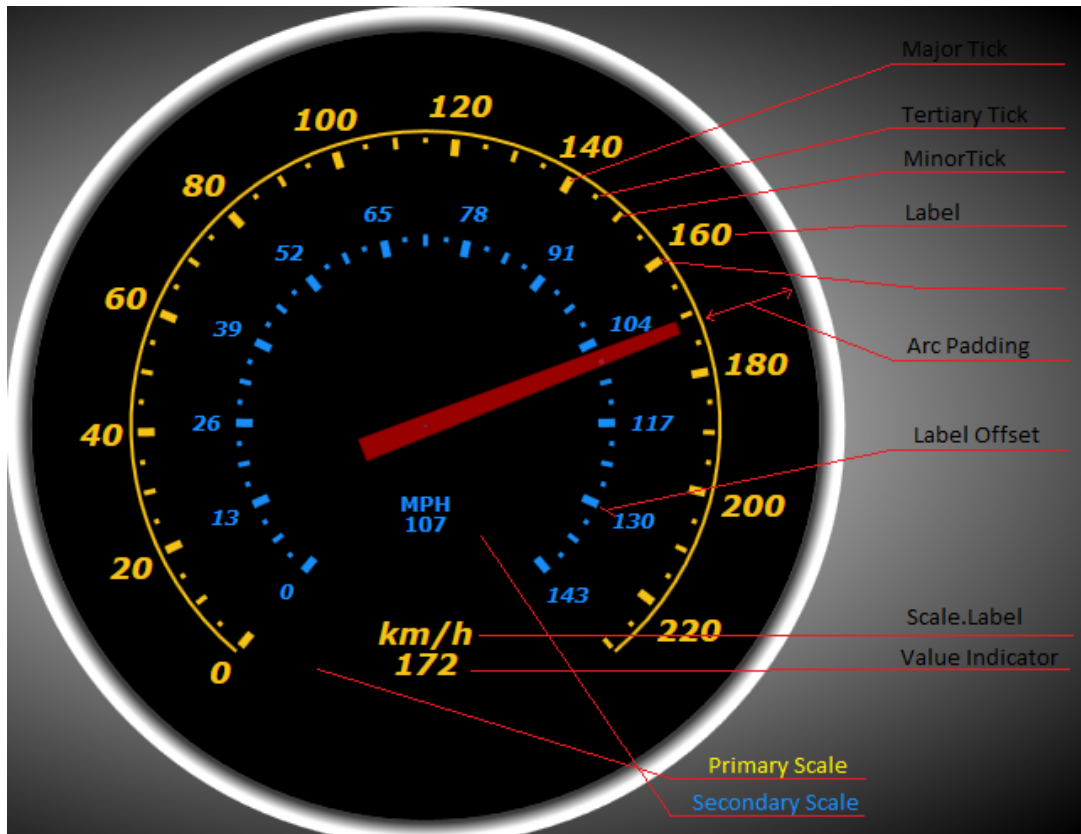


Figure 6-10. Basic gauge structure

### 6.2.1 Properties

User has a variety of properties to customise tick lines. But they are hidden; to open them, the user has to add new tick line for the scale by clicking on button “New”. The opened manager will offer several cases for selecting, see Figure 6-10.

Each of the tick lines has its own default properties, for example Major ticks are bigger and labels for each value are visible as on a Figure 6-9. Minor and Tertiary ticks have smaller sizes and show intermediate results without labels.

However, all the tick lines can be customized and look differently in the end.

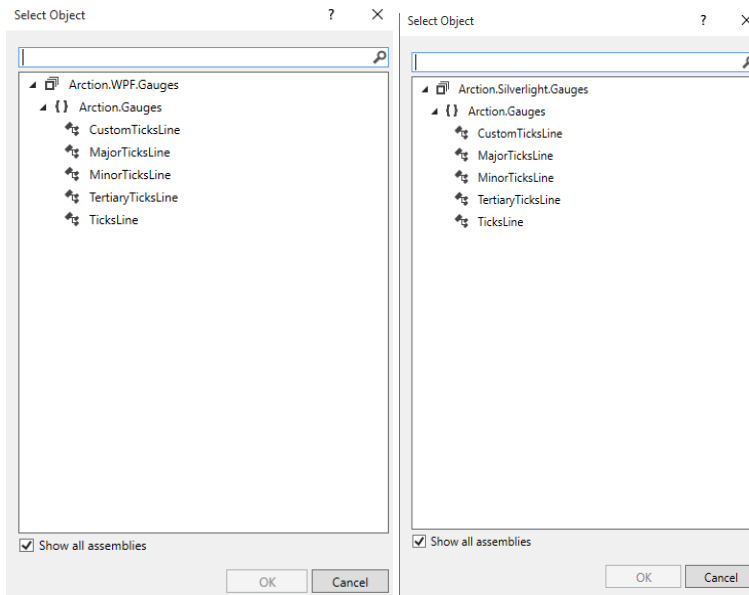


Figure 6-11. WPF and Silverlight manager to add new tick line

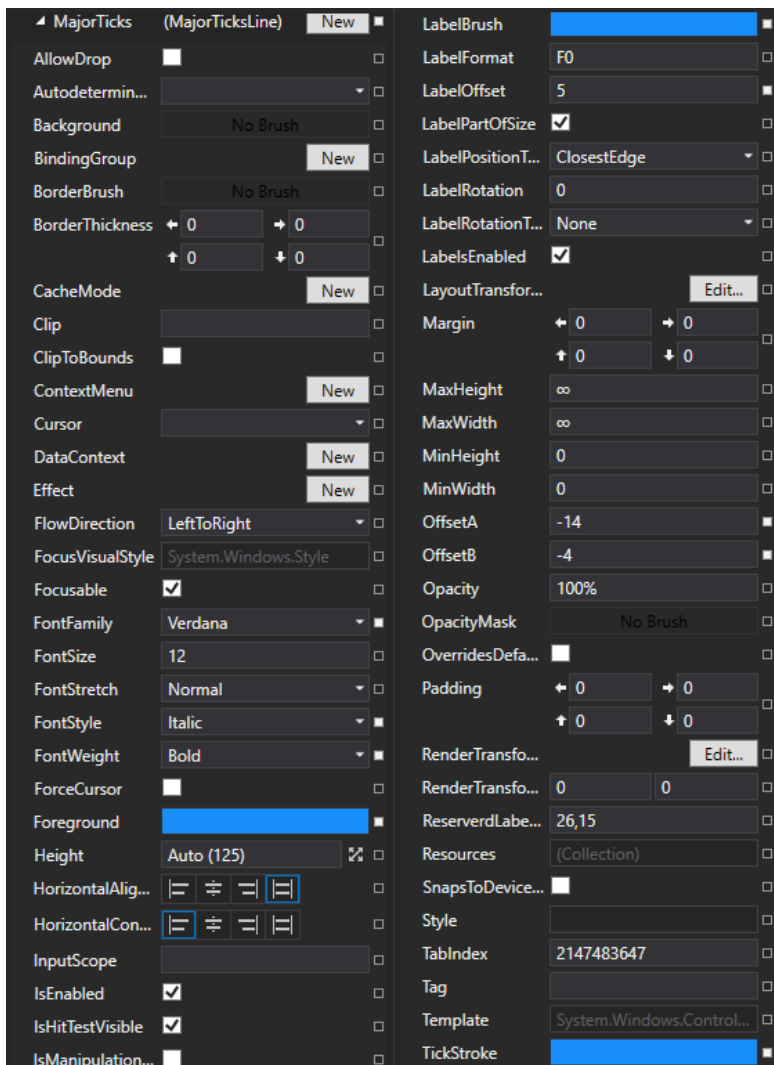


Figure 6-12. MajorTickLine properties, for the Minor, Tertiary and Custom lines properties are identical

**Major-/Minor-/Tertiary-TickCount.** This property defines amount of ticks that will appear on a scale.

**Major-/Minor-/Tertiary-TickDiv.** This property defines how frequently ticks will be placed and close to each other.

There are Font Properties for tick line labels, such as **FontFamily**, **FontSize**, **FontStretch**, **FontStyle**, **FontWeight**, and **Foreground**.

**IsEnabled.** Boolean property, if which one is true, the value indicator of a gauge will show the current value, otherwise, the value indicator will not shows the value changes, see Figure 6-2.

**Label Settings:**

- **LabelBrush.** This property defines the color of label.
- **LabelFormat.** This property defines to format in which label will present the value (floating point F2 – two digits, F3 – three digits, etc.).
- **LabelOffset.** This property defines a label offset from the scale stroke.
- **LabelPositionType.** This property defines a label positioning inside the rectangle where this label will be rendered.
- **LabelRotation.** This property defines an angle to rotate a label.
- **LabelRotationType.** This property defines four directions to rotate a label.
- **LabelsEnabled.** This property defines to hide labels or make them visible on a scale.

**Margin.** Additional settings to place labels and ticks on a scale.

**OffsetA** and **OffsetB.** The difference between these two properties defines the length of the tick; both offsets defined as an offset from a scale stroke.

**Opacity.** Regulates tick transparency.

**TickStroke.** This property defines the color of ticks.

**TickThickness.** This property defines the thickness of ticks.

## 7. Troubleshooting

### 7.1 Web support

See [www.arction.com/support](http://www.arction.com/support) for frequently asked questions.

Discussion forums are available at <http://www.arction.com/forum>