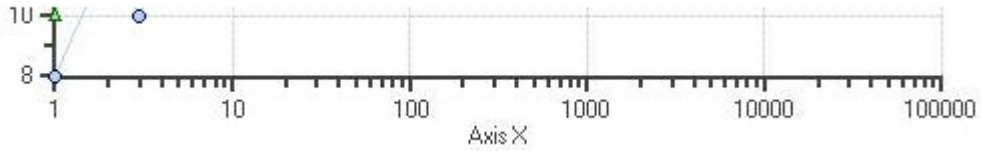
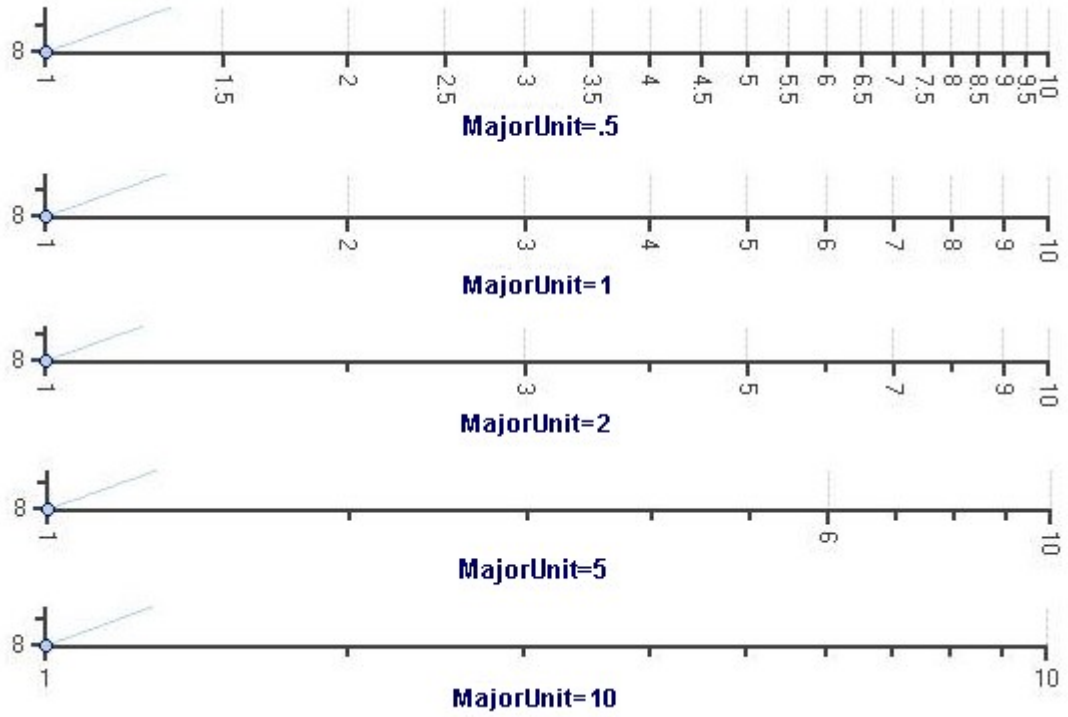
UnitMajor 和对数坐标轴

对于对数轴缩放，MajorUnit（在线文档
'MajorUnit属性'）作为一个乘数的每个周期，提供了一个提示注释间距在每个周期内的对数的底基值。即（MajorUnit
*基周期值）大约是每个周期内的注释增值。对于整数的对数基准值，结果通常是准确的。对于浮点值，注释是以很好的数字为线性缩放的。
关于关于UnitMajor以及以及Logarithmic坐标轴的详细解释坐标轴的详细解释
通常，当使用对数刻度时，一个图表轴的范围将跨越多个周期的对数基数。在这些情况下，对MajorUnit来说通常的线性规范不再是有道理的
，作为一个适当的值的一个给定的周期为上一个或下一个周期是没有意义的。对于MajorUnit设置的值而言，它必须是相对于对数基数的周期。
如果这对你没有意义，要想一想，固定的，增量的值，你可以使用以下的轴：



根据上面的推理，在对数坐标轴的图，假设MajorUnit 指定每个周期的基础值的分数。考虑下面的例子：



在每一种情况下，基准周期值为1。每个周期的下一个标注值=前一个数+（基数周期* MajorUnit）。MajorUnit
的最大值是LogarithmicBase。MajorUnit的自动值始终是LogBase。
当所有的注释值被计算，一个很好的四舍五入算法将应用到数值，使得该数值容易被阅读。该行为可能有点奇怪，但它是可容纳任何对数的基数，同时获取的标注的数值也是容易阅读的。
例如，以上绘图区是以10为底的常用对数，但是同样也有固有的其他底数值，比如说以2为底，以x为底等等。
图表图例
隐藏图表图例
以编程方式隐藏图表图例可以参考以下：
在XAML中向图例添加名称，之后您可以通过代码修改其可见性：legend.Visibility = ...

XAML

```
<c1chart:C1Chart x:Name="chart" >
<c1chart:C1ChartLegend x:Name="legend" /> ...

</c1chart:C1Chart>
```

改变图例的方向和位置为了在图表控件底部居中、水平显示图表图例，请使用以下代码：

C#

```
C1ChartLegend.Orientation = Horizontal;
C1ChartLegend.Position = C1.WPF.C1Chart.LegendPosition.BottomCenter;
```

图表视图

ChartView（在线文档 'ChartView 类'）对象表示图表中包含数据的区域（包括标题和图例，但是不包括坐标轴）。View 属性返回一个ChartView对象。

设置绘图区背景

设置绘图区背景是一种比较简单的方式添加一些自定义以及提高您图表的对比度。
下面的XAML标记允许您设置背景：

XAML

```
<c1:C1Chart.View>
<c1:ChartView PlotBackground="#FF343434">

</c1:ChartView>
</c1:C1Chart.View>
```

你也可以在代码中设置PlotBackground 属性。只需要给您的图表指定名字，然后将以下代码添加到InitializeComponent()方法中：

C#

```
chart.View.PlotBackground = Brushes.BlueViolet;
```

数据绑定

WPF及Silverlight版Chart中的C1Chart（在线文档 'C1Chart 类'）控件可以绑定到任意实现了System.Collections.IEnumerable的对象（比如说XmlDataProvider、DataSet、DataView、等等）。
一个数据表可以通过指定给C1Chart控件的ItemsSource属性绑定到图表。
下面的主题提供不同的数据绑定方法用于将数据发送到C1Chart控件的信息。

值的集合

你可以用几种方法将数据传送到图表中。一种方法是通过ValuesSource（在线文档 'ValuesSource 属性'）属性绑定一个值的集合。
任何支持IEnumerable接口的数值型值的集合可以用作数据系列的数据源。每个数据序列类型均具有适当的属性用作数据绑定。例如，DataSeries（在线文档 'DataSeries 类'）类使用ValuesSource属性用做数据绑定。
为了绑定数值的集合至DataSeries，您首先应当指定该绑定源作为一组double值类型的数组，如以下所示：

XAML

```
<!—绑定源 -->
```

```
<x:Array xmlns:sys="clr-namespace:System;assembly=mscorlib" x:Key="array" Type="sys:Double">
<sys:Double>1</sys:Double>
<sys:Double>4</sys:Double>
<sys:Double>9</sys:Double>
<sys:Double>16</sys:Double> </x:Array>
```

将数组传递给数据序列，使用下面的标记：

XAML

```
<!—绑定目标 -->
<c1chart:C1Chart Name="chart">
<c1chart:C1Chart.Data>
<c1chart:ChartData ItemsSource="{Binding Source={StaticResource array}, Path=Items}">
<c1chart:DataSeries ValuesSource="{Binding Source={StaticResource array},Path=Items}"/> </c1chart:ChartData>

</c1chart:C1Chart.Data>
</c1chart:C1Chart>
```

可以将数据值作为属性指定，这组值应当使用空格分隔，例如：

XAML

```
<c1chart:DataSeries Values="1 2 9 16"/>
```

之前的标记声明绑定DataSeries的ValuesSource属性至DataSeries对象的Items属性，给定了一个值"1 2 9 16"。

对象集合

当你有一个集合的对象，每个对象包括数字属性时，应该使用数据绑定。数据绑定过程中至少有两个图表属性参与这一过程。
ItemsSource属性 - 该对象集合分配到的源。
ValueBinding属性 - 获取或设置图表的数据系列的值绑定。指定对象的哪一个属性提供数据值。假设在资源中我们有一个点的数组。

XAML

```
x:Array x:Key="points" Type="Point">

<Point>0,0</Point>
<Point>10,0</Point>
<Point>10,10</Point>
<Point>0,10</Point>
<Point>5,5</Point>
</x:Array>
```

以下XAML代码片段表示一个具有两个数据系列的图表，一个绑定到点的X坐标，另一个绑定到该点的Y坐标：

XAML

```
<c1chart:C1Chart Name="chart2">
<c1chart:C1Chart.Data>
<c1chart:ChartData
ItemsSource="{Binding Source={StaticResource points}, Path=Items}">

<c1chart:DataSeries ValueBinding="{Binding Path=X}"/>
<c1chart:DataSeries ValueBinding="{Binding Path=Y}"/>
</c1chart:ChartData>
</c1chart:C1Chart.Data>
</c1chart:C1Chart>
```

下一个示例展示系列同时使用点的两个坐标值；请注意这里是XYDataSeries（在线文档 'XYDataSeries 类'）类处理两组分别关联到X-轴和Y-轴坐标值的数据值。

XAML

```
<c1chart:XYDataSeries
XValueBinding="{Binding Path=X}"
ValueBinding="{Binding Path=Y}"/>
```