ComponentOne

BarCode for WPF

GrapeCity US

GrapeCity
201 South Highland Avenue, Suite 301
Pittsburgh, PA 15206
Tel: 1.800.858.2739 | 412.681.4343
Fax: 412.681.4384
Website: https://www.grapecity.com/en/
E-mail: us.sales@grapecity.com

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 $25 (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

- Barcode for WPF 2
  - Getting Started 2
    - Help with WPF Edition 2
    - Key Features 2
    - Quick Start: BarCode for WPF 2
      - Step 1: Setting Up the Application 2-3
      - Step 2: Adding Code 3-5
      - Step 3: Running Your Application 5-8
  - Using BarCode for WPF 8
    - Supported Encodings 8-11
    - Customizing the C1Bar Code Control 11-12
    - Saving the C1BarCode Image 12-13
    - QR Codes 13

Barcode for WPF

Add barcode images to your applications with **BarCode for WPF**.

Unlike barcode fonts, **BarCode for WPF** automatically adds any necessary control symbols and checksums to the value being encoded, depending on the encoding being used, to eliminate reader errors.

BarCode for WPF is so easy to use – just add the control to your form, set the encoding type, and you're done!

Getting Started

Help with WPF Edition

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

Key Features

- **Supports 36 different encodings**

The C1BarCode control supports 36 encodings, including: Codabar, Code128 Auto, Code39, Code93, DataMatrix, Ean13, Ean8, PostNet, QRCode, and RSS14.

- **Automatically adds checksums**

The C1BarCode control automatically adds necessary control symbols and checksums to the value being encoded, depending on the encoding being used, to guarantee a good read on your barcodes.

- **Royalty-free DLL for easy deployment**

C1BarCode is a royalty-free DLL that can be deployed with your applications like any regular assembly.

Quick Start: BarCode for WPF

Step 1: Setting Up the Application

In this step, you begin by creating a new WPF application in Visual Studio and adding appropriate XAML markup to the MainWindow. Next, you add various controls including the C1Barcode control and other general controls like a combo box and three text blocks to set up the application.

1. Create a new WPF application in Visual Studio
2. Right-click the project in **Solution Explorer** and select **Add | New Folder**. Name the new folder **Resources**, right-click this folder, and select **Add | Existing Item**. The **Add Existing Item** dialog box appears.
3. Locate an image file to add to your application. For this example, we are using c1logo.png.
4. Select the file and click **OK**. The file gets added to the **Resources** folder.
5. Rebuild the application so that the image file is included in your application.
6. Open the **MainWindow.xaml** file and locate the **<Window></Window>** tag. Edit the tag so that it resembles the following markup to include the necessary namespaces for using C1BarCode control.

XAML copyCode

```
<Window x:Class="BarCodeApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
        xmlns:BarCode="clr-namespace:C1.BarCode;assembly=C1.WPF.BarCode.4"
```

```
        Title="BarCode Sample" Height="621.567" Width="1069.493">
```

7. Place your cursor between the **<Grid> </Grid>** tags on the MainPage.xaml and add the following markup to set up required resources for grid and defining its rows.

XAML copyCode

```
<Grid.Resources>
    <Style TargetType="TextBlock">
        <Setter Property="FontSize" Value="18"></Setter>
    </Style>
    <Style TargetType="TextBox">
        <Setter Property="FontSize" Value="18"></Setter>
    </Style>
    <Style TargetType="ComboBox">
        <Setter Property="FontSize" Value="18"></Setter>
    </Style>
</Grid.Resources>
<Border HorizontalAlignment="Center">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="20*"/>
            <RowDefinition Height="20*" />
            <RowDefinition Height="60*" />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="40*" />
            <ColumnDefinition Width="60*" />
        </Grid.ColumnDefinitions>
```

8. Add the following markup below the above markup to add three **TextBlock** controls, a **TextBox** control, and a **ComboBox** control.

XAML copyCode

```
        <TextBlock Text="CodeType:" VerticalAlignment="Center"></TextBlock>
        <ComboBox x:Name="cbCodeType"  HorizontalAlignment="Left" Grid.Column="1"
                  Width="414" Height="50"
                  SelectionChanged="cbCodeType_SelectionChanged" />
        <TextBlock Text="Text:" Grid.Row="1" VerticalAlignment="Center"></TextBlock>
        <TextBox x:Name="text" Text="{Binding Text, ElementName=barcode,
        UpdateSourceTrigger=PropertyChanged, FallbackValue='', Mode=TwoWay}"
                 HorizontalAlignment="Left" Grid.Column="1" Grid.Row="1"
                 Height="50" Width="414" TextChanged="text_TextChanged"/>

        <TextBlock Text="BarCode:" Grid.Row="2" VerticalAlignment="Center"/>
        <Grid Grid.Row="2" Grid.Column="1" Background="White" HorizontalAlignment="Left" Width="414">
        </Grid>
    </Grid>
</Border>
```

9. Add the following markup below the above markup to add and display the image file added to the **Resources** folder in Step 4 above.

XAMl copyCode

```
<Image Source="Resources/c1logo.png"
x:Name="image" Width="70" Height="70" Margin="378,310,312,124" />
```

Ensure that your markup ends with </Grid> and </Window> tags.

With this, you have set up the application for this Quick Start guide.

Step 2: Adding Code

In this step, you start by adding the interaction logic for the controls defined in the XAML markup.

Complete the following steps to add code in the MainWindow.xaml.cs file.

1. Switch to the code view that is MainWindow.xaml.cs file and add the following namespace.
    - **Visual Basic**

```
Imports C1.BarCode
```

- **C#**

```
using C1.BarCode;
```

2. Subscribe the following events within the class constructor, that is the MainWindow() constructor, one by one and press **TAB** key every time to generate event handlers for each of the subscribed event.
   - **Visual Basic**

```
AddHandler Me.Loaded, AddressOf MainWindow_Loaded
AddHandler cbCodeType.SelectionChanged, AddressOf cbCodeType_SelectionChanged
AddHandler text.TextChanged, AddressOf text_TextChanged
```

   - **C#**

```
this.Loaded += MainWindow_Loaded;
cbCodeType.SelectionChanged += cbCodeType_SelectionChanged;
text.TextChanged += text_TextChanged;
```

3. Add the following code to the **MainWindow_Loaded** event handler created in the above step. This code loads the BarCode control as soon as the MainWindow gets loaded.
   - **Visual Basic**

```
Private Sub MainWindow_Loaded(sender As Object, e As RoutedEventArgs)
    cbCodeType.ItemsSource = [Enum].GetValues(GetType(CodeType))
    cbCodeType.SelectedItem = barcode.CodeType
End Sub
```

   - **C#**

```
void MainWindow_Loaded(object sender, RoutedEventArgs e)
{
    cbCodeType.ItemsSource = Enum.GetValues(typeof(CodeType));
    cbCodeType.SelectedItem = barcode.CodeType;
}
```

4. Add the following code to the **cbCodeType_SelectionChanged** event handler created in Step 2. This code changes the type of BarCode that appears on the MainWindow when the user changes the selection in the ComboBox control.
   - **Visual Basic**

```
Private Sub cbCodeType_SelectionChanged(sender As Object, e As SelectionChangedEventArgs)
    If barcode IsNot Nothing Then
        Try
            barcode.CodeType = DirectCast(cbCodeType.SelectedItem, CodeType)
            If barcode.CodeType <> CodeType.QRCode
            OrElse Not Text.Text.Equals("http://www.componentone.com") Then
                Image.Opacity = 1
            Else
                Image.Opacity = 0
            End If
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End If
End Sub
```

   - **C#**

```
void cbCodeType_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (barcode != null)
    {
        try
        {
            barcode.CodeType = (CodeType)cbCodeType.SelectedItem;
            if (barcode.CodeType != CodeType.QRCode
                || !text.Text.Equals("http://www.componentone.com"))
            {
                image.Opacity = 1;
            }
            else
            {
                image.Opacity = 0;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
```

```
            }
        }
    }
```

5. Add the following code to the **text_TextChanged** event handler created in Step 2 to enable text change.
    - **Visual Basic**

```
Private Sub text_TextChanged(sender As Object, e As TextChangedEventArgs)
    If Not String.IsNullOrEmpty(Text.Text)
    AndAlso Text.Text.Equals("http://www.componentone.com")
    AndAlso barcode.CodeType = CodeType.QRCode Then
        Image.Opacity = 1
    Else
        Image.Opacity = 0
    End If
End Sub
```

    - **C#**

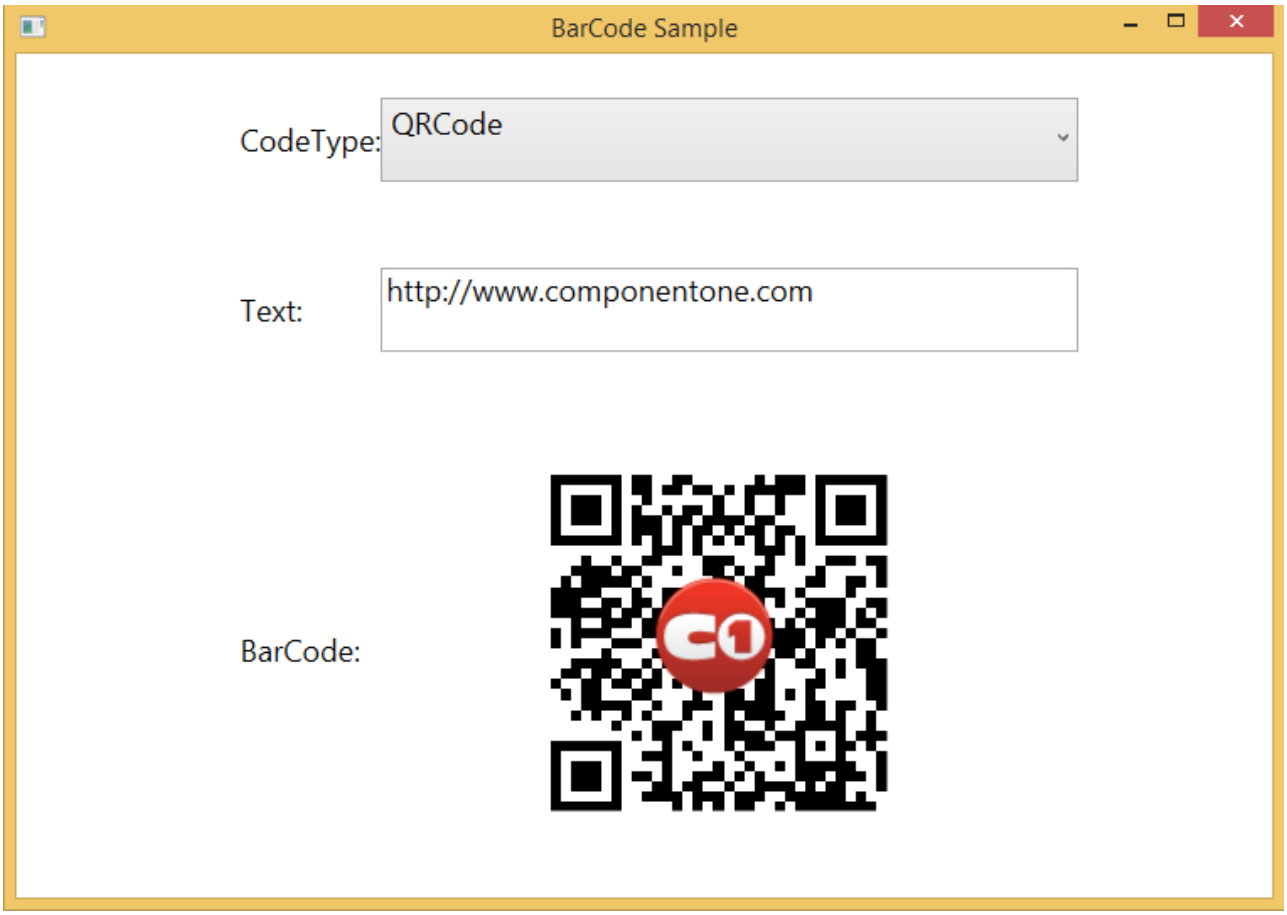
```
void text_TextChanged(object sender, TextChangedEventArgs e)
{
    if (!string.IsNullOrEmpty(text.Text) &&
        text.Text.Equals("http://www.componentone.com") &&
        barcode.CodeType == CodeType.QRCode)
        image.Opacity = 1;
    else
        image.Opacity = 0;
}
```

With this, you completed adding code to the interaction logic for XAML.

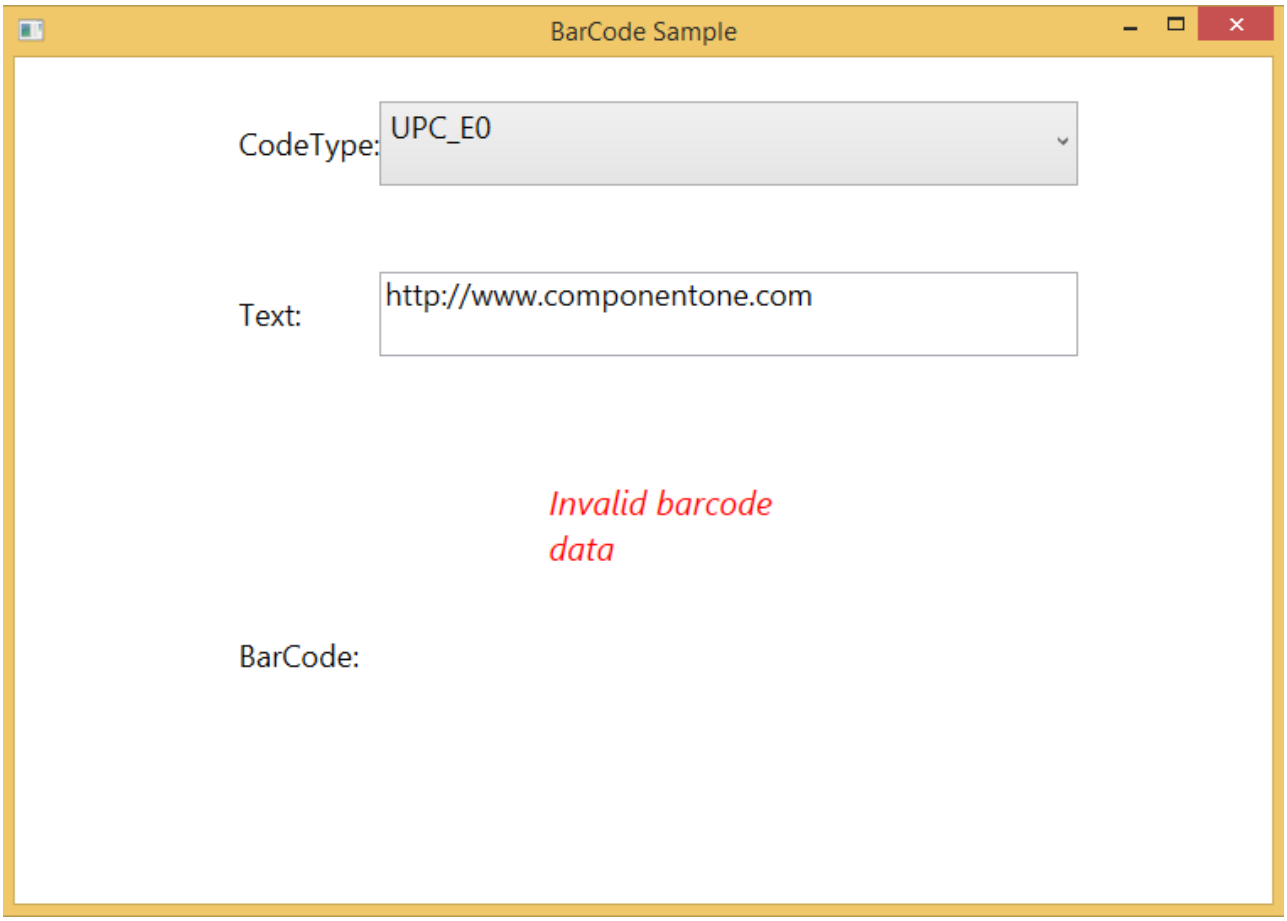
Step 3: Running Your Application

In this step, you run your application to see the control's appearance and working.

1. Select **Debug | Start Debugging** or press **F5** to run your application. The MainWindow appears similar to the following image.



2. Select a new **CodeType** from the drop down list. Since the text is set to a URL, an error message might display for many of the **CodeTypes**.



3. Finally, change the value in the **Text** TextBox.

With this, you complete all the steps of the Quick Start guide of **BarCode for WPF.**

Using BarCode for WPF

Supported Encodings

You can change the C1BarCode encoding type by setting the CodeType property. The **C1BarCode** control supports the following encodings:

Encoding	Description
BC412	The BC412 barcode was invented by IBM to meet the needs of the semiconductor wafer identification application.
Code11	Code11, also known as USD-8, is a high-density barcode symbology developed by Intermec in 1977. It is primarily used to label telecommunication equipments. This symbology is discrete and is able to encode numeric digits through 0-9, dash (-), and start/stop characters.
Code 39	Code 39 is an alpha-numeric encoding also known as 3 of 9 and LOGMARS. This was the first alphanumeric symbology developed, and is one of the most widely used encodings.
Code 39x	Code 39 Extended uses double character encoding, allowing it to support all ASCII 128 characters.
Codabar	Codabar may encode 16 different characters (0 through 9 plus -$:/.+), plus an additional 4 start/stop characters (A through D). Codabar is used by some US blood banks, photo labs, and on FedEx airbills.
Code 128A	Code 128 is a very high density alpha-numeric barcode. Code 128A uses ASCII characters 00 to 95 (0-9, A-Z and control codes), special characters, and FNC 1-4
Code 128B	Code 128 is a very high density alpha-numeric barcode. Code 128B uses ASCII characters 32 to 127 (0-9, A-Z, a-z), special characters, and FNC 1-4
Code 128C	Code 128 is a very high density alpha-numeric barcode. Code 128C uses 00-99 (encodes each two digits with one code) and FNC1.
Code 128 Auto	Code 128 Auto will encode your data with the shortest number of bars possible.
Code 2 of 5	Code 2 of 5 is a numeric only barcode. It encodes all of the information in the bars, with spaces of a fixed width.
Code 93	Code 93 is an alpha-numeric encoding that is slightly denser than code 39.
Code25intlv	Interleaved Code 2 of 5 encodes pairs of digits. The first digit is encoded in the first five bars with the second digit encoded in the five spaces.
Code39	Code 39 is an alpha-numeric encoding also known as 3 of 9 and LOGMARS. This was the first alphanumeric symbology developed, and is one of the most widely used encodings.
Code49	Code 49 is a stacked barcode that can encode the entire ASCII 128 character set.
Code93x	Code 93 Extended is based on Code 93 and can encode the entire ASCII 128 character set.
DataMatrix	Data Matrix is a high density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern.

EAN 13	EAN-13 was implemented by the International Article Numbering Association (EAN) in Europe. EAN-13 encodes a 12-digit code that consists of a 2 digit system code followed by a 5 digit manufacturer code and a 5-digit product code. The 12-digit code is followed by a checksum digit (automatically added by the control).
EAN 8	EAN-8 provides a short barcode for small packages. It encodes a 7-digit code that consists of a 2 or 3 digit system code followed by a 4 or 5 digit product code. The 7-digit code is followed by a checksum digit (automatically added by the control).
EAN128FNC1	EAN128FNC1 is a UCC/EAN-128 (EAN128) type barcode that allows you to insert a FNC1 character at any place and to adjust the bar size, etc. To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime. This is not available in UCC/EAN-128.
HIBCode39	HIBCCode39 is a Health Industry Bar Code 39 implementation.
HIBCode128	HIBCCode128 is a Health Industry Bar Code 128 implementation.
Iata25	Represents an IATA 2 of 5 barcode.
IntelligentMail	Intelligent Mail, formerly known as the 4-State Customer Barcode, is a 65-bar code used for domestic mail in the U.S.
IntelligentMailPackage	Intelligent Mail Package Barcode.
ISBN	The International Standard Book Number (ISBN) is special commercial book identifier which encodes 9 numeric digits apart from the start number "978", "979".
ISMN	The International Standard Music Number or ISMN (ISO 10957) is a thirteen-character alphanumeric identifier for printed music developed by ISO.
ISSN	The International Standard Serial Number (ISSN) is an eight-digit number used for printed or electronic periodical publications like magazines, etc. This ISSN system was drafted as an International Standard in 1971 and published as ISO 3297 in 1975.
ITF14	ITF14 barcode is the GS1 implementation of an Interleaved 2 of 5 bar code to encode a Global Trade Item Number. It is continuous, self-checking, bidirectionally decodable and it will always encode 14 digits. ITF14 is used on packaging levels of a product in general.
JapanesePostal	This is the barcode used by the Japanese Postal system. Encodes alpha and numeric characters consisting of 18 digits including a 7-digit postal code number, optionally followed by block and house number information. The data to be encoded can include hyphens.
Matrix 2of5	Matrix 2 of 5 is a higher density barcode consisting of 3 black bars and 2 white bars. Matrix 2 of 5 uses only numbers.
MicroPDF417	MicroPDF417 is two-dimensional (2D), multi-row symbology, derived from PDF417. Micro-PDF417 is designed for applications that need to encode data in a two-dimensional (2D) symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits) with the minimal symbol size.
MicroQRCode	MicroQRCode is a variant of QR Code 2005. Compared with other regular QR Codes, it has only one position detection pattern which reduces the barcode size so that it can be used to applications where the space for barcode image is severely restricted.
MSI	MSI Code uses only numbers.
Pdf417	The Pdf417 barcode is a stacked, linear barcode that consists of 3 to 90 rows. Each of these

	rows is like a small linear bar code.
Pharmacode	Pharmacode, also known as Pharmaceutical Binary Code, is a barcode standard, 1D barcode that is used in the pharmaceutical manufacturing industry as a packing control system.
Plessey	MSI barcode, also known as Modified Plessey, is a numeric symbology developed by the MSI Data Corporation, which is used primarily for marking retail shelves for inventory control. Though continuous and self-checking, MSI Plessey provides several module checksum situations.
PostNet	PostNet is a numeric encoding used by the US postal service. It differs from most others in that it is based on the height of the bars rather than on their width.
PZN	PZN or Pharma-Zentral-Nummer is a barcode standard used in the German pharmaceutical industry for identification of medicines and health-care products.
QRCode	QR codes are machine-readable, matrix barcodes. They can encode a wide variety of information, including alphanumeric data, numeric information, byte data, and kanji characters. This symbology can encode up to 7,366 characters.
RM4SCC	RM4SCC is used to encode information used by the Royal Mail for its Cleanmail service. It encodes alphanumeric information using up to 36 possible symbols: 26 letters and 10 numbers.
RSS14	RSS14 is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is a 14-digit EAN.UCC item identification for use with omnidirectional point-of-sale scanners.
RSS14 Stacked	RSS14Stacked is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is the same as RSS14Truncated, but stacked in two rows when RSS14Truncated is too wide.
RSS14 Stacked Omnidirectional	RSS14StackedOmnidirectional is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is the same as RSS14, but stacked in two rows when RSS14 is too wide.
RSS14 Truncated	RSS14Truncated is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is a 14-digit EAN.UCC item identification plus Indicator digits for use on small items, not for point-of-sale scanners.
RSS Expanded	RSSExpanded is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is a 14-digit EAN.UCC item identification plus AI element strings (expiration date, weight, etc.) for use with omnidirectional point-of-sale scanners.
RSS Expanded Stacked	RSSExpandedStacked is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is the same as RSSExpanded, but stacked in two rows when RSSExpanded is too wide.
RSS Limited	RSS Limited is a Reduced Space Symbology that encodes Composite Component (CC) extended EAN and UPC information in less space. This version is a 14-digit EAN.UCC item identification with indicator digits of 0 or 1 in a small symbol that is not scanned by point-of-sale scanners.
SSCC 18	Serial Shipping Container Code-18 (SSCC-18) Barcode is a type of barcode that can print in the lower 2-inch (or local equivalent) extended area of the Thermal 4" x 8" or 4" x 8¼" (or local equivalent) label.
Telepen	Telepen is a name of a barcode symbology designed in the UK, in 1972, to directly represent

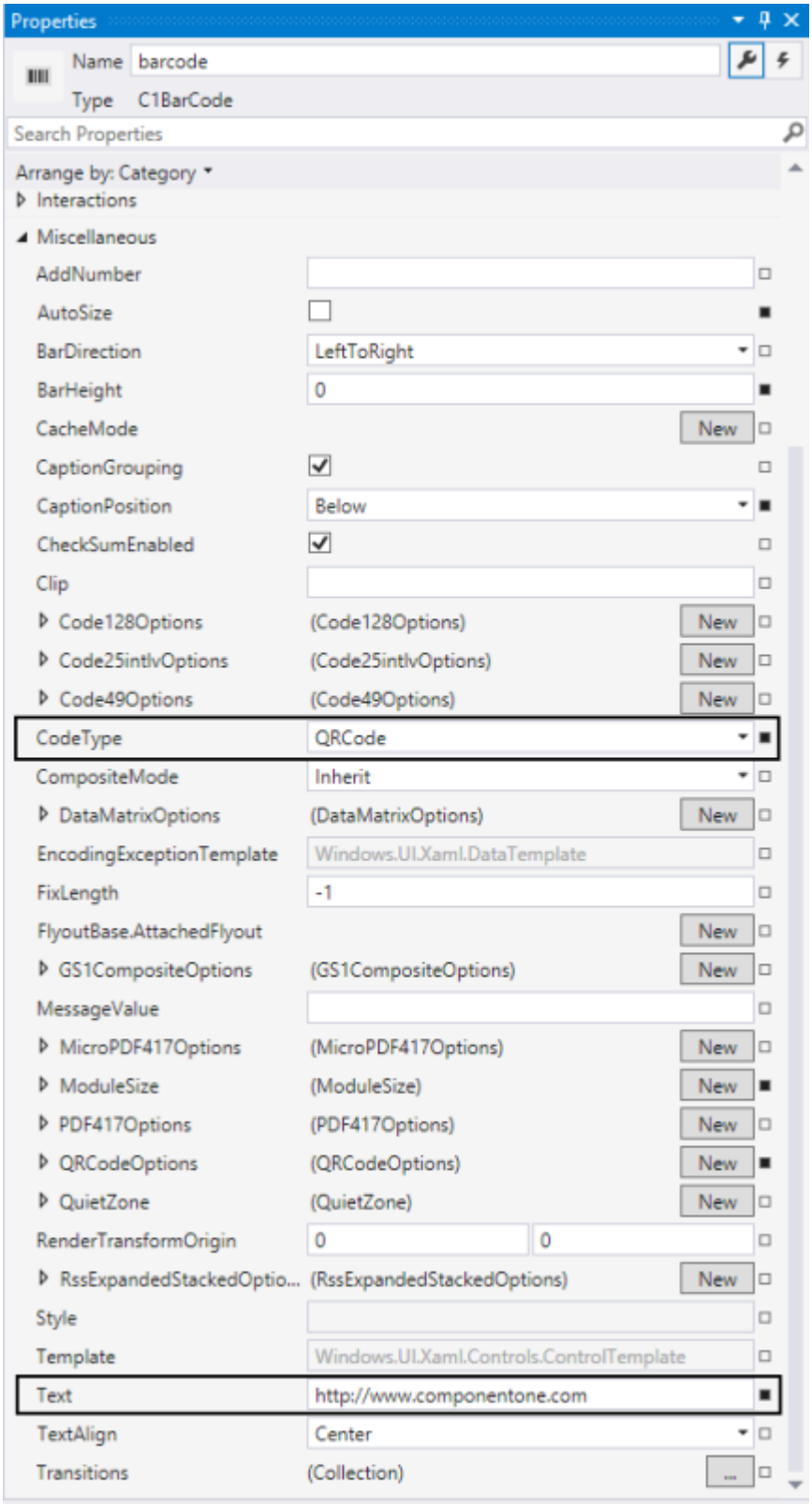
	the full ASCII character set without using shift characters for code switching, and use only two different widths for bars and spaces.
UCC/EAN-128	UCC/EAN –128 uses the complete ASCII character Set. This is a special version of Code 128 used in HIBC applications.
UPC A	UPC-A is the common encoding you will find on virtually every consumer good on the shelves of your local supermarket, as well as books, magazines, and newspapers. It is similar to EAN-13, and encodes 11 digits of numeric data along with a trailing check digit.
UPC E0	UPC-E0 uses only numbers. Used for zero-compression UPC symbols. For the Caption property, you may enter either a six-digit UPC-E code or a complete 11-digit (includes code type, which must be 0 (zero)) UPC-A code. If an 11-digit code is entered, the Barcode control will convert it to a six-digit UPC-E code, if possible. If it is not possible to convert from the 11-digit code to the six-digit code, nothing is displayed.
UPC E1	UPC-E1 uses only numbers. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters.

Customizing the C1Bar Code Control

To use the C1BarCode control, set the CodeType property to the type of encoding you want to use, then set the Text property to the value you want to encode.

> **Note**: Some encodings have a minimum character requirement, while others will only work with numeric values.

The following image shows the C1BarCode control set to the **QRCode** CodeType, and the Text set to an URL. Depending on the type of barcode used in an application, there will be more options available for customization.



Saving the C1BarCode Image

You can save the image of your BarCode control and use it for other purposes. The C1BarCode class provides Save() method that can be used for saving the image of the barcode control in an appropriate image format including Bmp, Png and Jpeg.

To save the C1BarCode as a .Png file, complete the following steps.

The code example given below uses the sample created in the **Quick Start** section.

1. Navigate to the **Toolbox** and add a button control to the designer. This adds a button control to the designer and a **<Button>** tag in the XAML view.
2. Edit the **<Button>** tag to set the properties of the added button control in XAML as illustrated in the following code.

```
XAML copyCode
<Button x:Name="button" Content="Save BarCode Image" HorizontalAlignment="Left" VerticalAlignment="Top"
                    Width="138" Margin="502,510,0,0" RenderTransformOrigin="0.266,0.099" Height="51"/>
```

3. Subscribe the **button_Click** event within the class constructor, that is the MainWindow() constructor, as illustrated in the following code.
   - **Visual Basic**

```
AddHandler button.Click, AddressOf button_Click
```

   - **C#**

```
button.Click += button_Click;
```

4. Press **TAB** key twice to generate handler for the **button_Click** event.
5. Add the code to the **button_Click** event handler as illustrated below.
   - **Visual Basic**

```
Private Sub button_Click(sender As Object, e As RoutedEventArgs)
    Using stm = System.IO.File.Create("../Pictures/barcode.jpg")
        barcode.Save(stm, ImageFormat.Jpeg)
    End Using
End Sub
```

   - **C#**

```
void button_Click(object sender, RoutedEventArgs e)
{
    using (var stm = System.IO.File.Create("../Pictures/barcode.jpg"))
    {
        barcode.Save(stm, ImageFormat.Jpeg);
    }
}
```

   In the code above, you need to specify the path or location in **Create()** method where the image is to be saved on button click .

QR Codes

The QR code (Quick Response code) format is one of the most popular 2D barcode formats available today, with free readers available for virtually all smart phones. Developed by the DENSO-WAVE company, the QR code format is efficient and compact, it doesn't require a special scanner to read it, and it is an open and freely available standard (ISO/IEC18004 and others).

The code is simply made up of black and white pixels arranged in patterns. With the C1QRCode control, the QR patterns are based on the value you want to encode, which you can specify in the Text property.