# QR Code Barcode

*https://www.barcoderesource.com/qrcodebarcode.shtml*

# Table of Contents

# 1. QR Code Barcode

## 1.1 QR Code Barcode

The QR Code (Quick Response) barcode is a 2-dimensional barcode consisting of black square patterns on a white background. The barcode is capable of storing more information than a conventional barcode. It is developed by Denso-Wave in Japan and is one of the more popular 2-dimensional barcodes. Another reason for this barcode popularity is because it is adopted by many mobile or smartphone applications for linking physical world objects to a web URL (Uniform Resource Locator).

## 1.2 ConnectCode QR Code Barcode Font

This is a professional True Type (TTF) barcode font that is used to create a QR Code barcode by selecting a font in your favourite text editor. The package includes a standalone encoder, a .Net Dynamic Link Library (DLL), true type font for creating a QR Code barcode that strictly adheres to industry specifications.

### 1.2.1 Data Compaction

The QR Code is able to pack large amount of data using the various compaction methods. Each of the compaction method is optimized for a specific type of data. For example, the Numeric method is optimized for numbers. The ConnectCode encoder automatically scans through the data and detects the most optimized compaction method. On top of that, it also switches among the different compaction methods if one method is unable to fully encode the data.

- Numeric - Optimized for numbers.
- Alphanumeric - Optimized for numbers and alphabets. This compaction method is less optimized than Numeric.
- Binary - Optimized for any 8-bit binary data.
- Kanji - Optimized for Kanji data.

## 1.3 Parameters of the QR Code Barcode

The following sections detail the different configurable parameters of the QR Code barcode. If you are new to this barcode, it is recommended that you use the default or automatic settings mentioned below.

### 1.3.1 Error Correction

QR Code uses the Reed-Solomon error correction technique. This allows the barcode to be partially damaged without causing any loss of data. There are four different levels of error correction that can be chosen. The higher the level, the more resilient the barcode is to withstand damage. However the drawback is that more data codewords in the barcode are needed to store the error correction codewords instead of the actual data.

- Level L - 7% of codewords can be recovered
- Level M - 15% of codewords can be recovered
- Level Q - 25% of codewords can be recovered
- Level H - 30% of codewords can be recovered

### 1.3.2 Mask

QR Code's reliability can be improved by a method called masking. Masking regularizes the distribution of the black square patterns. Different types of masking patterns according to the specifications are supported by the ConnectCode QR Code. The default automatic masking option is recommended if you do not want to delve into the technical implementation of the barcode.

- Mask 0 – Mask Pattern 0
- Mask 1 – Mask Pattern 1
- Mask 2 – Mask Pattern 2
- Mask 3 – Mask Pattern 3
- Mask 4 – Mask Pattern 4
- Mask 5 – Mask Pattern 5
- Mask 6 – Mask Pattern 6
- Mask 7 – Mask Pattern 7
- Auto – Automatic Masking

## 1.4 QR Code Barcode Fonts

The following is the description of the QR Code barcode font used by the encoder or .Net DLL.

| Font Name | Description | Recommended Sizes |
|---|---|---|
| **CCodeQR**<br><br>(CCodeQR_Trial for the Trial version) | Standard QR Code Barcode Font. | Font Size 2..64 |

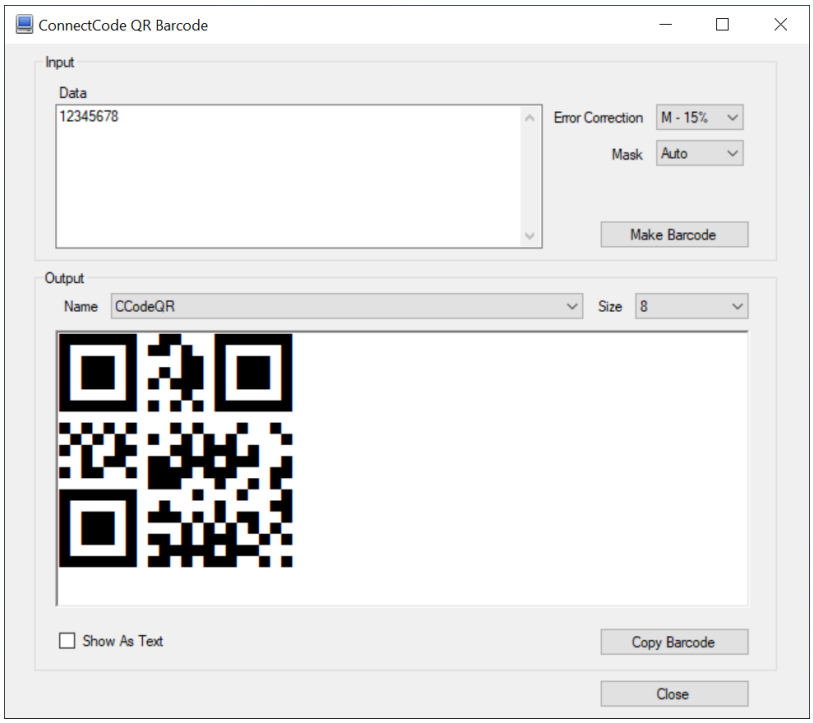
**Note**

1. You may see spaces between multiple Rows when you use the QR Code barcode fonts in certain font sizes. The spaces can be easily removed by increasing or decreasing the font size by 1 point.

# 2. Font Encoder

The QR Code Barcode Font package in ConnectCode comes bundled with a Font Encoder that allows you to encode the barcode quickly and easily. This is useful if you like to encode a single barcode to be pasted into your brochure, on packaging or product items. The Encoder supports all parameters as described in the sections above.

The Error Level and Mask are parameters that are supported (see previous section for detailed description).




The Font Name and Font Size in the "Output" section can be changed after the QR Code barcode is created. This allows the height and the size of the barcode to be changed after the barcode is created.

The "Show As Text" option allows you to see the text output of the barcode in a normal text font. The "Copy Barcode" button allows the barcode to be copied and pasted onto other applications easily.

# 3. .NET SDK

A .NET Barcode SDK is also bundled in the ConnectCode QR Code Barcode Font package. This SDK can be bundled in your applications if you purchase the necessary distribution licenses.

**Library Name**

QRCode.dll

**Namespace**

ConnectCode.BarcodeFonts2D

**Class Name**

QRCode

**Requirements**

.NET 2.0 and onwards

**Constructors and Functions**

**QR(String data, String errorcorrectionlevel, int mask);**

This is the constructor for the QR barcode. It is used for initializing the QR barcode.

data : The data input string to be encoded as a barcode.
errorcorrectionlevel : The Error Correction Level "L", "M", "Q" or "H".
mask : Mask Pattern. Any number from 0..7 or 8 for Auto.

**String Encode();**

This function encodes the barcode based on the parameters specified in the constructor. The result will be returned as a string.

**int LengthExceeded();**

The Encode() function may return an empty output string either due to invalid inputs or the length of the data exceeded the length specified by the QR Code specifications. A call to this function after the Encode() function allows you to determine whether the data length has exceeded.

**Sample Usage (C#)**

```
Using ConnectCode.BarcodeFonts2D;
.
.
.
QR barcode = new QR("12345678",”M”,0);
String result = barcode.Encode();
Font font = new Font("CCodeQR", 8);
richTextBox1.Text = outputstr; //private System.Windows.Forms.RichTextBox richTextBox1;
richTextBox1.SelectAll();
richTextBox1.SelectionFont = font;
```

**Sample Visual Studio Project**

- Name - ConnectCode Encoder
- Solution Name - ConnectCode.sln
- Language - C#
- Requirements - .NET 2.0 and onwards, Visual Studio 2008, 2010, 2012, 2013, 2017 and onwards.

## 3.1 .NET Framework 4.0 Notes

.NET Framework 4.0 includes and uses CLR 4.0. It does not automatically use its version of the common language runtime to run applications that are built with earlier versions of .NET Framework. This is unlike .NET 2.0-3.5 where the framework uses CLR 2.0 to run applications. Basically, there is no version 3 of the CLR.

Hence, ConnectCode 2D Barcode SDK provides two sets of .NET DLLs for different versions of the .Net Framework as shown below:

For .NET 2.0 to 3.5 please use the DLLs and samples in

- /Resource subdirectory
- /.Net Samples subdirectory

For .NET 4.0 please use the DLLs and samples in

- /Net4 subdirectory
- /Net4/.Net Samples subdirectory

# 4. .NET Standard SDK

.NET Framework is a software framework that is developed by Microsoft to run primarily on Microsoft Windows. Over the years, the framework has been forked and enhanced to serve many different purposes. For example, the Universal Windows Platform uses a specific set of APIs from the .NET Framework to help programmers develop apps for the Windows Store. The .NET Core Framework also uses a subset of APIs from the .NET Framework to support the development of applications that can run on different operating systems such as Windows, Mac, and Linux.

.NET Standard is a specification of common APIs that are available on the different .NET frameworks. By creating a class library (DLL) that targets the .NET Standard, a developer can be assured that his library can be used or shared by projects developed on the various .NET frameworks.

As of .NET Standard v2.0, the .Net Standard specification is implemented by the following frameworks:

- .NET Core
- .NET Framework
- Mono
- Xamarin.iOS
- Xamarin.Android
- Universal Windows Platform

**.NET Standard QR Code Library**

- Resource/NETStandard/netstandard2.0/NETStandardQRCode.dll

The QR Code Barcode package includes a .NET Standard compliant QR Code class library (also available as a nuget) that targets the .NET Standard 2.0 specification. This library can be used by projects developed on different .NET frameworks and when used together with ConnectCode Barcode Fonts, generates barcodes of the highest quality that is able to meet the strictest requirements of the auto-id industry.

**Sample ASP.NET Core Sample**

- Resource/NETStandard/ASPNETCoreQRCodeSample

# 5. Windows UI (WinUI)

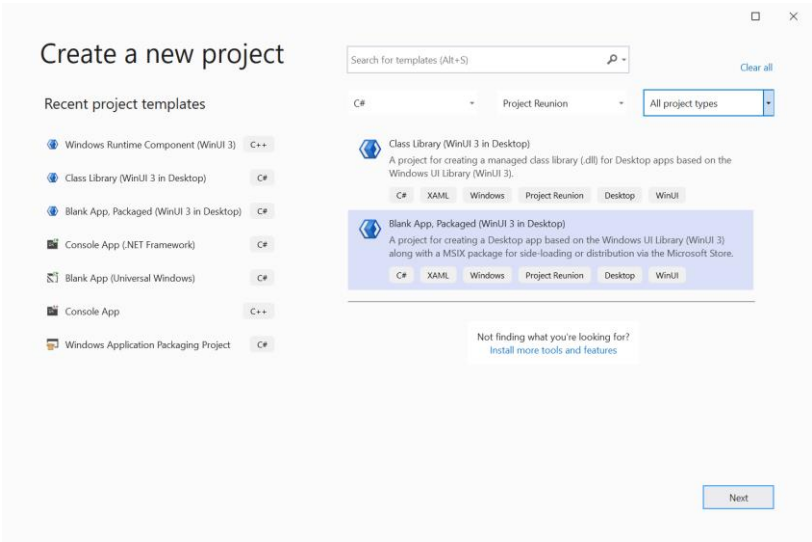
In this tutorial, we are going to illustrate how to create a standards-compliant QR Code barcode in a WinUI (Windows UI Library) Desktop project. WinUI is a user experience framework that unites Win32 and UWP developers to a common API for developing apps with modern user interfaces in Windows. The creation of a QR Code barcode involves the use of a .NET Standard 2.0 class library for encoding input data and a Microsoft WinUI Canvas for displaying barcode. The class library ensures the the QR Code compliance with the ISO/IEC 18004:2015 standards by validating input data, generating Error Correction Codewords, and applying necessary QR Code mask patterns.

The class library returns a text string in the format of ones (1 - Black) and zeros (0 - White). This string provides the flexibility of rendering QR Code in components such as a WinUI Canvas (reference code in this tutorial), other .NET Canvas, or System.Drawings. The QR Code can also be displayed by placing the text string into a WinUI TextBlock, TextBox, or RichTextBlock, and then applying a QR Code font (part of ConnectCode QR Code package). Finally, many modern document formats such as Microsoft Word and Adobe PDF support font embedding. This means a Line of Business (LOB) app can generate a document that includes a high-quality QR Code barcode by just using a text string and a QR Code font embedded into the document.

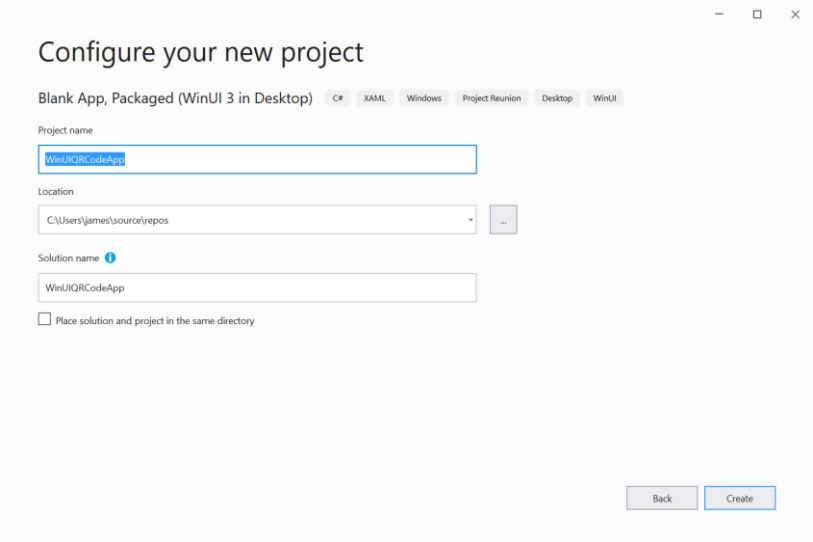
**Prerequisite**

- ConnectCode QR Code package is installed
- Visual Studio 2019 16.9 (or onwards)
- WinUI 3.0 (Project Reunion 0.5 or onwards)

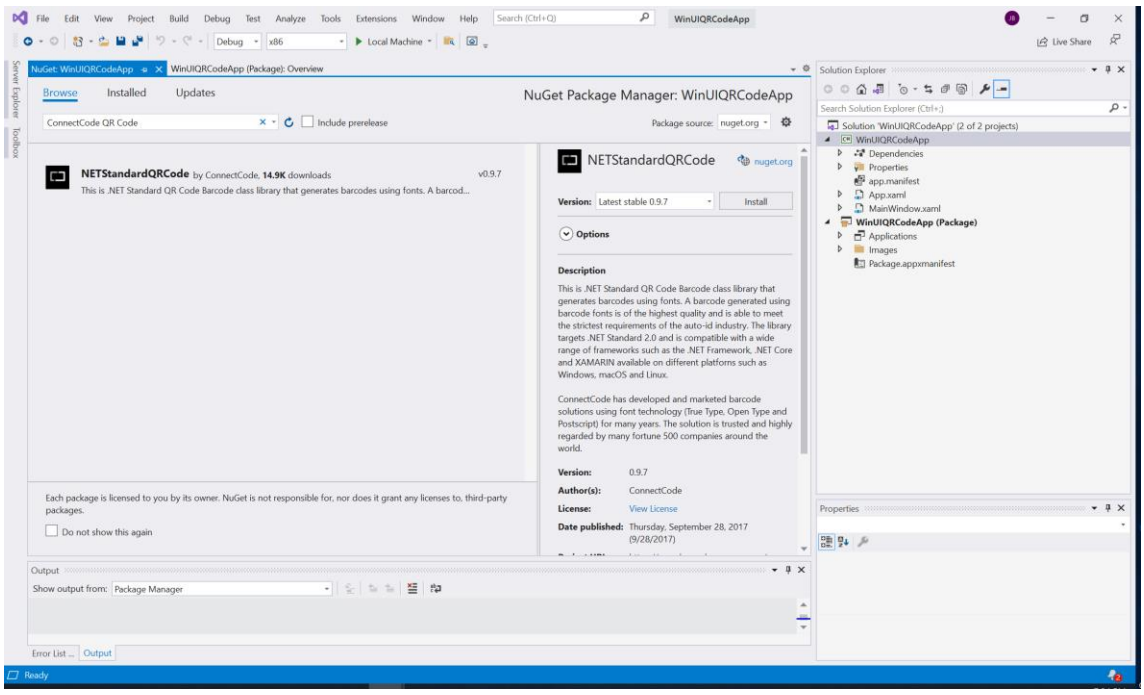
1. Launch Visual Studio and create a new project. Select the "Blank App, Packaged (WinUI in Desktop)" C# template and click on the Next button.



2. In Configure your new project dialog, enter "WinUIQRCodeApp" as Project name, leave everything else as default values, and click on the Create button. A Visual Studio solution containing WinUIQRCodeApp and WinUIQRCodeApp (Package) is created.




3. Next, we add the QR Code class library available in nuget.org to the project. In the Solution Explorer, right-click on WinUIQRCodeApp and select Manage Nuget Packages. In the Nuget Package Manager shown below, click on Browse and search for "ConnectCode QR Code". Select "NETStandardQRCode" and click on Install to add the class library.



4. Add the QR Code font into the WinUIQRCodeApp Visual Studio project by right-clicking on WinUIQRCodeApp project in Solution Explorer. Select Add -> New Folder and name the folder as "Fonts". Next, right-click on "Fonts" folder, select Add -> Existing Item and navigate to the following folder:

```
C:\Program Files (x86)\ConnectCodeQRCode

(or C:\Program Files (x86)\ConnectCodeQRCodeTrial if you are using the Trial version)
```

Select the QR Code font:

```
CCodeQR.ttf (or CCodeQR_Trial.ttf)
```

5. We have added a QR Code class library and a QR Code font to the project. As described in the overview, we are going to generate a QR Code using the library and then display it on a WinUI Canvas or with a barcode font. Click on MainPage.xaml in Solution Explorer and change the XAML to the following:

```
<Window
   x:Class="WinUIQRCodeApp.MainWindow"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:local="using:WinUIQRCodeApp"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   mc:Ignorable="d">

   <StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center">
      <Button x:Name="myButton" Click="myButton_Click">Click Me
      <TextBlock x:Name="QRCodeBarcode"
        Margin="10">
      <Canvas x:Name="QRCodeCanvas" Margin="10">
   </StackPanel>

</Window>
```

The TextBlock is used to display the text string returned by the class library. We will apply the font to this text string later in the tutorial.

6. Next, we add the C# code to encode the input data. Click on MainWindow.xaml.cs and add the following to the top of the file.

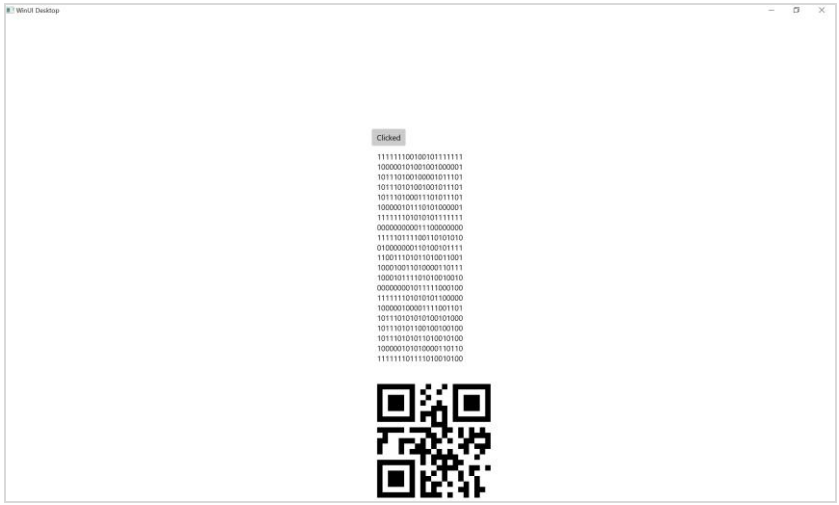
```
using Net.ConnectCode.BarcodeFontsStandard2D;
```

Locate myButton_Click function and add the following to the function.

```
myButton.Content = "Clicked";
QR qr = new QR("12345678","L",8);

//ECL L-7% M-15% Q-25% H-30%
//Mask 0..7 or 8 for Auto
//There are 8 masks patterns to apply to a QR Code
//The purpose is to make the QR Code easier to read for the scanners.

string qrCodeStr = qr.Encode();
QRCodeBarcode.Text = qrCodeStr;

int squareD = 10;
double posX = 0;
double posY = 0;

foreach (char c in qrCodeStr)
{
   var rect = new Microsoft.UI.Xaml.Shapes.Rectangle();
   rect.Width = squareD;
   rect.Height = squareD;
   Canvas.SetLeft(rect, posX);
   Canvas.SetTop(rect, posY);
   if (c=='1')
   {
      rect.Fill = new SolidColorBrush(Microsoft.UI.Colors.Black);
      posX = posX + squareD;
   }
   else if (c=='0')
   {
      rect.Fill = new SolidColorBrush(Microsoft.UI.Colors.White);
      posX = posX + squareD;
   }
   else if (c == '\n')
   {
      posX = 0;
      posY = posY + squareD;
   }
   QRCodeCanvas.Children.Add(rect);
}
```

The code above uses the Net.ConnectCode.BarcodeFontsStandard2D.QR class to generate a QR Code barcode text string using the input data "12345678", an ECL (Error Correction Level) of "L", and auto QR Code masks. A higher ECL setting generates a QR Code that can be recovered even when it is partially damaged. However, this means more redundancy is added to the QR Code, resulting in a larger QR Code in size. If you are not sure of which ECL to use, we recommend starting with "M" which provides a good balance between redundancy and size.

7. Run the app in Visual Studio, click on "Click Me" button to see the generated QR Code.




The first part after the button shows the text string of the QR Code class library. The second part shows the QR Code rendered on a WinUI Canvas.

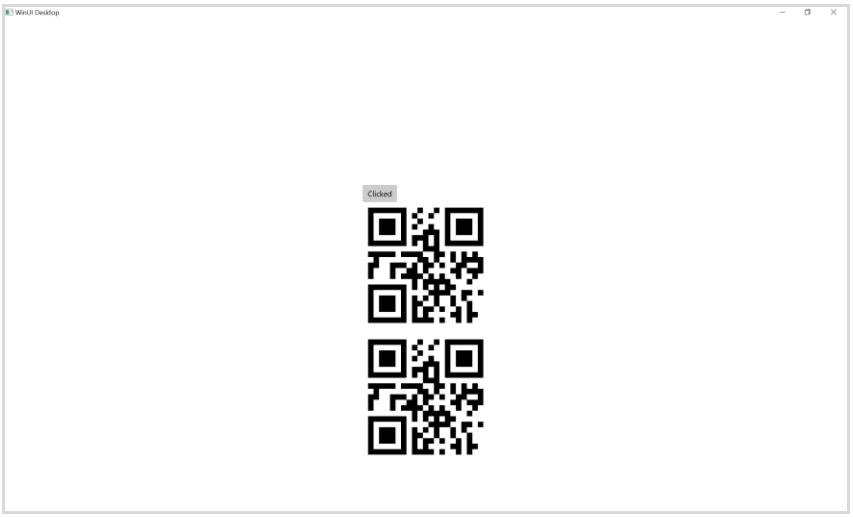
8. We can change the raw text string above into a QR Code by applying a QR Code font. Change the code in MainWindow.xaml to the following:

```
<Window
   x:Class="WinUIQRCodeApp.MainWindow"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:local="using:WinUIQRCodeApp"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   mc:Ignorable="d">

   <StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center">
      <Button x:Name="myButton" Click="myButton_Click">Click Me
      <TextBlock x:Name="QRCodeBarcode"
               FontFamily="/Fonts/CCodeQR.ttf#CCodeQR"
               FontSize="10"
               Margin="10">
      <Canvas x:Name="QRCodeCanvas" Margin="10">
   </StackPanel>
</Window>
```

If you are testing with the Trial version of the QR Code font, use the following:

```
               FontFamily="/Fonts/CCodeQR_Trial.ttf#CCodeQR_Trial"
```

Run the application again and you should see the following:



# 6. Blazor

In this tutorial, we illustrate how to create an ISO/IEC 18004:2015 compliant QR Code barcode in a Blazor WebAssembly app. Blazor is a framework developed by Microsoft for building interactive web apps using C#, .NET, and HTML. The framework supports apps hosted on ASP.NET Core Server (Blazor Server), and Single-page apps (Blazor WebAssembly) that are downloaded onto your web browser before running. Both scenarios are supported by the method for QR Code generation described in this tutorial.

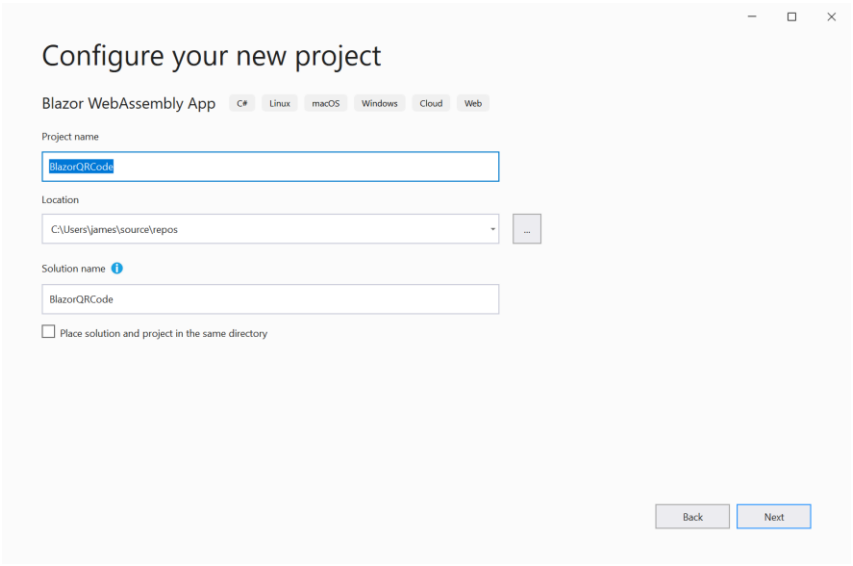
The method involves the use of a .NET Standard 2.0 class library for encoding input data and applying a barcode font to the result for display. This has the advantage of having no dependencies on any underlying graphics API. The class library validates input data, generates Error Correction Codewords, and applies necessary QR Code mask patterns before returning a text string of ones (1 - Black) and zeros (0 - White). In this tutorial, we are using Visual Studio on Windows to demonstrate the QR Code generation process. It is possible to create QR Code using the same method with Visual Studio Code on other platforms.

**Prerequisite**

- ConnectCode QR Code package is installed
- Visual Studio 2019 16.6 (or onwards)

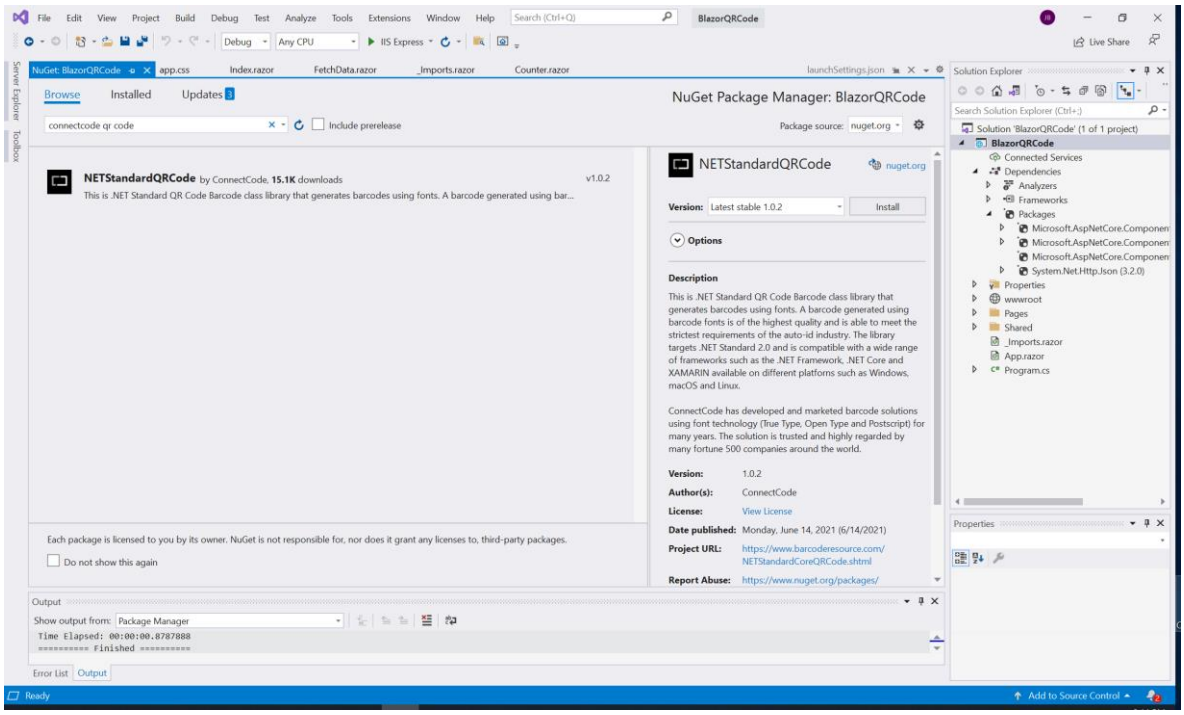
1. Launch Visual Studio and click on Create a new project. Select the "Blazor WebAssembly App" and click on Next button.

2. In Configure your new project dialog, enter "BlazorQRCode" as Project name, leave everything else as default values, and click on the Next button. A Visual Studio solution containing WinUIQRCodeApp and WinUIQRCodeApp (Package) is created.




In Additional Information dialog, choose .NET Core 3.1 (LTS) target framework and then the Create button. You can also choose .NET 5 as the Target Framework if you like to.

3. After the project is created, add the QR Code class library available in nuget.org to the project. In Solution Explorer, right-click on BlazorQRCode and select Manage Nuget Packages. In Nuget Package Manager shown below, click on Browse and search for "ConnectCode QR Code", select "NETStandardQRCode", and click on Install to add the class library (select version 1.0.2 or above).




4. Next, add the QR Code font into the project. In Solutions Explorer, right-click on wwwroot, select Add -> New Folder, and name the folder as "fonts". Next, right-click on "fonts" folder, select Add -> Existing Item and navigate to the following folder:

C:\Program Files (x86)\ConnectCodeQRCode\Resource\Fonts

(or C:\Program Files (x86)\ConnectCodeQRCodeTrial\Resource\Fonts if you are using the Trial version)

Select the Web Open Format (WOFF) QR Code font:

CCodeQR.woff (or CCodeQR_Trial.woff)

If your web project requires compatibility with older browsers that do not support WOFF, you can also add the Open Type font in the same folder.

5. In Solutions Explorer -> Pages, click on Index.razor, and add the following code.

```
<div id="qrBarcodeFonts">@((MarkupString)qrCodeStr)</div>

@code {

   private string qrCodeStr = "";
   protected override async Task OnInitializedAsync()
   {
      QR qr = new QR("12345678", "M", 8);
      qr.NewLine = "<br />";
      qrCodeStr = qr.Encode();
   }
}
```

The razor code defines qrCodeStr with qrBarcodeFonts CSS style. qrCodeStr is initialized with the result returned by the QR Code class library. For the code above, we are generating a QR Code with "12345678" as the input, "M" (L, M, Q, and H) as the Error Correction Level, and "8" (Auto) as the Mask Pattern.

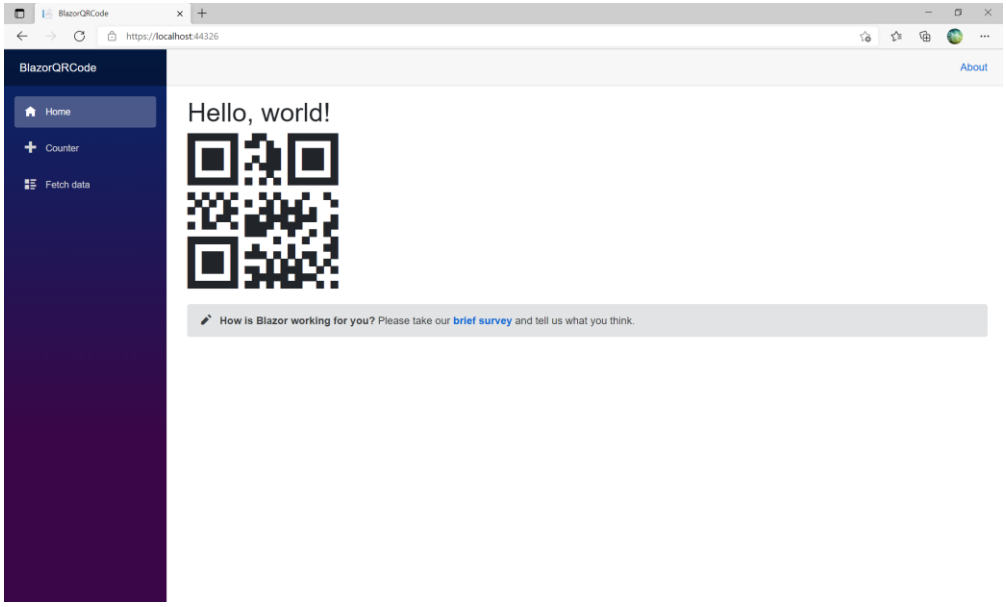
If you are using the class library in a server or in a native application, the library will automatically detect the environment to return \n or \r\n using Microsoft Environment.NewLine. In our scenario, we are using the class library in a HTML environment, so we overwrite the NewLine character with the HTML br tag.

6. The last step is to apply QR Code barcode font to the text string result returned by the class library. We have already specified qrBarcodeFonts style for the div element in the previous step, thus we just need to define this style in our CSS. In Solutions Explorer, click on css -> app.css, and add the following CSS definitions into the file.

```
@font-face {
   font-family: CCodeQR;
   src: url("../fonts/CCodeQR.woff") format("woff");
}

#qrBarcodeFonts {
   font-weight: normal;
   font-style: normal;
   line-height: normal;
   font-family: 'CCodeQR', sans-serif;
   font-size: 12px;
   letter-spacing: -1px;
   line-height: 98%;
}
```

Some browsers automatically add a very small space between characters vertically and horizontally. This may result in the square patterns in the QR Code looking spaced out. This issue can be removed by specifying the letter-spacing and line-height CSS property.

7. In Visual Studio, click on IIS Express in the toolbar to run the application. You should see the following Blazor WebAssembly application with a standards-compliant QR Code barcode.



# 7. JavaScript SDK

This is an elegant solution that enables you to generate high quality QR Code barcodes using JavaScript with a HTML5 Canvas or World Wide Web Consortium (W3C) compliant Web Fonts. The generated QR Code barcode is able to meet the strictest industry requirements required by the auto-id industry.

**JavaScript QR Code Resource Folder**

- Resource/Javascript

The snippet below illustrates the use of the HTML5 Canvas tag on a HTML page.

```
<canvas id="barcodeCanvas" width=300 height=300>12345678</canvas>
```

The following JavaScript code shows how to render a QR Code barcode on the HTML5 Canvas.

```
var elementBarcode = document.getElementById("barcodeCanvas");
var barcode = new QRCode(elementBarcode.innerHTML, "L", 8);
barcode.drawOnCanvas("barcodeCanvas");
```

Parameters

Input Data: elementBarcode.innerHTML (or any other input string)

Error Correction Level: "L" ("L", "M", "Q" or "H")

L - Allows recovery of up to 7% data loss
M - Allows recovery of up to 15% data loss
Q - Allows recovery of up to 25% data loss
H - Allows recovery of up to 30% data loss

Mask: 8 (0 to 7 or 8 for Auto)

The purpose of a mask pattern is to make the QR code easier for QR scanner to read.

## 7.1 QR Code Barcode with JavaScript and Barcode Web Fonts

The output generated by the JavaScript library can also be rendered as a QR Code Barcode through the use of a Web Open Font Format Font (WOFF). WOFF is an optimized font format recommended by World Wide Web Consortium (W3C) for use in web pages. It uses compression on Open Type or True Type fonts to achieve file size reduction so that it can be efficiently distributed over the web.

The QR Code Barcode WOFF fonts provided by ConnectCode have been tested vigorously to display and print on different desktop and mobile browsers. It is important to know that a font raster to the output device and is not limited to DPI (Dots per Inch) of the computer screen. This enables very high quality QR Code barcodes to be generated.

The font solution for generating barcodes is based on ConnectCode's True Type barcode font engine that has passed numerous independent audits and is widely adopted by many Fortune 500 companies. QR Code fonts in Embedded Open Type (EOT), and Open Type (OTF) format are also provided to ensure that the solution works on legacy browsers that have yet to fully support WOFF.

# 8. Component Object Model Library

This tutorial illustrates the use of a COM (Component Object Model) object library with a True Type Font (QR Code Barcode Font), provided in ConnectCode QR Code package, to create a ISO/IEC 18004:2015 standard-compliant QR Code in a .NET Windows Form application.

**Prerequisites**

- ConnectCode QR Code package is installed
- QRCodeCOMLibrary.dll in the Resource\QRCodeCOMLibrary subdirectory of ConnectCode QR Code package. The QR Code class library has been compiled with the "Register for COM interop" Visual Studio project property, exposing a COM-callable wrapper that enables COM interaction.
- Visual Studio 2015/2017
- Administrator Rights

## 8.1 Tutorial on creating QR Code using COM

1. Launch the Visual Studio Developer Command Prompt as Administrator from the Windows Start Menu.

2. In the Developer Command Prompt, use the "cd" command to go to the QR Code COM Library folder.

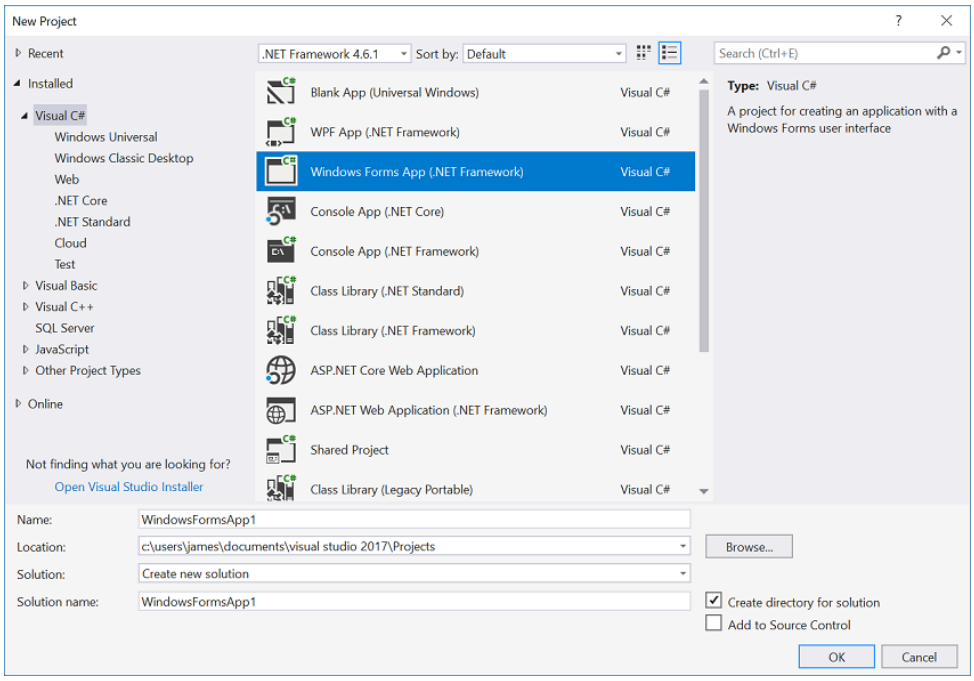
```
cd C:\Program Files (x86)\ConnectCodeQRCode\Resource\QRCodeCOMLibrary
(or ConnectCodeQRCodeTrial if you are using the Trial package)
```

3. Enter the following command in the Developer Command Prompt to use Regasm.exe to register the QRCode COMLibrary assembly for use with COM.

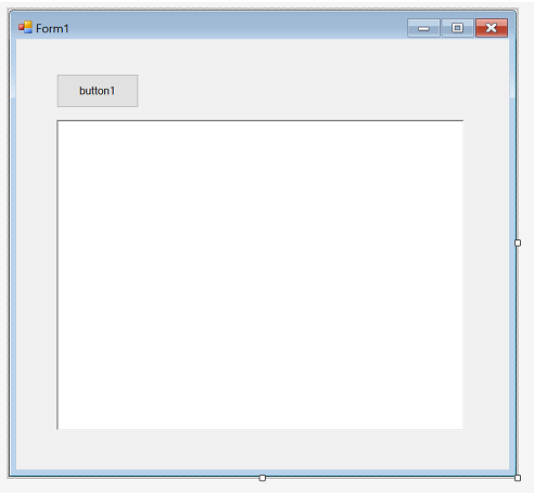
```
Regasm QRCodeCOMLibrary.dll /tlb:QRCodeCOMLibrary.tlb /codebase
```

Regasm.exe adds information about the class to the system registry so that COM clients can use the .NET Framework class transparently. The tlb option generates a type library defined within the assembly.

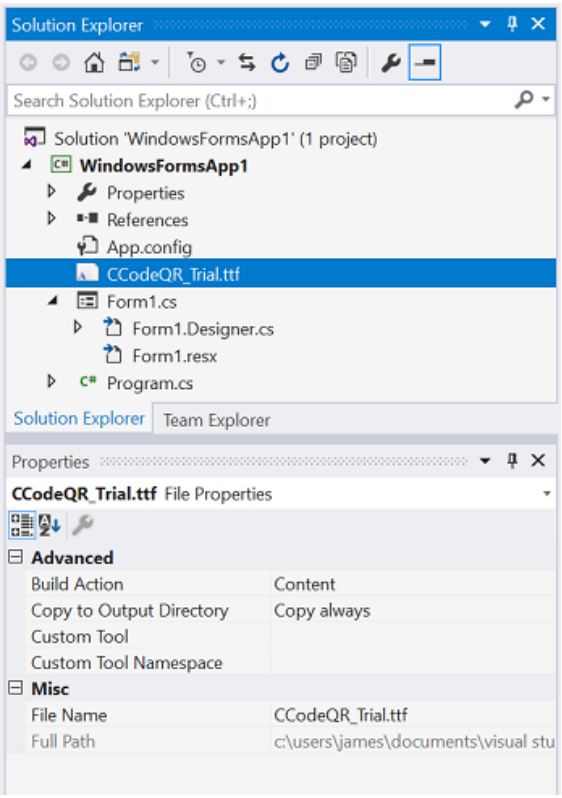
4. Launch Visual Studio. Create a new Windows Form project by clicking on "File->New Project", select a "Windows Forms App" and click on "Create" button.



5. Double click on "Form1.cs" in the "Solution Explorer" and add a "Button" and a "RichTextBox" from the Visual Studio Toolbox. You should see your Windows Form similar to the screenshot below.




6. Right click on the "WindowsFormApp1" project in the "Solution Explorer" and select "Add->Existing Item". Navigate to the "C:\Program Files (x86)\ConnectCodeQRCode\" folder and select the "CCodeQR.ttf" font. If you are using the trial version, select the "CCodeQR_Trial.ttf" font instead.

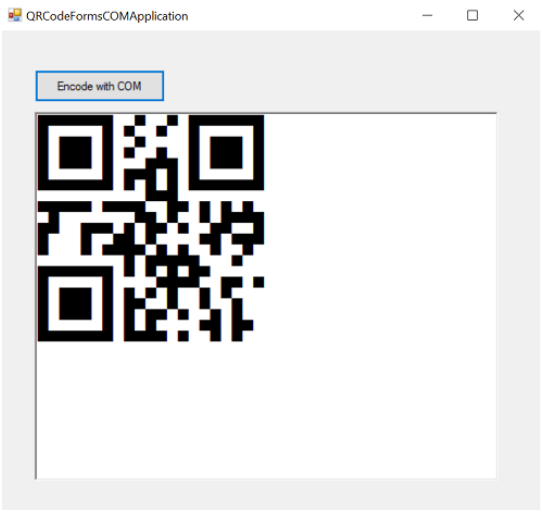


7. In "Solution Explorer", select the "CCodeQR.ttf" font and change the "Build Action" to "Content" and "Copy to Output Directory" to "Copy Always" in the Properties pane. This will ensure that Visual Studio deploy the QR Code barcode font for use with the Windows Form application.

8. Double click on "Form1.cs" in the "Solution Explorer". In the designer, double click on the Button. This will generate the button1_Click function in the editor. Enter the C# programming codes as shown below:

```
using System.Drawing.Text;
.
.
.
.
private void button1_Click(object sender, EventArgs e)
{

        try
        {

           Type comObjectType =
             Type.GetTypeFromProgID("Net.ConnectCode.QRCodeCOMLibrary");
           dynamic theComObject = Activator.CreateInstance(comObjectType, false);

           //or
           //Guid myGuid = new Guid("5E206D5A-D9C2-45AA-BBFD-2E9B36AD437D");
           //Type comObjectType = Type.GetTypeFromCLSID(myGuid);
           //dynamic theComObject = Activator.CreateInstance(comObjectType, false);

           string inputData = "12345678";
           string ecl = "L"; //L, M, Q or H
           int mask = 8; //0 to 7 or 8 for Auto
           string result1 = theComObject.Encode_QRCode(inputData,ecl,mask);
           string result = result1;
           richTextBox1.Text = result;
           System.Diagnostics.Debug.WriteLine(result1);
           PrivateFontCollection pfc = new PrivateFontCollection();
           pfc.AddFontFile("CCodeQR.ttf"); //pfc.AddFontFile("CCodeQR_Trial.ttf");
           richTextBox1.Font = new Font(pfc.Families[0], 8, FontStyle.Regular);

        }
        catch (Exception ex)
        {
           System.Diagnostics.Debug.WriteLine(ex);

        }

}
```

The C# function above creates a QR Code COM object and then uses it to generate a QR Code barcode with the input data "12345678". The result is placed in "richTextBox1" and the final output is displayed with the CCodeQR True Type font. When you run the application, click on the "Encode with COM" button, you should see the QR Code barcode as shown below.




The "QRCodeCOMApplication" folder in "C:\Program Files (x86)\ConnectCodeQRCode\Resource" contains the full source code of the above application.

# 9. PowerBuilder

This tutorial illustrates the use of a COM (Component Object Model) library and a barcode font available in ConnectCode QR Code package for creating QR Code barcode in PowerBuilder. The generated QR Code complies with the ISO/IEC 18004:2015 standards and is able to meet the strictest requirements of the Auto-ID industry.

**Prerequisites**

- PowerBuilder v12 (or APPEON PowerBuilder 2017. In 2016, SAP and Appeon has entered into an agreement whereby Appeon would be responsible for developing and marketing PowerBuilder.)
- ConnectCode QR Code package is installed

## 9.1 Tutorial on creating a QR Code in PowerBuilder

1. Launch a Windows Command Prompt as Administrator. In the Command Prompt, enter the following command to go to the QRCodeCOMLibrary folder.

```
cd C:\Program Files(x86)\ConnectCodeQRCode\Resource\QRCodeCOMLibrary
```

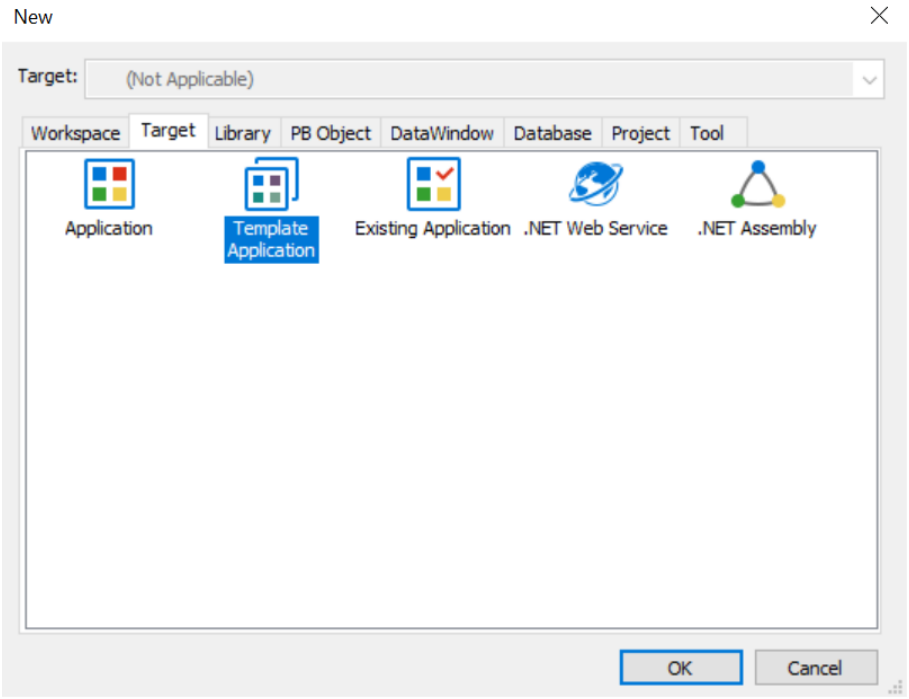
2. Next, use the Assembly Registration Tool (Regasm.exe) to register the QRCodeCOMLibrary.dll assembly with COM.

```
Regasm QRCodeCOMLibrary.dll /tlb:QRCodeCOMLibrary.tlb /codebase
```

If Regasm is not available, you can verify if the following folder exists and add the folder into your PATH. The folder may be different depending on your version of .NET.

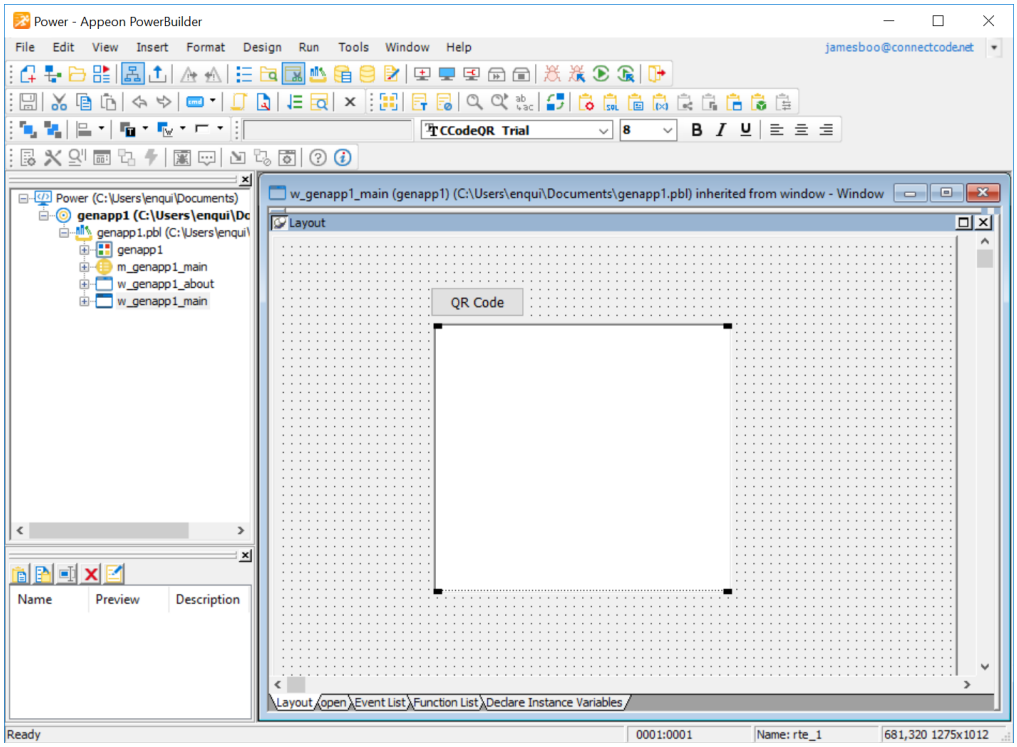
```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\
```

3. Launch PowerBuilder and create a new Template Application in the Target tab.




4. You can select “SDI application” as the "Application Type" and “None” in Connectivity Options” to create a sample application.

5. Click on w_genapp_main when the application is created. From the Menu select "Insert->Control->CommandButton" and rename the button to "QR Code". Next, insert a TextEdit control and layout the components as shown in the screenshot below.




Select the TextEdit control, change the Font to **CCodeQR** (or CCodeQR_Trial) and the Font Size to 8 to fit the barcode nicely on the TextEdit control. The output generated by the QRCodeCOMLibrary will be placed in the TextEdit control and displayed as a barcode with the CCodeQR True Type font.

6. Next, double click on the button and add the following script:

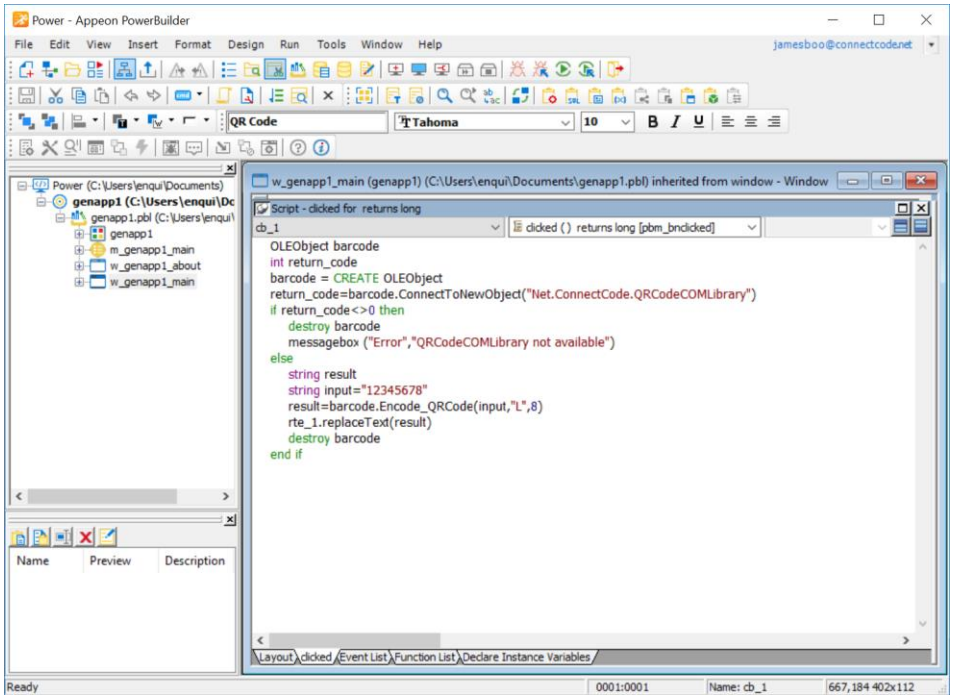
```
OLEObject barcode
int return_code
barcode = CREATE OLEObject
return_code=barcode.ConnectToNewObject("Net.ConnectCode.QRCodeCOMLibrary")
if return_code<>0 then
        destroy barcode
        messagebox ("Error","QRCodeCOMLibrary not available")
else
        string result
        string input="12345678"
        result=barcode.Encode_QRCode(input,"L",8)
        rte_1.replaceText(result)
        destroy barcode
end if
```

In the source code above, an OLE object is created with the QRCodeCOMLibrary. The "Encode_QRCode" method is used to generate the barcode.

Error Correction Level: "L" ("L", "M", "Q" or "H")

L - Allows recovery of up to 7% data loss
M - Allows recovery of up to 15% data loss
Q - Allows recovery of up to 25% data loss
H - Allows recovery of up to 30% data loss

Mask: 8 (0 to 7 or 8 for Auto)



7. Run the application by going to the menu and select “Run->Select and Run”. Click on the "QR Code" button and see that you get the QR Code barcode output in the TextEdit control. If you get an error message saying that "QRCodeCOMLibrary is not available", check that you have carried step 2 successfully.

# 10. Crystal Reports UFL

This tutorial illustrates the use of a UFL (User Function Library for Crystal Reports) with a True Type Font (QR Code Barcode Font), provided in ConnectCode QR Code package, to create a ISO/IEC 18004:2015 standard-compliant QR Code barcode in Crystal Reports. A User Function Library is a dynamic link library that enables Crystal Reports to add customized functions to Formula Workshop.

**Prerequisites**

Crystal Reports 2016

- Crystal Reports 2016 (the UFL works on earlier versions as well)
- ConnectCode QR Code package is installed
- CRUFL_QRCodeBarcode.dll in the Resource\CrystalReportsUFL subdirectory of ConnectCode QR Code package.
- x86 Native Tools Command Prompt
- Administrator Rights

Crystal Reports 2020

- Crystal Reports 2020
- ConnectCode QR Code package is installed
- CRUFL_QRCodeBarcode.dll in the Resource\CrystalReportsUFL\x64 subdirectory of ConnectCode QR Code package.
- x64 Native Tools Command Prompt
- Administrator Rights

Note – x86/x64 Native Tools Command Prompt is available in the Desktop Development with C++ Workload of Visual Studio.

## 10.1 Tutorial on creating QR Code with Crystal Reports UFL

1. Launch Windows Explorer and go to the ConnectCode QR Code package folder. The QR Code package is installed in "C:\Program Files (x86)\ConnectCodeQRCode" folder by default.

**Crystal Reports 2016**

In the "Resource\CrystalReportsUFL" subfolder, locate the "CRUFL_QRCodeBarcode.dll" and "crw32.exe.config" files, and copy the DLL to the Crystal Reports Library folder. The Crystal Reports Library folder is in a folder like the one below.

```
C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win32_x86
```

The "crw32.exe.config" file contains the following to enable mixed-mode assembly.

```
<?xml version ="1.0"?>
<configuration>
        <startup useLegacyV2RuntimeActivationPolicy="true" >
        <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

**Crystal Reports 2020**

In the "Resource\CrystalReportsUFL\x64" subfolder, locate the "CRUFL_QRCodeBarcode.dll" and "crw32.exe.config" files, and copy the DLL to the Crystal Reports Library folder. The Crystal Reports Library folder is in a folder like the one below.

```
C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win64_x64
```

The "crw32.exe.config" file contains the following to enable mixed-mode assembly.

```
<?xml version ="1.0"?>
<configuration>
        <startup useLegacyV2RuntimeActivationPolicy="true" >
        <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

2. For Crystal Reports 2016, launch the "x86 Native Tools Command Prompt for VS 2017/2019" (the 32-bit prompt is required for loading a UFL to Crystal Reports 2016) as Administrator from the Windows Start Menu.

For Crystal Reports 2020, launch the "x64 Native Tools Command Prompt for VS 2017/2019" (the 64-bit prompt is required for loading a UFL to Crystal Reports 2020) as Administrator from the Windows Start Menu.

3. In the Command Prompt, use the "cd" command to go to the Crystal Reports Library folder.

Crystal Reports 2016

```
cd C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win32_x86
```

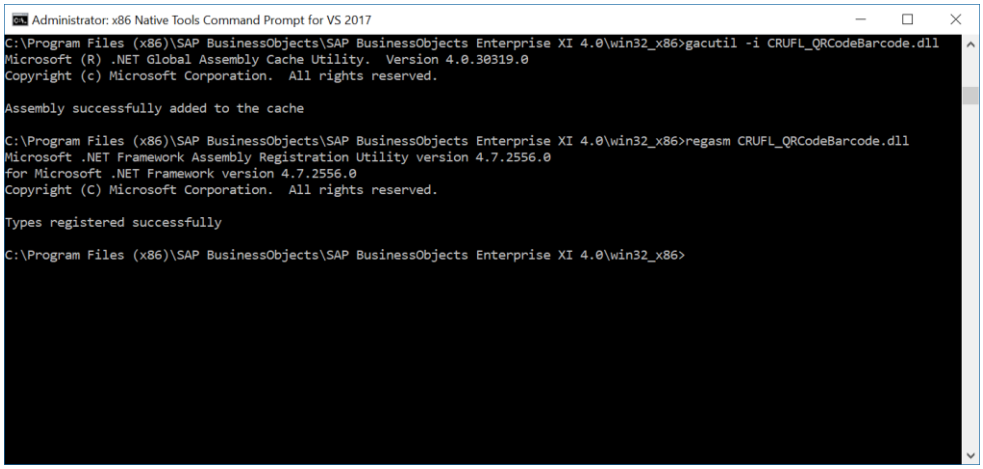
Crystal Reports 2020

```
cd C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0\win64_x64
```

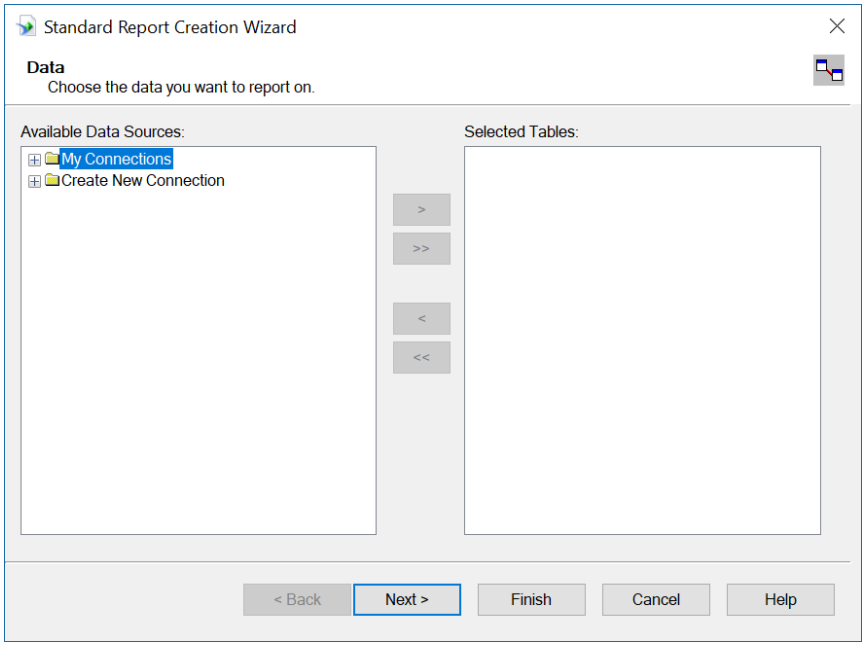
4. Enter the following command in Command Prompt to load the UFL.

```
gacutil -i CRUFL_QRCodeBarcode.dll

Regasm CRUFL_QRCodeBarcode.dll
```

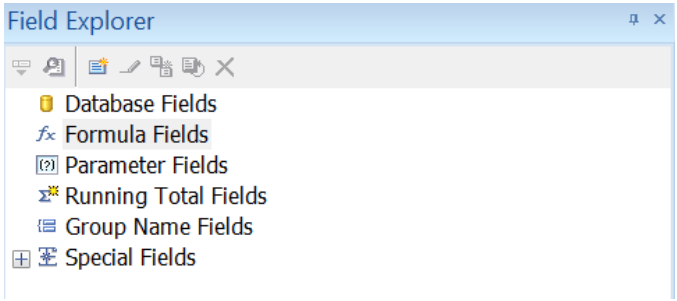
Ensure that the UFL is loaded successfully as shown in the screenshot below.



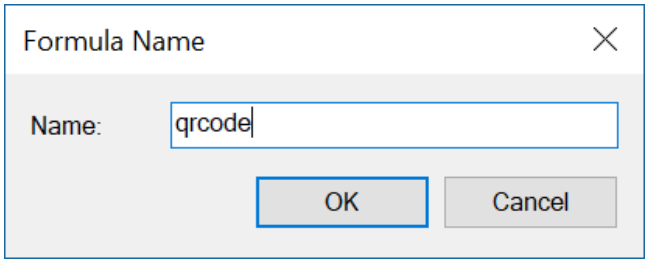

5. Launch Crystal Report and create a new "Standard Report". Click on the Finish button when prompted to select the data that you want to report on.



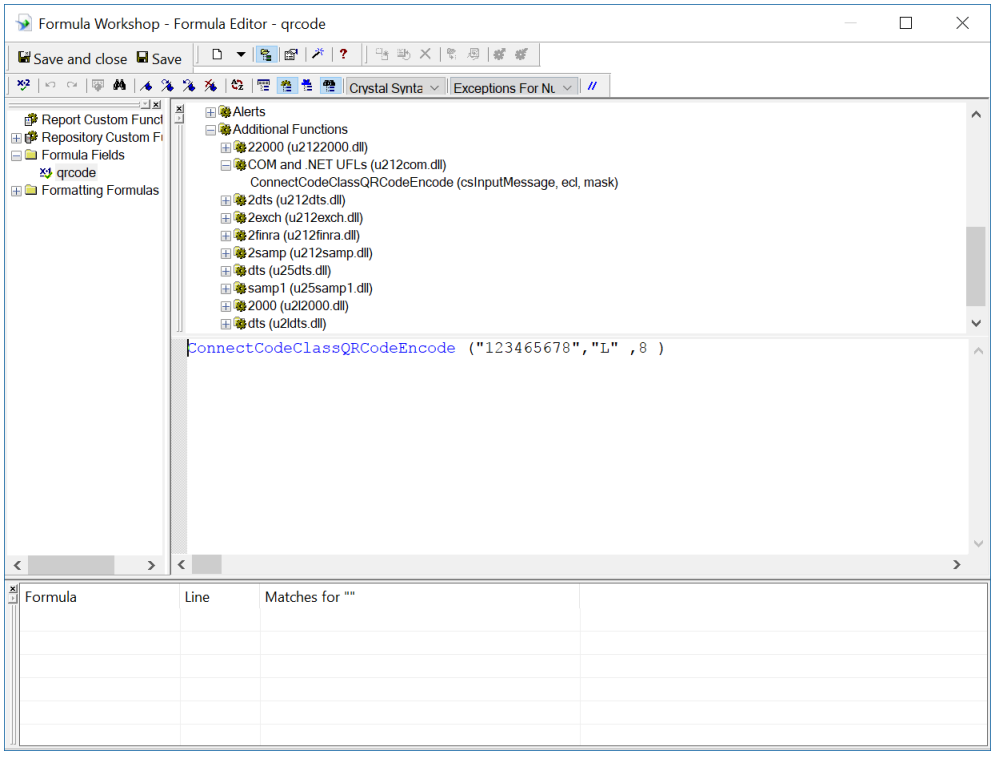
6. When the report is created, right click on "Formula Fields" in the "Field Explorer" and select "New" to create a new formula.




7. Name the formula as "qrcode".




8. In the "Formula Workshop", expand "Functions->Additional Functions->COM and .NET UFLs (u212com.dll)". Check that you see the "ConnectCodeClassQRCodeEncode" formula. If you do not see this formula, please ensure that you have run steps 2-4 successfully.



9. Double click on the formula, change “Crystal Syntax”to “Basic Syntax” and enter the following VBA programming codes below:

```
ConnectCodeClassQRCodeEncode("12345678","L",8)
Dim x As Number
Dim Result As String
For x = 1 To ConnectCodeClassQRCodeNumBlocks()
      Result=Result + ConnectCodeClassQRCodeGetBlocks(x)
Next x
formula = Result
```

The formula uses "12345678" as the input data, "L" as the error correction level and 8 as the Mask.

Error Correction Level: "L" ("L", "M", "Q" or "H")

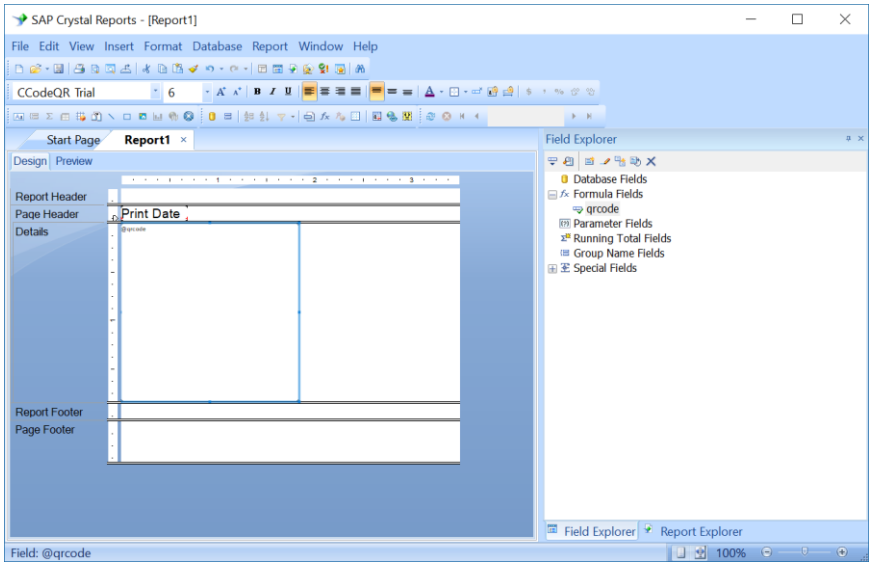
L - Allows recovery of up to 7% data loss
M - Allows recovery of up to 15% data loss
Q - Allows recovery of up to 25% data loss
H - Allows recovery of up to 30% data loss

Mask: 8 (0 to 7 or 8 for Auto)

The purpose of a mask pattern is to make the QR code easier for QR scanner to read.

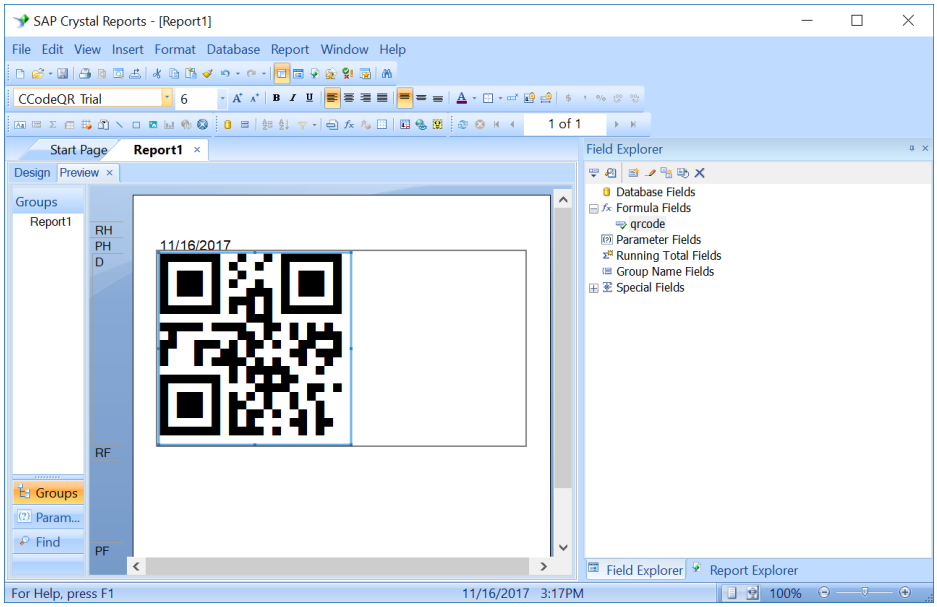
**After the QR Code is encoded, a loop is required to return the output in blocks. The reason is because Crystal Reports enforces a 255 characters length limit on the output returned by a UFL formula.**

10. When ready, click on the "Save and close" button. In the designer, drag the "qrcode" formula onto the report.




On the Design tab, select the object created and change the font to "CCodeQR" (or "CCodeQR_Trial"). Change the Font Size to 6 to fit the barcode nicely on the report.

11. Click on "View->Print Preview" to preview the report with the QR Code barcode.



# 11. Microsoft Reporting Services

SQL Server Data Tools (SSDT) is a modern development tool for building SQL Server relational databases, Azure SQL databases, Analysis Services data models, Integration Services packages, and Reporting Services reports. This tutorial illustrates how to create industrial quality barcodes in a Reporting Services Report with SSDT using Visual Studio.

**Prerequisites**

- Visual Studio 2017 or 2019 installed
- SQL Server Data Tools (SSDT) standalone or SSDT for Visual Studio installed. The SSDT component can be found in Visual Studio installer -> Workloads -> Data storage and processing -> SSDT.
- SQL Server 2017 or 2019 installed
- AdventureWorks database (or other databases) installed
- Reporting Services installed
- ConnectCode QR Code package installed

## 11.1 Configuring Visual Studio

**Copying "QRCodeLibrary.dll"**

You need to copy "QRCodeLibrary.dll" (Resource\ReportingServices subdirectory) to the "SSRS" directory of Visual Studio.

C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\CommonExtensions\Microsoft\SSRS

or

C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\CommonExtensions\Microsoft\SSRS

You will also need to copy "QRCodeLibrary.dll" to the "PrivateAssemblies" directory of Visual Studio.

C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\PrivateAssemblies

or

C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\PrivateAssemblies

**Editing "RSPreviewPolicy.config" to grant FullTrust permission**

The "RSPreviewPolicy.config" is the preview policy file of Report Designer.

```
C:\Program Files (x86)\Microsoft Visual
Studio\2019\Enterprise\Common7\IDE\CommonExtensions\Microsoft\SSRS\RSPreviewPolicy.config

or

C:\Program Files (x86)\Microsoft Visual
Studio\2017\Enterprise\Common7\IDE\CommonExtensions\Microsoft\SSRS\RSPreviewPolicy.config
```

The following tag needs to be added to the "RSPreviewPolicy.config" file to grant FullTrust permission to the "QRCodeLibrary.dll". This is required for previewing a Report that uses a DLL in Report Designer.

```
<CodeGroup class="UnionCodeGroup" Name="QRCodeBarcodeFonts" version="1"
PermissionSetName="FullTrust"
Description="This code group grants QRCodeLibrary.dll FullTrust permission.">
<IMembershipCondition class="UrlMembershipCondition" version="1"
Url="C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\
CommonExtensions\Microsoft\SSRS\QRCodeLibrary.dll"/>
</CodeGroup>

or

<CodeGroup class="UnionCodeGroup" Name="QRCodeBarcodeFonts" version="1"
PermissionSetName="FullTrust"
Description="This code group grants QRCodeLibrary.dll FullTrust permission.">
<IMembershipCondition class="UrlMembershipCondition" version="1"
Url="C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\
CommonExtensions\Microsoft\SSRS\QRCodeLibrary.dll"/>
</CodeGroup>
```

## 11.2 Configuring Reporting Services

At a file level SQL Server Reporting Services 2017 is now completely separated from SQL Server file structure. This differs from previous versions of Reporting Services and SQL Server.

**Copying "QRCodeLibrary.dll"**

You need to copy "QRCodeLibrary.dll" from Resource\ReportingServices subdirectory to the "ReportServer\bin" directory of SQL Server Reporting Services.

```
C:\Program Files\Microsoft SQL Server Reporting Services\SSRS\Reporting
Services\ReportServer\bin
```

**Editing "rssrvpolicy.config" to grant FullTrust permission**

The "rssrvpolicy.config" is the Report Server policy configuration file.

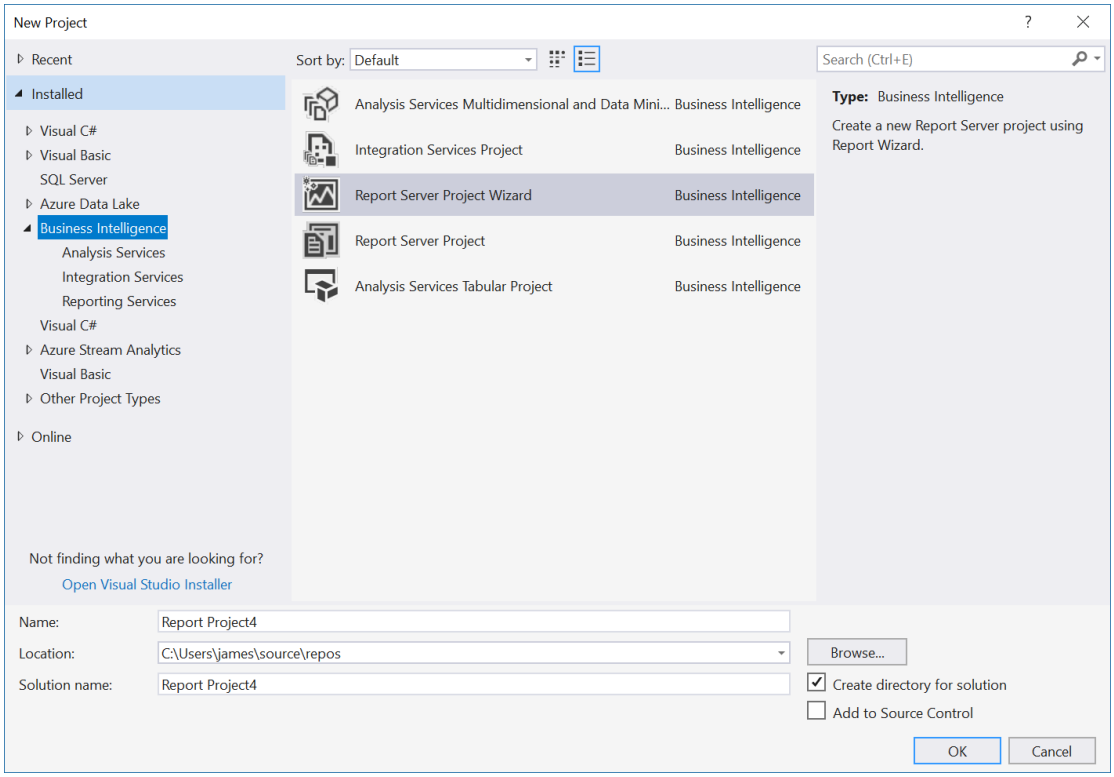
```
C:\Program Files\Microsoft SQL Server Reporting Services\SSRS\Reporting
Services\ReportServer\rssrvpolicy.config
```

The following tag needs to be added to the "rssrvpolicy.config" file. It grants FullTrust permission to "QRCodeLibrary.dll" in Report Server.

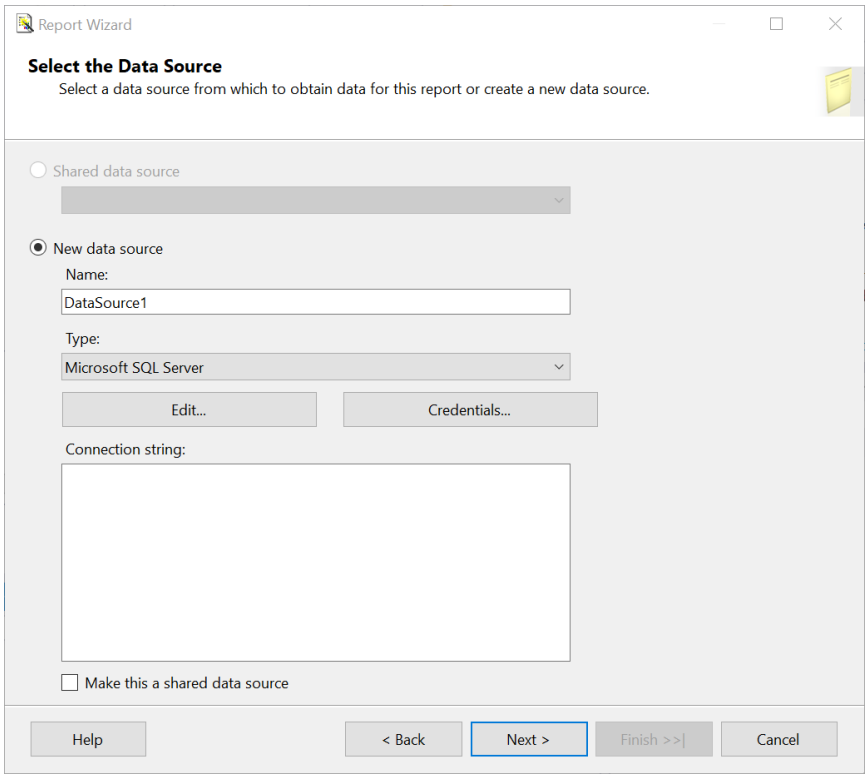
```
<CodeGroup class="UnionCodeGroup" Name="QRCodeBarcodeFonts" version="1"
PermissionSetName="FullTrust"
Description="This code group grants QRCodeLibrary.dll FullTrust permission.">
<IMembershipCondition class="UrlMembershipCondition" version="1"
Url="C:\Program Files\Microsoft SQL Server Reporting Services\SSRS\
Reporting Services\ReportServer\bin\QRCodeLibrary.dll"/>
</CodeGroup>
```

## 11.3 Create barcodes in a Microsoft Reporting Services (SSDT)

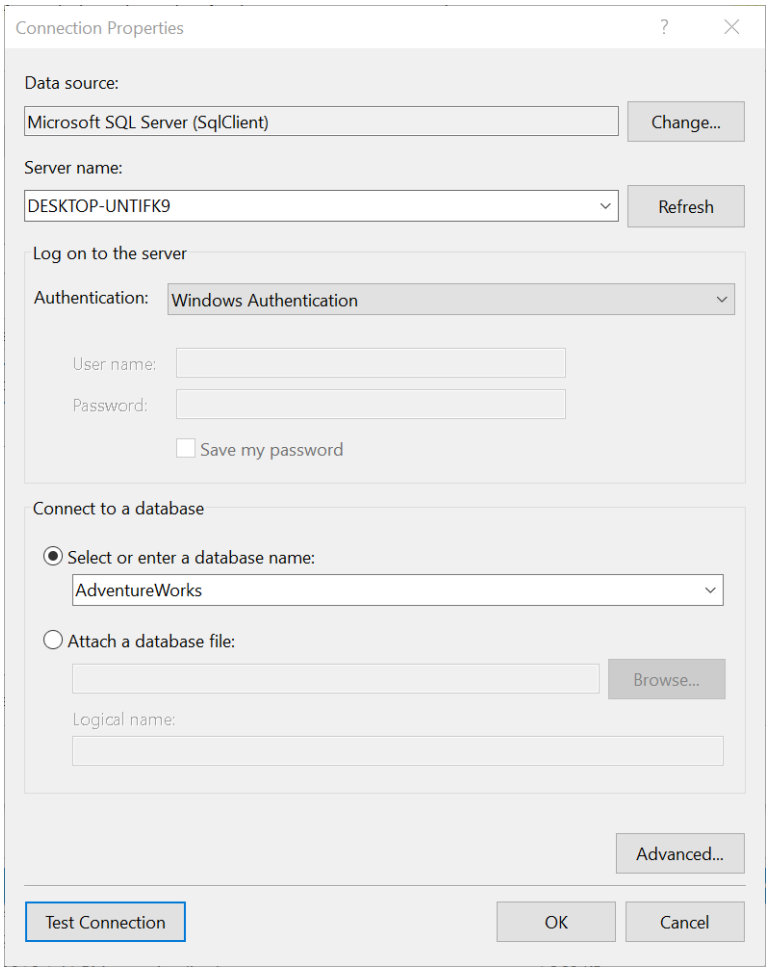
1. Launch Visual Studio and create a new Project. Select " Report Server Project Wizard". Enter a name for your project and click on the "OK" button.



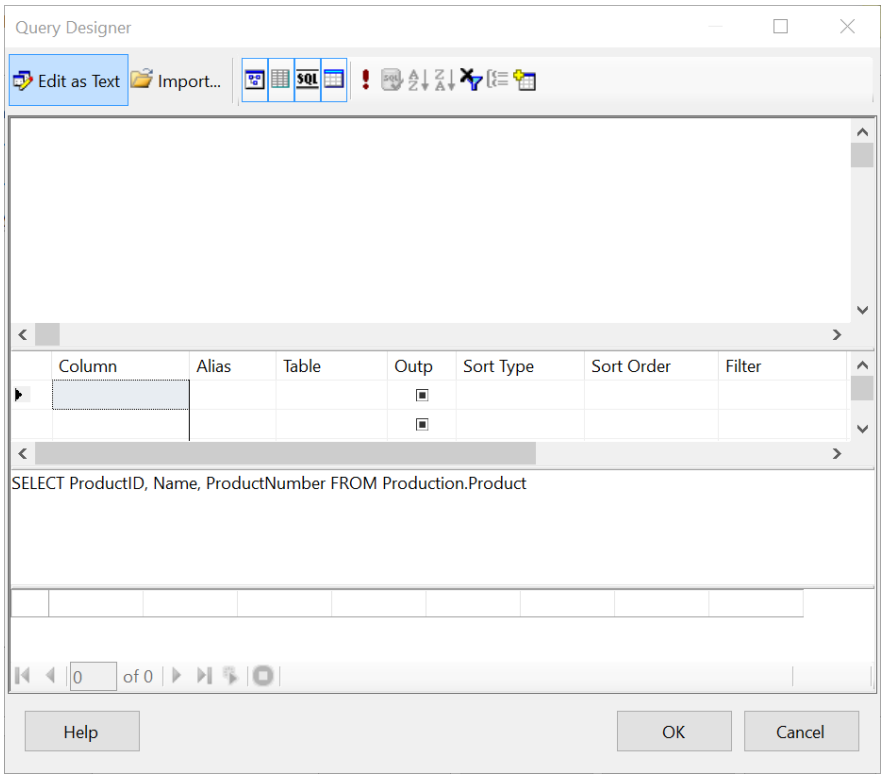
2. A "Welcome to the Report Wizard" will be launched to provide you with an overview of the steps required to create a report. Click on the "Next" button. You should see the "Select the Data Source" dialog as shown below:



3. Click on the "Edit" button to setup the Data source. In the screenshot below our "AdventureWorks" database is stored in the "DESKTOP-UNTIFK9" Server with Windows Authentication. You can choose to setup the connection to your database. After setting up the properties, you can click on the "Test Connection" button to ensure a successful connection before proceeding. Click on the "OK" button when you are ready.



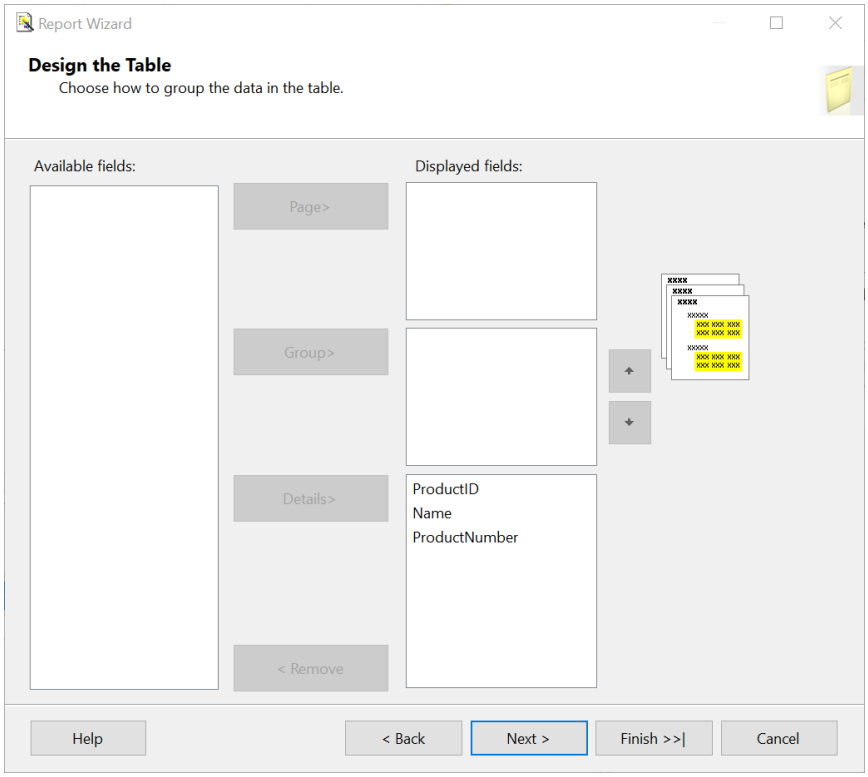
4. In the "Select the Data Source" dialog, click on the "Next" button. In the "Design the Query" dialog, click on the "Query Builder" button. You should see the following dialog as shown in the screenshot below:



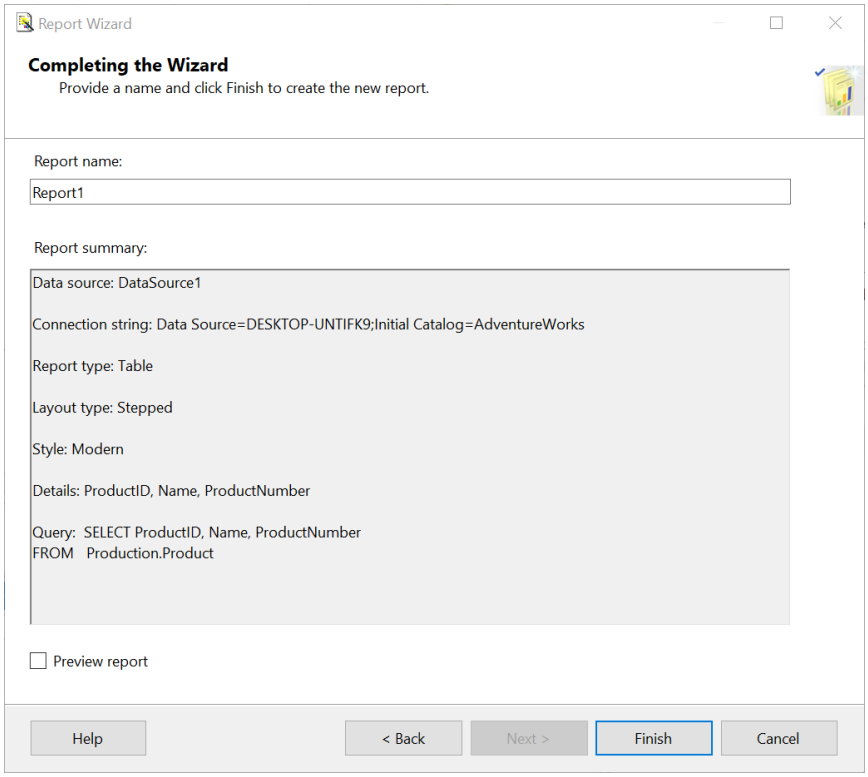
5. We are going to select 3 fields (columns) from the "Production" table. Enter the following query and click on the "OK" button:

```
SELECT ProductID, Name, ProductNumber from Production.Product
```

6. In the "Select the Report Type" dialog, select a "Tabular" report and click on the "Next" button. In the "Design the Table" dialog, select all the available fields and add them to the "Details" section as shown below:

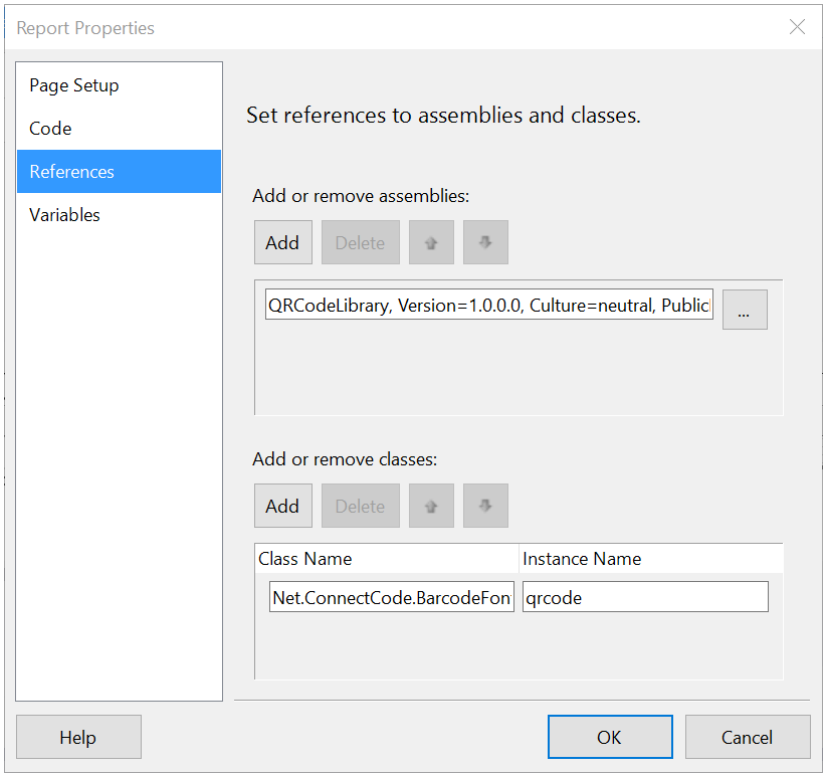


7. Click on the "Next" button followed by the "Finish" button.



8. We have now successfully created Reporting Services (.rdl) report in Visual Studio. If you are using Visual Studio 2019, click on "Report1.rdl" in "Solutions Explorer", and then the "..." button in "Properties -> Assemblies". If you are using Visual Studio 2017, click on the "Report -> Report Properties" menu item. We are going to add a reference to our "QRCodeLibrary.dll". This DLL (Dynamic Link Library) will be used to help you generate a QR Code in the report.

Click on the "References" tab and click on the "Add" button followed by the "..." button to add an assembly. In the "Add Reference" dialog, click on the "Browse" tab and navigate to the "C:\Program Files (x86)\ConnectCodeQRCode\Resource\ReportingServices" folder. Select the "QRCodeLibrary.dll" and click on the "OK" button.

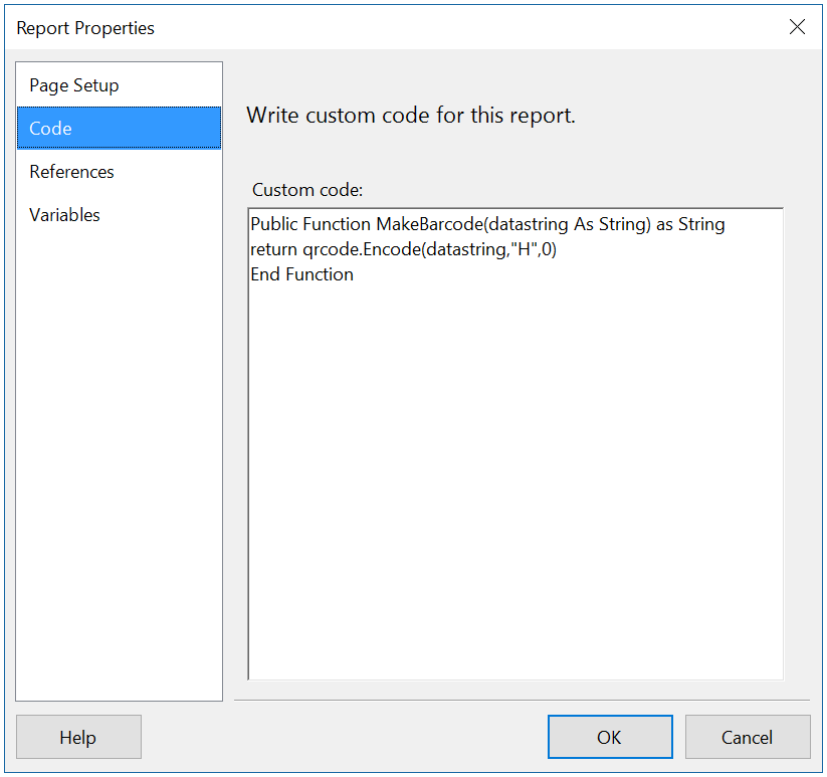



Click on the "Add" button in "Add or remove classes". Enter "Net.ConnectCode.BarcodeFontsStandard2D.QR" as the class name and "qrcode" as the instance name and click on the "OK" button. We have successfully added a reference to the DLL and created an instance object for generating QR Codes.

9. Click on the Code tab and enter the following:

```
Public Function MakeBarcode(datastring As String) as String
return qrcode.Encode(datastring,"H",0)
End Function
```

The programming codes above use the barcode instance object to generate a QR Code barcode with the "datastring" parameter as the input data, "H" as the Error Correction Level, and 0 as the QR Code Mask.

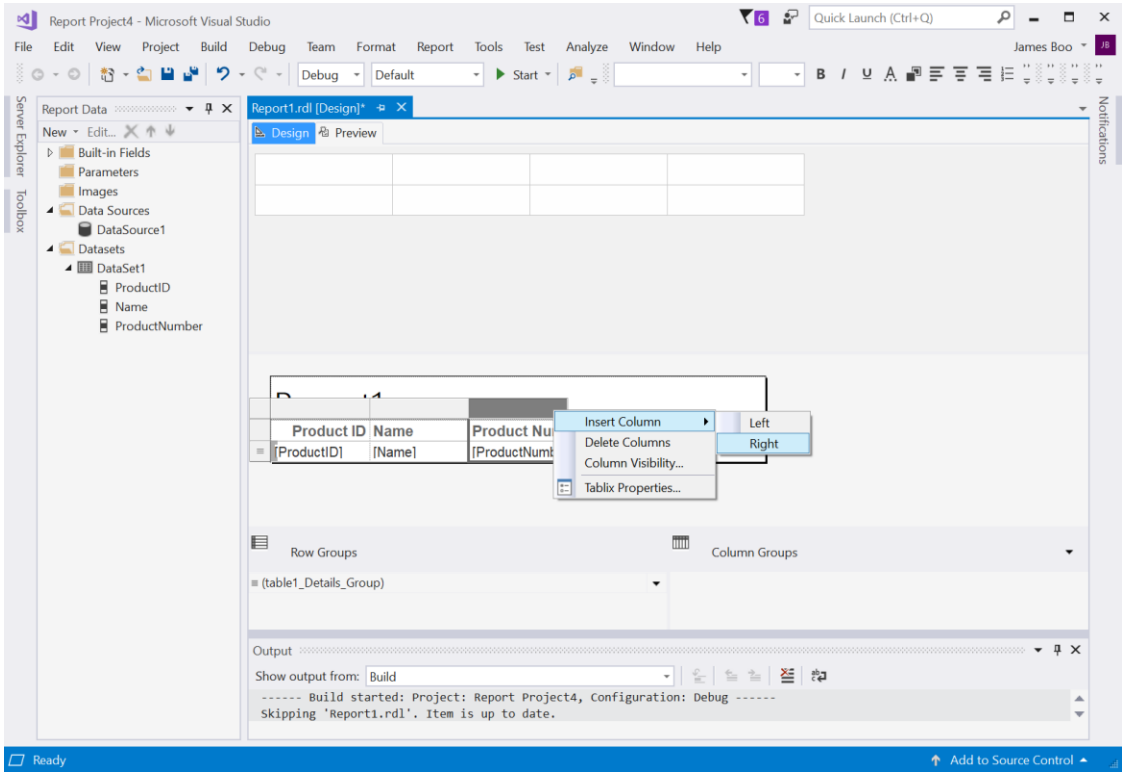



Supported Error Correction Level
L - Allows recovery of up to 7% data loss
M - Allows recovery of up to 15% data loss
Q - Allows recovery of up to 25% data loss
H - Allows recovery of up to 30% data loss

Supported Mask: 8 (0 to 7 or 8 for Auto)

Next, click on the "OK" button to exit from the "Report Properties" dialog.

10. In the "Design" tab of "Report1.rdl", right-click on the last column of the table and select "Insert Column->Right". We are going to add a column for our QR Codes.



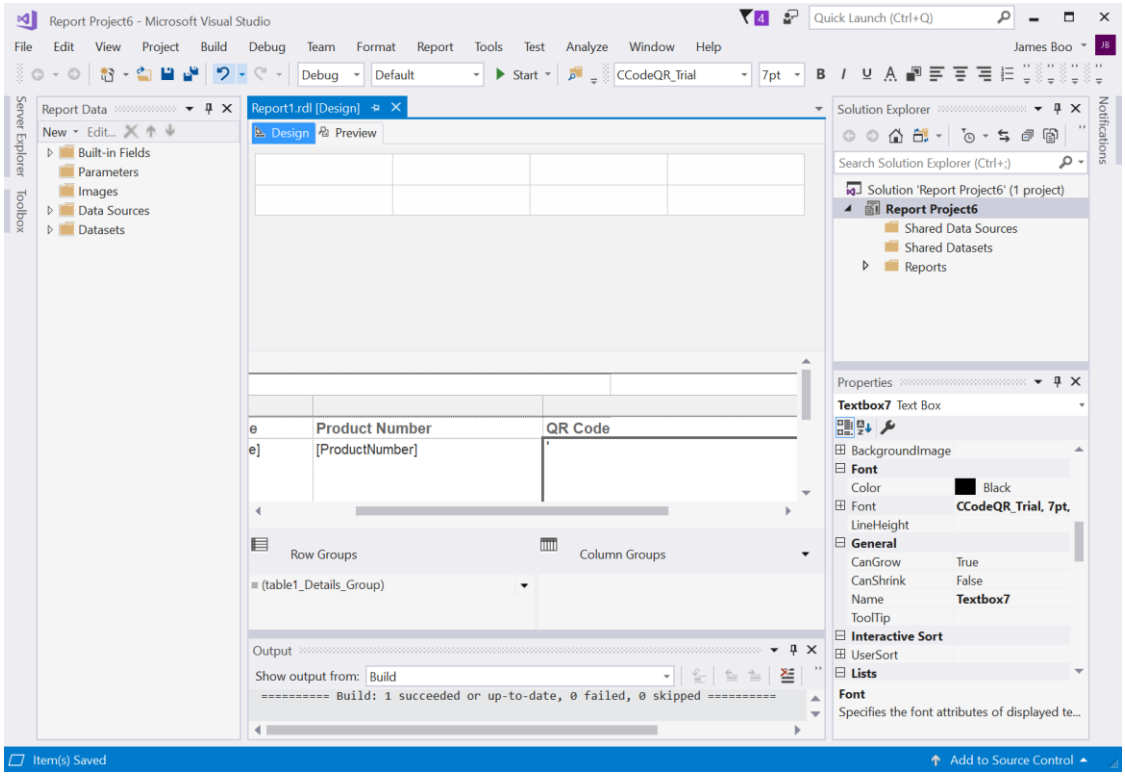

11. In the Toolbox, select a "Text box" object and add it into the column from the previous step. The output characters generated by the DLL will be placed in the "Text box".

12. Right-click on the "Text Box" object and select "Expression". Enter the following expression:

```
=Code.MakeBarcode(Fields!ProductNumber.Value)
```

If you recall, "MakeBarcode" is a function we have defined in "Report Properties". And, in the above, we apply the function on the "ProductNumber" field. The result will be placed in the "Text box" field.

13. In the previous steps, we have defined a "MakeBarcode" function that uses the "QRCodeLibrary.dll" to generate a QR Code. To be more exact, the "MakeBarcode" function returns a stream of output characters. These output characters when applied with QR Code barcode font gives you an industrial quality QR Code barcode.




Next, we are going to apply a barcode font to the "Text box" that contains our output characters. Click on the "Text box" object and expand the "Font" properties as shown below. Set the "Font" to "CCodeQR" (or "CCodeQR_Trial") and "FontSize" to "7". You can reduce or increase the "FontSize" later to meet the size requirements of your QR Code.

14. Save all the files and click on the "Preview" tab. You should see the following report with QR Code barcodes generated using the "ProductNumber" field.