

# AtalasoftSupportUtils.QRCodeBarcodeWriter

## Version

This is for v11.5.0.10.0.364 Also known as 11.5.0.10 of DotImage

## Description

This is an Atalasoft-compatible QR Code Barcode Writer class (AtalasupportUtils.QRCodeBarcodeWriter)

It was designed to implement familiar functionality to our DatamatrixBarcodeWriter class, but adds some useful convenience methods to make the creation of QR Codes with Atalasoft even easier

This is very much an "Atalasoft-compatible" wrapper around Shane32/QRCoder. Atalasoft Support created this as a means to provide a missing option for barcode writing due to the immense popularity of QR Codes

## Installation / Deployment

You will need to extract the AtalasoftSupportUtils.QRCodeBarcodeWriter.zip file to a known location

If you have admin rights it can go into

```
C:\Program Files (x86)\AtalasoftSupportUtils.QRCodeBarcodeWriter
```

Alternately,

```
C:\ProgramData\AtalasoftSupportUtils.QRCodeBarcodeWriter
```

works nicely.

## Target Framework and bitness

### Bitness

Our SDK "has Bitness" so your solution must target x86 or x64

DLLs for both are supplied

### Supported .NET

Currently, this project support .NET Framework 4.6.2-4.8 and .NET (Core) 8.0 + (windows only)

#### NOTE

In order to target .NET 8.0+ you must ensure you specify Windows and set UseWindowsForms true in your csproj/vbproj file

```
<TargetFramework>net8.0-windows</TargetFramework>  
<UseWindowsForms>true</UseWindowsForms>
```

## Referencing QRCodeBarcodeWriter

You will need to Add Reference to the framework and bitness of your solution from

### .NET Framework 4.6.2-4.8.1

- %AtalasoftSupportUtils.QRCodeBarcodeWriter%/bin/4.6.2/x64
- %AtalasoftSupportUtils.QRCodeBarcodeWriter%/bin/4.6.2/x86

## .NET 8+

- %AtalasoftSupportUtils.QRCodeBarcodeWriter%/bin/8.0/x64
- %AtalasoftSupportUtils.QRCodeBarcodeWriter%/bin/8.0/x86

Include AtalasupportUtils.QRCodeBarcodeWriter.dll in your solution by referencing the appropriate DLL

## Dependencies

You need to reference (and license)

- Atalasoft.dotImage.dll
- Atalasoft.dotImage.Lib.dll
- Atalasoft.dotImage.Barcoding.Writing.dll
- Atalasoft.Shared.dll

These can be found under

C:\Program Files (x86)\Atalasoft\DotImage\11.5\bin\

(subfolders for 4.6.2 / 6.0 and sub sub folders for x86 and x64)

### NOTICE

This was built against 11.5.0.10.364

## Embedding Licensing

In order to embed your licensing, you'll need these two lines in your licenses.licx file

```
Atalasoft.Imaging.AtalaImage, Atalasoft.dotImage
Atalasoft.Barcoding.Writing.BarcodeWriter, Atalasoft.dotImage.Barcoding.Writing
```

Please see [HOWTO: License an EXE for Deployment \(.NET Framework\)](#)

## Usage

Once you've got your dependencies referenced, you should now be able to create a QRCodeBarcodeWriter and use its features

### Simple Barcode Image with text

```
using AtalaSupportUtils;

//... this is presumed to be inside a suitable class/method call

QRCodeBarcodeWriter writer = new QRCodeBarcodeWriter();

using (AtalaImage sampleCode = writer.Generate("This is my First Barcode, Hello world!"))
{
```

```
sampleCode.Save("sample_barcode.png", new PngEncoder(), null);  
}
```

## Barcode Image with Embedded Icon

```
using AtalaSupportUtils;  
  
//... this is presumed to be inside a suitable class/method call  
  
QRCodeBarcodeWriter writer = new QRCodeBarcodeWriter();  
  
using (AtalaImage sampleCode = writer.Generate("This is my First Barcode, Hello world!",  
BitmapContainingIconHere ))  
{  
    sampleCode.Save("sample_barcode.png", new PngEncoder(), null);  
}
```

## QRCodeBarcodeWriter API Reference

### Constructor

```
public QRCodeBarcodeWriter()
```

The default constructor for QRCodeBarcodeWriter.

Currently, we don't offer any overloads, you can set properties after construction

### Properties

#### ErrorCorrectionLevel

```
public ECCLevel ErrorCorrectionLevel { get; set; }
```

Defines the levels of error correction available in QR codes.

Each level specifies the proportion of data that can be recovered if the QR code is partially obscured or damaged.

#### ModuleSize

```
public int ModuleSize { get; set; }
```

The pixel size each b/w module is drawn

#### BackColor

```
public Color BackColor { get; set; }
```

The color of the dark/black modules (Defaults to Color.Black)

#### ForeColor

```
public Color ForeColor { get; set; }
```

The color of the light/white modules (Defaults to Color.White)

## Bounds

```
public Rectangle Bounds { get; }
```

Gets the bounds of the last render call. For Generate, returns a rectangle with the size of the size of the rendered image.

## Methods

### Dispose()

```
public void Dispose()
```

Implements

**IDisposable.Dispose()**

### Generate(string)

```
public AtalaImage Generate(string text)
```

Simple Generate call will return an AtalaImage that contains the provided text rendered via default settings or using whatever settings have been applied to the exposed properties of QRCodeBarcodeWriter

### Generate(string, Bitmap[, int [, int [, Color]]])

```
public AtalaImage Generate(  
    string text,  
    Bitmap icon,  
    int iconBorder = 6,  
    int iconPercent = 15,  
    Color? iconBackgroundColor = null  
)
```

Overload of Generate to allow QR Codes with an icon in the center..

The provided icon can be a Bitmap of any size - it will be scaled to fit the percent/border specified

#### NOTE

Pay attention to your iconPercent and iconBorder widths:

If you take up 25% of your barcode, you really want to be sure you've set the Error Correction Level (ECCLevel) high enough or else you risk crating an invalid QR Code - We will not throw an error but if the total size of icon and border is Greater than the allowable ECClevel, the QR Code will not read.

Suggested that you do not set this value higher than 24%

## Parameters

### text

Type: **\*\*System.String\*\***  
Text to Encode

### icon

Type: **\*\*System.Drawing.Bitmap\*\***  
**\*\*Bitmap\*\*** that will be overlaid over the center of the barcode

### iconBorder [optional]

Type: **\*\*System.Int32\*\***  
**\*\*Int\*\*** for border width (valid values 1-99)  
Default: 6

### iconPercent [optional]

Type: **\*\*System.Int32\*\***  
**\*\*Int\*\*** for percentage of the barcode to cover (valid values 1-99)  
Default: 15  
NOTE: if you cover more percent than your ECCLevel provides you can end up with a non-valid QRCode - we do NOT validate

### iconBackgroundColor [optional]

Type: **\*\*System.Drawing.Color\*\***  
**\*\*Color\*\*** to use for the background of the icon  
If passed null or not passed, the barcodes background color will be used instead (default)

## Render(byte[], Graphics, Rectangle)

```
public void Render(  
    byte[] buffer,  
    Graphics g,  
    Rectangle rect  
)
```

Use this Render overload if you have NON-ASCII or other content for your Barcode.

### buffer

Type: **\*\*System.Byte[]\*\***  
**\*\*Byte\*\*** containing content to Encode

### text

Type: **\*\*System.Drawing.Graphics\*\***  
**\*\*Graphics\*\*** object where the bar code will be rendered.

## rect

Type: **\*\*System.Drawing.Rectangle\*\***  
Layout rectangle for the bar code

## Render(string, Graphics, Rectangle)

```
public void Render(  
    string text,  
    Graphics g,  
    Rectangle rect  
)
```

Use this Render mode if you want to encode standard ASCII text into your Barcode

## text

Type: **\*\*System.String\*\***  
Text to Encode

## text

Type: **\*\*System.Drawing.Graphics\*\***  
**\*\*Graphics\*\*** object where the bar code will be rendered.

## rect

Type: **\*\*System.Drawing.Rectangle\*\***  
Layout rectangle for the bar code

## Events

- NONE

## ECCLevel API Reference

### Syntax

```
public enum ECCLevel
```

### Members

Member name	int Value	Description
Default	-1	Default error correction level, which will select Level M (Medium) unless otherwise specified by the payload. Level M allows approximately 15% of data to be recovered, offering a balance between data capacity and error recovery.
L	0	Level L: Low error correction (approximately 7% of data can be recovered). This level allows the highest data density.
M	1	Level M: Medium error correction (approximately 15% of data can be recovered). Offers a balance between data capacity and error recovery.
Q	2	Level Q: Quartile error correction (approximately 25% of data can be recovered). More robust error correction at the cost of reduced data capacity.
H	3	Level H: High error correction (approximately 30% of data can be recovered). Provides the highest level of error recovery, ideal for environments with high risk of data loss.

## Required Licensing Disclosures

This project uses code from <https://github.com/Shane32/QRCoder>  
It is used under the MIT license

```
The MIT License (MIT)

Copyright (c) 2013-2025 Raffael Herrmann
Copyright (c) 2024-2025 Shane Krueger

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
```

IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN  
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.