TBarCode .NET

Barcode Generator Software for .NET

Version 1.1.0

Barcodes in Microsoft SQL Server Reporting Services (SSRS)

10 November 2015

TEC-IT Datenverarbeitung GmbH
Hans-Wagner-Str. 6
A-4400 Steyr, Austria

t ++43 (0)7252 72720
f ++43 (0)7252 72720 77
office@tec-it.com
www.tec-it.com

1 Content

2 Disclaimer

The actual version of this product (document) is available as is. TEC-IT declines all warranties which go beyond applicable rights. The licensee (or reader) bears all risks that might take place during the use of the system (the documentation). TEC-IT and its contractual partners cannot be penalized for direct and indirect damages or losses (this includes non-restrictive, damages through loss of revenues, constriction in the exercise of business, loss of business information or any kind of commercial loss), which is caused by use or inability to use the product (documentation), although the possibility of such damage was pointed out by TEC-IT.

TEC-IT Datenverarbeitung GmbH
Wagnerstr. 6

A-4400 Austria
t.: +43 (0)7252 72720
f.: +43 (0)7252 72720 77
http://www.tec-it.com

3 Haftungsausschluss

Dieses Produkt (bzw. Dokument) steht Ihnen in der aktuellen Version „WIE BESEHEN – ohne Gewährleistung“ zur Verfügung. TEC-IT weist alle Garantien, die über das anwendbare Recht hinausgehen, zurück. Risiken, die aus der Benutzung des Produkts und der Dokumentation entstehen, trägt der Lizenznehmer bzw. Benutzer. TEC-IT und seine Vertragspartner dürfen nicht für direkte oder indirekte Schäden oder Verluste belangt werden (dies beinhaltet, uneingeschränkt, Schäden durch den Verlust von Einkünften, Einschränkungen in der Geschäftsausübung, Verlust von Geschäftsinformationen sowie andere wirtschaftliche Verluste), die aus der Benutzung oder Unfähigkeit zur Benutzung des Produkts (der Dokumentation) entstanden sind, selbst wenn TEC-IT auf die Möglichkeit solcher Schäden hingewiesen hat.

TEC-IT Datenverarbeitung GmbH
Wagnerstr. 6

A-4400 Austria
t.: +43 (0)7252 72720
f.: +43 (0)7252 72720 77
http://www.tec-it.com

4 Before Starting

4.1 MS SQL Server Reporting Services and TBarCode .NET

In the following samples we will create an SSRS report with barcode images using *TBarCode .NET* and Visual Studio .NET.

There is a method with direct integration (bar code generation “on the fly”) and a method with indirect integration (bar code images stored in the database before running the report).

4.1.1 Prerequisites

Basically you need to install *TBarCode SDK* on your system. You can download a free evaluation version of TBarCode directly from our web site:

http://www.tec-it.com/download/tbarcode/windows/Download.aspx

From the SDK we will use the *TBarCode .NET* component. The MSI setup will install the *TBarCode .NET* assemblies into the GAC[1].

Additional you need the following applications installed:

- Microsoft SQL Server – Reporting Services
- Microsoft .NET Framework 3.5 or higher
- Microsoft Visual Studio .NET 2008 or higher

[1] For alternative assembly locations see here: https://msdn.microsoft.com/en-us/library/ms155034(v=sql.110).aspx

5 Direct Integration

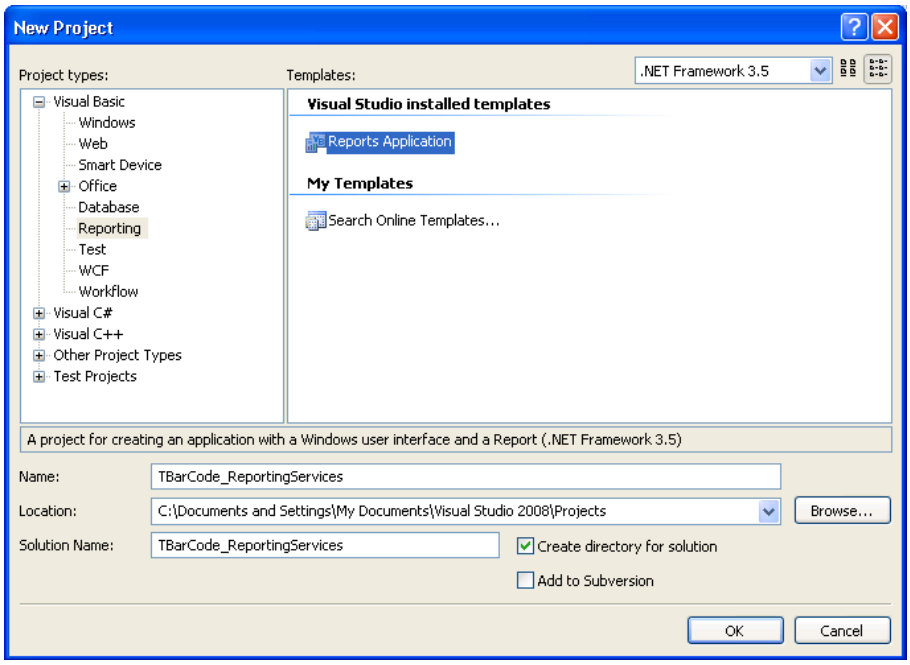
The following method integrates bar code generation directly into the report. Each time the report is run, the bar codes are generated “on the fly” and inserted into the image box on the report.

The advantage of this solution is that you don’t have to change anything in the database and the bar codes are always “up to date”. Disadvantage of this method is that you need to deal with security and trusted codes modules.

For this solution we provide sample code and a sample Visual Studio project, which uses a local report.

5.1 Creating the Reporting Project

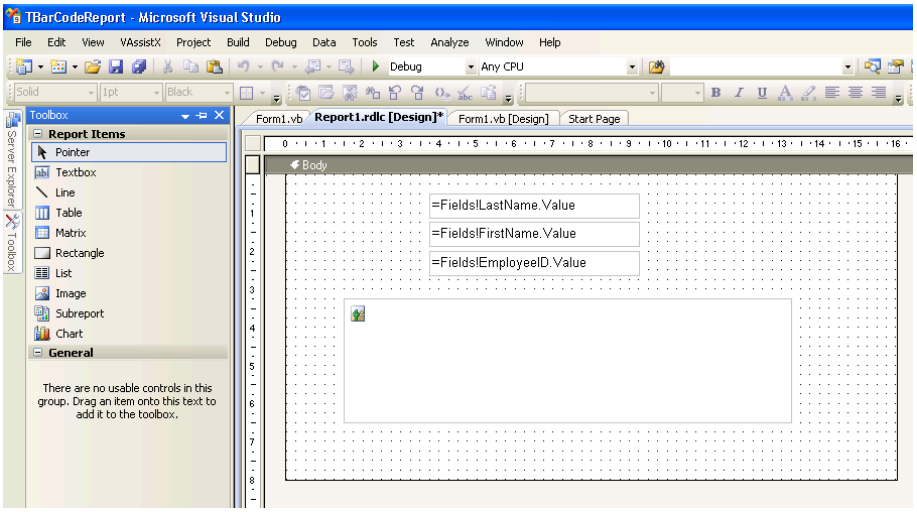
- Open Visual Studio .NET 2008 and create a new report project



- In the project wizard choose your database connection (in the sample the Northwind database is used).

5.2 Report Design

- Open the automatically generated “Report1.rdlc” and design your report as you want.
- Insert an image object into your report at the position where you want to create the barcode in.

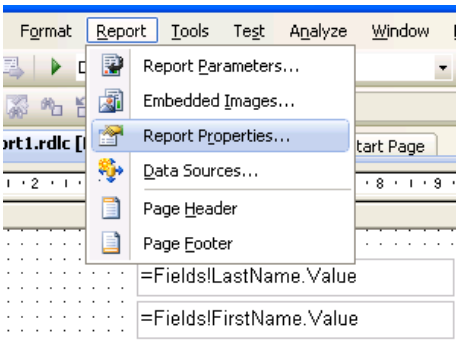


5.3 Creating the Barcode

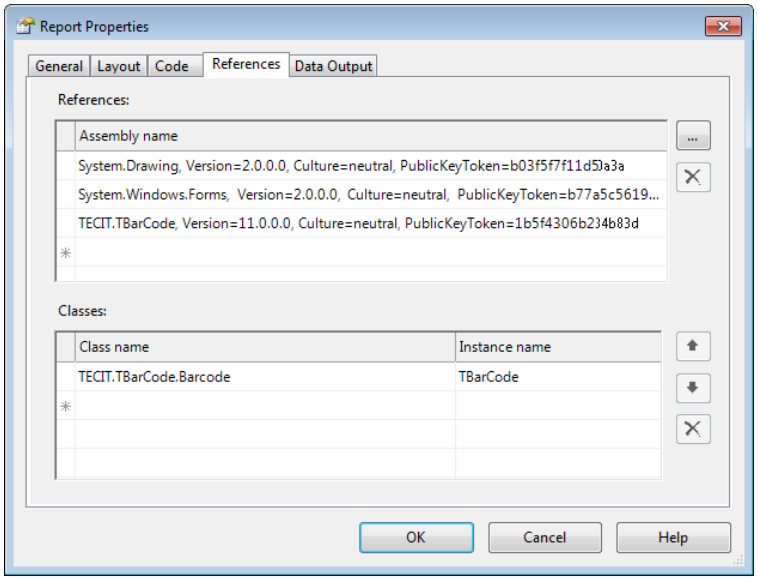
In this step we’ll create the barcode. This sample creates a Code 128 barcode, but with TBarCode you’re able to create over 100 different barcode types. A list of all supported barcode symbologies is available here: Barcode symbologies supported by TBarCode

5.3.1 Report Properties

- Open the Report1.rdlc in Visual Studio .NET 2008. In the menu select *Report* → *Report Properties…*



- Open the *References* tab and add the following references:
  - TECIT.TBarCode
  - System.Drawing
  - System.Windows.Forms



- Now you have to create an instance of TBarCode. Therefore add the following *Class name*:

TEC-IT.TBarCode.Barcode

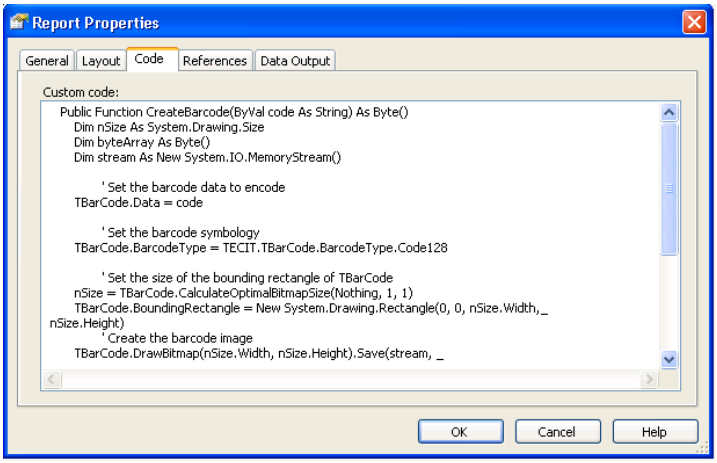
For the *Instance name* you use:

TBarCode

5.3.2 Barcode Generator Code (VB)

- In the next step you have to open the *Code* tab. Enter the function as specified below. The basic principle is to generate a barcode stream which is than shown in the image object.

```
Public Function CreateBarcode(ByVal code As String) As Byte()
    Dim nSize As System.Drawing.Size
    Dim byteArray As Byte()
    Dim stream As New System.IO.MemoryStream()

    ' Set the barcode data to encode
    TBarCode.Data = code

    ' Set the barcode symbology
    TBarCode.BarcodeType = TECIT.TBarCode.BarcodeType.Code128

    ' Set initial default size
    TBarCode.BoundingRectangle = New System.Drawing.Rectangle (0, 0, 200, 150)

    ' calculate pixel accurate width (for screen resolution)
    nSize = TBarCode.CalculateOptimalBitmapSize(Nothing, 1, 1)
    TBarCode.BoundingRectangle = New System.Drawing.Rectangle(0, 0,_
                        nSize.Width, nSize.Height)

    ' Create the barcode image (BMP) and write to stream
    TBarCode.DrawBitmap(nSize.Width, nSize.Height).Save(stream,
                         System.Drawing.Imaging.ImageFormat.Bmp)

    ' Set the stream position to the beginning of the stream.
    stream.Seek(0, System.IO.SeekOrigin.Begin)

    ' Read all Bytes from the stream.
    byteArray = New Byte(CType(stream.Length, Integer)) {}
    stream.Read(byteArray, 0, stream.Length)

    Return byteArray
End Function
```



5.4 Security ➔ Trusted Code Modules

- To use the specified assemblies you have to trust them[2]. Therefore open the *Form* which was created with the project. Open this form in the *Code View* and add a *Load* function with the following lines:

```
Me.ReportViewer1.LocalReport.ExecuteReportInCurrentAppDomain(AppDomain.CurrentDomain.Evid
ence)
Me.ReportViewer1.LocalReport.AddTrustedCodeModuleInCurrentAppDomain("TECIT.TBarCode,
Version=11.0.0.0, Culture=neutral, PublicKeyToken=1b5f4306b234b83d")
Me.ReportViewer1.LocalReport.AddTrustedCodeModuleInCurrentAppDomain("System.Drawing,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a")
Me.ReportViewer1.LocalReport.AddTrustedCodeModuleInCurrentAppDomain("System.Windows.Forms
, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089")
```

  This moves the report into the actual app domain which treats our code modules as trusted.

- Finally you have to change the settings of the image object in the report. Set the *Source* property to ***Database*** and the *Value* property to call the *CreateBarcode*.

  For example:
  `=Code.CreateBarcode(Fields!LastName.Value.ToString())`

[2] More information:
http://blogs.msdn.com/b/mosharaf/archive/2005/12/20/localreportcustomcode.aspx
https://msdn.microsoft.com/en-us/library/ms155034(v=sql.110).aspx

6 Indirect Integration

The following method generates bar codes as image data and stores them in the database. The report reads the image data from the database and displays the bar codes during runtime.

The advantage of this solution is that you don't have to deal with security during report runtime (no additional code modules required in the report). The disadvantage is that you have to adapt your database table(s) and update/insert the bar code images whenever the encoded data changes.

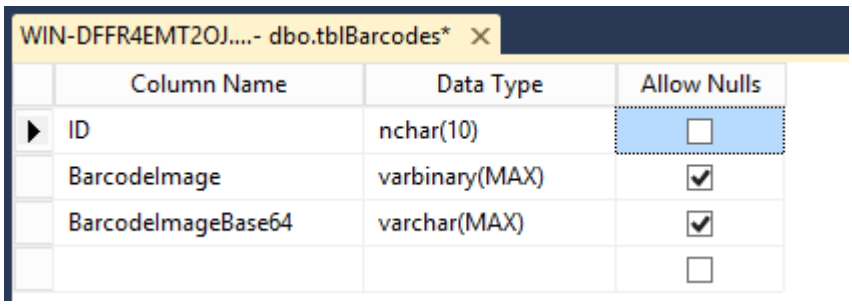
► Indirect integration is the recommended method for Microsoft Dynamics NAV® (2013 and later).

The following tutorial / sample code shows how to

- Add a binary field in your table
- Create a bar code image (GIF) and stores it into the binary field in the table
- Display the bar code image from the binary field in your SSRS report

6.1 Table

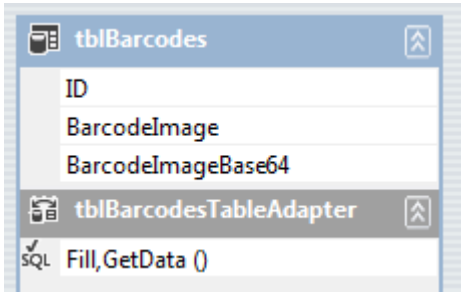
For this tutorial we have created a table, which has a *varbinary* field for storing the bar code image in binary format and also a *varchar* field for storing the barcode image in base64 encoded form.



You can use one of the two field variants depending on your database. In our sample we use both field variants for demonstration.

6.2 Data Set

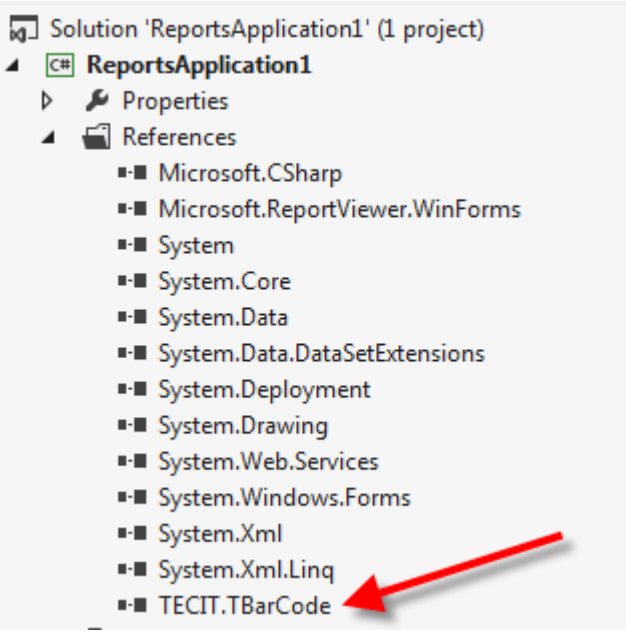
In our Visual Studio project, we have created a data set as follows:



This will help us to update and insert the bar code images. You can use also other methods (e.g. LINQ) or whatever you want.

6.3 References

Prerequisite for bar code generation: Add a reference to the assembly *TECIT.TBarCode.dll* to your project (see TBarCode .NET Developer manual, section 7 for details):



6.4 Barcode Generator Code (C# .NET)

We need this barcode creation function – add this to a suitable place into your project:

6.4.1 Linear Code (Code-128)

```
public Byte[] GenerateBarcodeStream (String data)
{
  TECIT.TBarCode.Barcode barcode = new TECIT.TBarCode.Barcode();
  barcode.BarcodeType = TECIT.TBarCode.BarcodeType.Code128;
  barcode.Data = data;
  barcode.Font = new System.Drawing.Font("Arial", 10f);
  barcode.Dpi = 100;
  Size optimalSize = barcode.CalculateOptimalBitmapSize(null, 1, 1);
  barcode.BoundingRectangle = new Rectangle(0, 0, optimalSize.Width, 60 /* 60px */);

  using (System.IO.MemoryStream ms = new System.IO.MemoryStream())
  {
    Bitmap bitmap = barcode.DrawBitmap();
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
    ms.Position = 0;
    return ms.ToArray();
  }
}
```

6.4.2 2D Code (Data Matrix)

```
public Byte[] GenerateBarcodeStream2D (String data)
{
  TECIT.TBarCode.Barcode barcode = new TECIT.TBarCode.Barcode();
  barcode.BarcodeType = TECIT.TBarCode.BarcodeType.DataMatrix;
  barcode.Data = data;
  barcode.Dpi = 100;
  Size optimalSize = barcode.CalculateOptimalBitmapSize(null, 2, 2);
  barcode.BoundingRectangle = new Rectangle(0, 0, optimalSize.Width,
    optimalSize.Height);

  using (System.IO.MemoryStream ms = new System.IO.MemoryStream())
  {
    Bitmap bitmap = barcode.DrawBitmap();
    bitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
    /*
    // or save as 24 bit JPG
    ImageCodecInfo jpgCodec = ImageCodecInfo.GetImageEncoders().Where(codec =>
                  codec.FormatID.Equals(ImageFormat.Jpeg.Guid)).FirstOrDefault();
    System.Drawing.Imaging.EncoderParameters parameters = new
                  System.Drawing.Imaging.EncoderParameters();
    parameters.Param[0] = new EncoderParameter
                  (System.Drawing.Imaging.Encoder.ColorDepth, 24);
    bitmap.Save(ms, jpgCodec, parameters);
    */
    ms.Position = 0;
    return ms.ToArray();
  }
}
```

6.5 Insert Barcode Image for One Record

If we want to insert one record with a bar code image, we could to it like that:

```
Byte[] barcodeImageStream = GenerateBarcodeImage ("12345");

DataRow myRow = TestDataSet.tblBarcodes.NewRow();
myRow [0] = "12345";
myRow [1] = barcodeImageStream;                          // add as binary data stream
myRow [2] = Convert.ToBase64String(barcodeImageStream); // add as base64 string
TestDataSet.tblBarcodes.Rows.Add(anyRow);

tblBarcodesTableAdapter.Update(TestDataSet);
TestDataSet.tblBarcodes.AcceptChanges();
```

▶ As you can see, the sample shows both variants, creating the bar code image as binary data (Byte array) and creating it as base64 encoded string. Choose the method which apply best to your database and table layout.

After creating the barcode image and writing the bitmap to the table it looks as follows:

Results | Messages

	ID	BarcodeImage	BarcodeImageBase64
1	1	0x4749463839614F003C00F700000000000000003300006600...	R0lGODlhTwA8APcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArM...
2	1	0x4749463839614F003C00F700000000000000003300006600...	R0lGODlhTwA8APcAAAAAAAAAMwAAZgAAmQAAzAAA/wArAAArM...

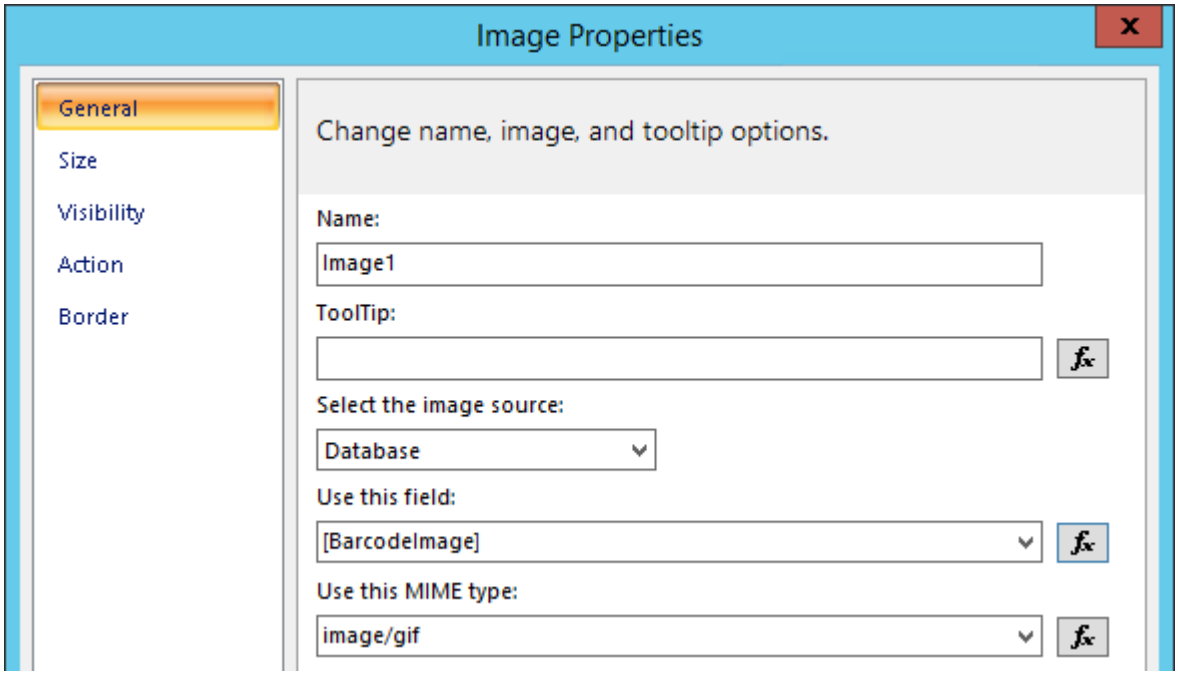
6.6 Update Barcode Images for All Records

If you want to iterate through a table and convert a specific field (e.g. an article number) into a bar code image field, you could do it that way:

```
tblBarcodesTableAdapter.Fill(TestDataSet.tblBarcodes);

foreach (DataRow row in TestDataSet.tblBarcodes.Rows)
{
  if (row["ID"] != null)
  {
    string myData = row["ID"].ToString().Trim();
    try
    {
      Byte[] barcodeImageStream = GenerateBarcodeImage(myData);
      row["BarcodeImage"] = barcodeImageStream;
      row["BarcodeImageBase64"] = Convert.ToBase64String(barcodeImageStream);
    }
    catch (Exception ex)
    {
      // error in bar code generation (invalid data?)
    }
  }
}

tblBarcodesTableAdapter.Update(TestDataSet);
TestDataSet.tblBarcodes.AcceptChanges();
```

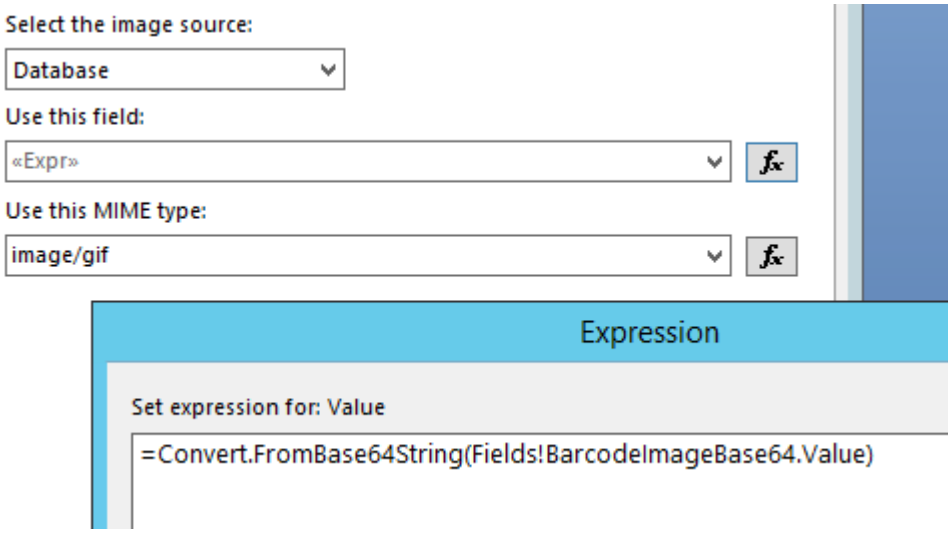
6.7 Display Barcode Image in Report

In order to display the bar code image in your report you have to insert an *Image* report item from the toolbox and adjust the properties as follows:



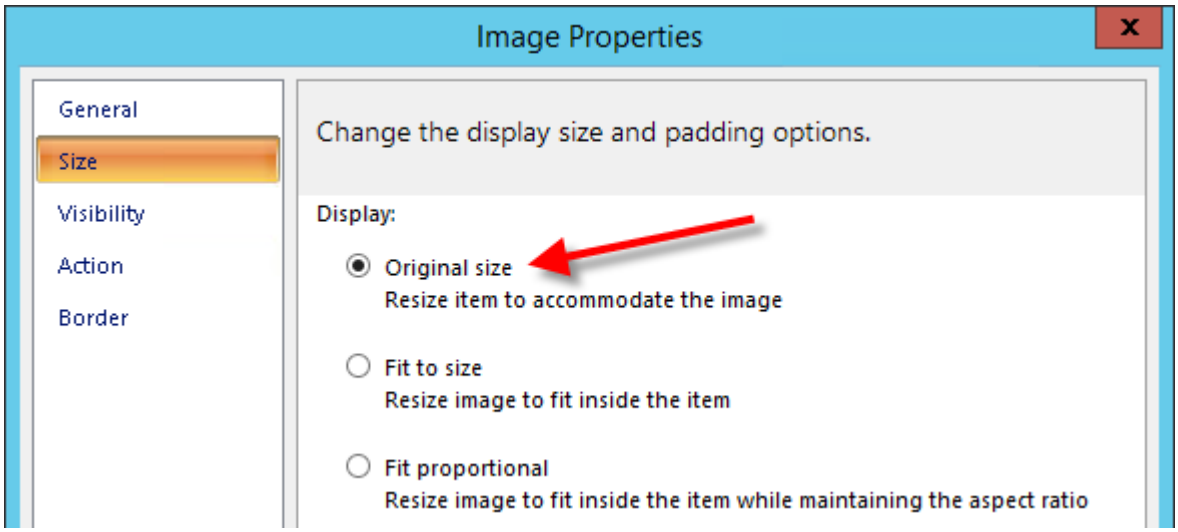
In case you want to store (or have stored) the image in base64 format you need to convert it back to binary format by using an expression:

```
=Convert.FromBase64String(Fields!BarcodeImageBase64.Value)
```



6.7.1 Image Size

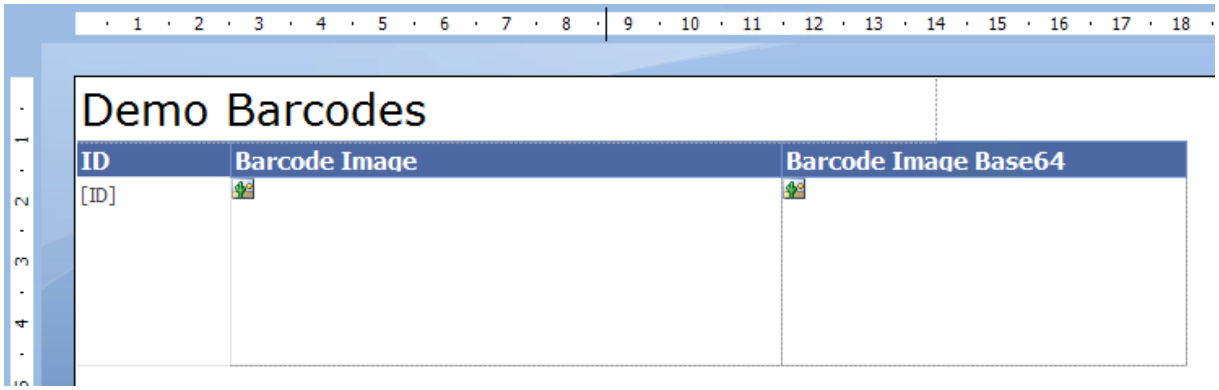
If you take a look into our bar code generation routine you will see the command *CalculateOptimalBitmapSize*. That means the size is already optimized for best quality and therefore we should keep the original size of the image:



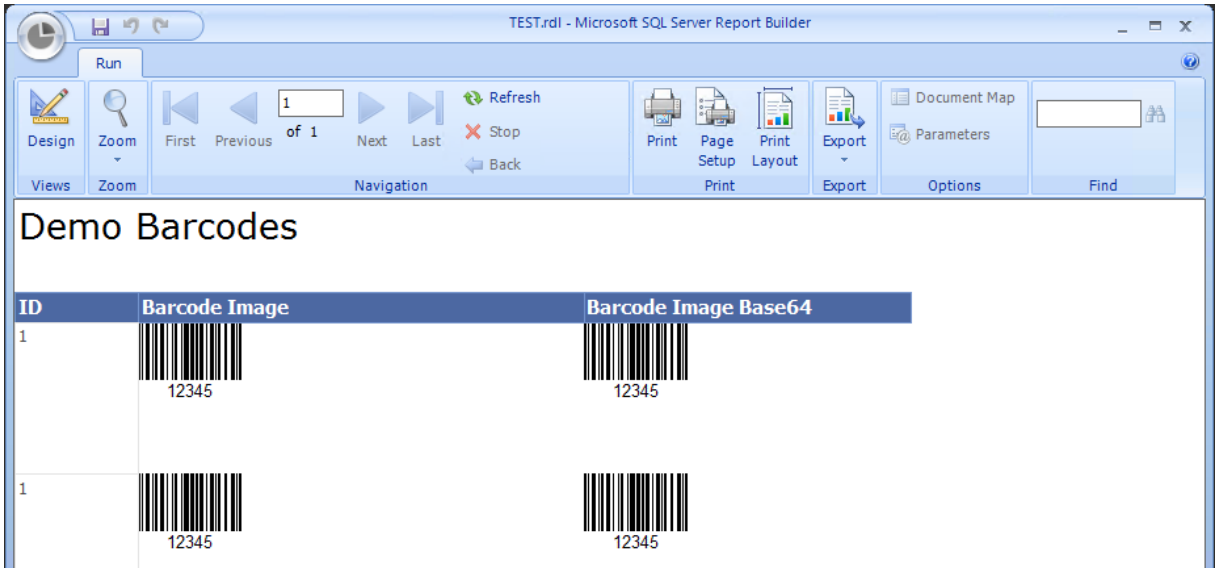
- If you want to change the size, modify the parameters in the *CalculateOptimalBitmap* function.
- For linear bar codes adjust the height in Pixels in the *BoundingRectangle*.
- If this does not lead to the desired result you can change the image properties to *Fit proportional* (but only with high printing quality/output resolution and test scans).

6.8 Report Output

With the following (simple) report design



you will get the following output when running the report:



7 Contact and Support Information

TEC-IT Datenverarbeitung GmbH

Address: Hans-Wagnerstr. 6
AT-4400 Steyr
Austria/Europe
Phone: +43 / (0)7252 / 72 72 0
Fax: +43 / (0)7252 / 72 72 0 – 77
Email: office@tec-it.com
Web: http://www.tec-it.com