

Softek Software Ltd

Softek Barcode Reader Toolkit for Windows

Product Documentation (including the PDF
Extension)



V7.5.1

1 Contents

2	Overview	1
3	Installation	2
3.1	Installation of the toolkit for Development	2
3.1.1	Completing the Installation.....	2
3.1.2	Demonstration Application.....	3
3.1.3	Evaluation License Keys	3
3.1.4	Sample Projects.....	3
3.2	Redistribution and Installation on Production Systems	3
3.2.1	Using a Separate Folder for the DLL files	4
3.2.2	Installation in Windows System Folders	5
3.2.3	Installation in the Same Folder as Application.....	5
3.3	Upgrading.....	5
3.3.1	License Key.....	5
3.3.2	DLL Dependencies.....	5
4	Evaluating the SDK	6
4.1	Using the SoftekSDKDemo application	6
4.2	Integration into applications.....	7
5	Licensing.....	7
5.1	License Keys	7
5.2	Desktop Developer License.....	7
5.3	Desktop Run-Time License.....	7
5.4	Server License	8
5.5	Desktop Distribution License	8
5.6	Desktop and Server Distribution License	8
5.7	Site License.....	8
5.8	Examples	8
6	Contact Information.....	9
6.1	Sales	9
6.2	Support.....	9
7	Interfaces to the Toolkit.....	10
7.1	Windows DLL.....	10
7.1.1	Handling String Values	10
7.1.2	Distribution	11

7.2	OCX/ActiveX.....	11
7.2.1	Distribution	11
7.3	COM Object.....	12
7.3.1	Distribution	12
7.4	.Net 2+ Components	13
7.4.1	Distribution of SoftekBarcodeLib2.dll.....	14
7.4.2	Distribution of SoftekBarcodeLib3.dll.....	14
7.5	.Net 1 Component.....	14
7.5.1	Distribution	15
7.6	Java Class.....	15
7.6.1	Distribution	15
8	Using the .net components on x86 and x64 systems	17
8.1	SoftekBarcodeLib2.dll	17
8.1.1	Approach 1: Build for "Any CPU"	17
8.1.2	Approach 2: Build for "x86" or "x64"	17
8.2	SoftekBarcodeLib3.dll	18
8.2.1	Approach 1: Install SoftekBarcode.dll etc in the Windows system folders	18
8.2.2	Approach 2: Store SoftekBarcode.dll etc in a specific folder.....	18
8.3	Handling PDF Documents in ASP on x64 Systems.....	18
9	Supported Image Formats	20
9.1	TIFF.....	20
9.2	BMP.....	20
9.3	Adobe PDF.....	20
9.4	Other formats	20
10	Supported Barcode Formats	21
10.1	1-D Barcode Formats	21
10.2	2-D and Stacked Barcode Formats.....	21
11	Reading Barcodes from Adobe PDF Documents.....	22
12	Understanding Confidence Levels for Barcode Reading.....	23
13	Reading Barcodes from Color Images.....	24
14	Splitting Documents According to Barcode Position	25
15	Tips on Reading Barcodes	27
15.1	Skewed Barcodes	27
15.2	Badly defined edges to bars.....	27

15.3	Noisy background to the image	27
15.4	White speckles in the black bars.....	28
	Lines and other marks close to the left or right hand end of the barcode.....	28
16	Reading Barcodes from Bitmaps Held in Memory.....	29
17	Using XML Files to store sets of properties	30
18	Unicode File Paths.....	31
18.1	PDF Documents.....	31
19	Appendix A: Methods Reference	32
19.1	List of methods.....	32
19.2	CreateBarcodeInstance.....	33
19.3	DestroyBarcodeInstance.....	34
19.5	ExportXMLSettings.....	35
19.6	GetBarString.....	38
19.7	GetBarStringPos, GetBarStringRect, BarStringPage, BarStringTopLeftX etc	39
19.8	GetBarStringType	41
19.9	GetBarStringDirection.....	42
19.10	GetLastError	43
19.11	GetLastWinError	45
19.12	LoadXMLSettings.....	46
19.13	ProcessXML	49
19.14	SaveResults	51
19.15	ScanBarCode	53
19.16	ScanBarCodeFromBitmap	54
19.17	ScanBarCodeFromDIB	56
19.18	SetScanRect.....	57
20	Appendix B: Properties Reference.....	58
20.1	Setting and Getting Property Values	61
20.2	AllowDuplicateValues	62
20.3	BitmapResolution.....	62
20.4	CodabarMaxVariance.....	63
20.5	Code25Checksum.....	63
20.6	Code39Checksum.....	64
20.7	Code39NeedStartStop	65
20.8	ColorChunks	65

20.9	ColorProcessingLevel	65
20.10	ColorThreshold.....	66
20.11	ConvertUPCEToEAN13	66
20.12	DatabarOptions.....	67
20.13	Despeckle	67
20.14	Encoding.....	68
20.15	ErrorCorrection	68
20.16	ExtendedCode39	68
20.17	FilePathEncoding.....	69
20.18	GammaCorrection.....	69
20.19	LicenseKey.....	69
20.20	LineJump	70
20.21	MaxLength	70
20.22	MedianFilter.....	70
20.23	MinLength	71
20.24	MinOccurrence	71
20.25	MinSeparation.....	71
20.26	MinSpaceBarWidth	71
20.27	MultipleRead.....	72
20.28	NoiseReduction	72
20.29	PageNo	72
20.30	PatchCodeMinOccurrence	72
20.31	Pattern	73
20.32	Pdf417Debug.....	73
20.33	PdfBpp.....	74
20.34	PdfDpi.....	74
20.35	PdfImageExtractOptions	74
20.36	PdfImageOnly.....	75
20.37	PdfImageRasterOptions	75
20.38	Photometric	76
20.39	PrefOccurrence	76
20.40	QuietZoneSize	77
20.41	ReadCodabar	77
20.42	ReadCode128.....	77

20.43	ReadCode25	78
20.44	ReadCode25ni	78
20.45	ReadCode39	78
20.46	ReadCode93	79
20.47	ReadDatabar	79
20.48	ReadDataMatrix	80
20.49	ReadEAN13.....	80
20.50	ReadEAN8.....	80
20.51	ReadMicroPDF417	81
20.52	ReadNumeric.....	81
20.53	ReadPatchCodes	81
20.54	ReadPDF417	81
20.55	ReadQRCode	82
20.56	ReadShortCode128	82
20.57	ReadUPCA	82
20.58	ReadUPCE.....	83
20.59	ScanDirection	83
20.60	ShortCode128MinLength	84
20.61	ShowCodabarStartStop.....	84
20.62	ShowCheckDigit	84
20.63	SkewLineJump.....	84
20.64	SkewTolerance	85
20.65	TifSplitMode.....	85
20.66	TifSplitPath.....	86
20.67	UseOldCode128Algorithm	86
20.68	UseOverSampling.....	87
20.69	UseRunCache	87
20.70	WeightLongerBarcodes.....	87
21	Appendix C: Installation Files	88
21.1	x86 Applications.....	88
21.2	x64 Applications.....	89
21.3	“Any CPU” Applications	91
22	Appendix D: Release Notes	92
22.1	Version 7.5.1.1	92

22.2	Version 7.4.2.1	92
22.3	Version 7.4.2.2	92
22.4	Version 7.4.2.3	92
22.5	Version 7.4.1	92
22.5.1	Version 7.4.1.1	92
22.5.2	Version 7.4.1.2	93
22.5.3	Version 7.4.1.3	94
22.5.4	Version 7.4.1.4	94
22.5.5	Version 7.4.1.5	94
22.5.6	Version 7.4.1.6	94
22.5.7	Version 7.4.1.7	94
22.6	Version 7.3.1	94
22.7	Version 7.2.1	95
22.8	Version 7.1.4	96
22.9	Version 7.1.3	96
22.10	Version 7.1.2b	97
22.11	Version 7.1.2	97
22.12	Version 7.1.0a	98
22.13	Version 7.1.0	98
22.14	Version 7.0.10	99
22.15	Version 7.0.9	99
22.16	Version 7.0.8	100
22.17	Version 7.0.7	100
22.18	Version 7.0.6	100
22.19	Version 7.0.5	100
22.20	Version 7.0.4	100
22.21	Version 7.0.3	100
22.22	Version 7.0.2	101
22.23	Version 7.0.1a	101
22.24	Version 6.2.1a	101
22.25	Version 6.2.1	101
22.26	Version 6.2.0d	102
22.27	Version 6.1.1	103
22.28	Version 6.1.0	104

22.29	Version 6.0.10	105
22.30	Version 6.0.9	105
22.31	Version 6.0.8	105
22.32	Version 6.0.7	105
22.33	Version 6.0.6	106
22.34	Version 6.0.4	106
22.35	Version 6.0.3	106
22.36	Version 6.0.2	106
22.37	Version 6.0.1	107

2 Overview

The Softek Barcode Reader Toolkit for Windows is a set of dll files that can be used by applications to extract barcode values from image files and bitmaps held in memory. The toolkit uses no configuration files or registry settings and requires minimal [installation](#) on a production system. Some of the interfaces only require installation of a single dll file. Licenses are required for both development and run-time use of the toolkit – please refer to the section on [licensing](#) for more details.

The install set includes dll files for all interfaces to the toolkit, sample images and sample projects. Please note that the COM and OCX interfaces both require registration of a dll file – see the section on [Installation](#) for more details

The usual way to use the toolkit is to set a few properties, such as which types of barcode to read, and then call the [ScanBarcode](#) method to process the image. Note that by default the toolkit will exit when the first barcode is detected but it can be configured (see [MultipleRead](#)) to scan the entire document for barcodes, building up a list of detected values. The [ScanBarcode](#) method returns the number of detected barcodes and the value of each barcode can then be retrieved using the [GetBarString](#) method.

The following code sample shows how the toolkit is typically used:

```
` Set some properties, such as which barcodes should be scanned for  
  
barcode.SetLicenseKey("MY LICENSE KEY")  
barcode.SetReadCode39(true)  
barcode.SetReadDataMatrix(true)  
barcode.SetMultipleRead(true)  
  
` Scan for barcodes  
nBarCodes = barcode.ScanBarcode(inputFile)  
  
` Get the barcode values  
for i = 1 to nBarCodes  
do  
    strValue = barcode.GetBarString(i)  
next i
```

The sample projects are an ideal way to start learning about the toolkit. There are examples for VB.Net, VC++ and VC# along with other samples, some of which have been contributed by Softek's customers.

Softek offers free pre-sales support and 12 months support and upgrade cover after purchase – please refer to the section on [contact information](#) for more details.

3 Installation

3.1 Installation of the toolkit for Development

The toolkit is delivered as a self extracting installer containing dll files, sample images, sample projects and installers for redistribution of the dll files. The installer simply copies the toolkit files to the target folder.

Please note the following:

- No dll files in the toolkit are registered during the installation. Please see the section on [completing the installation](#) below.
- The toolkit can't be installed into a network drive because this might prevent the successful registration of components (if required).

The following table shows the contents of the installation folder:

images	Sample barcode images
java	Java class and sample java program
projects	Sample projects
README.TXT	Important information (similar to that contained in this section)
RunDemo.bat	Runs x86/SoftekSDKDemo.exe
Uninstall	Uninstallation exe files
x64	x64 based dll files and a demonstration application called SoftekSDKDemo.exe
x86	x86 based dll files and a 64-bit demonstration application called SoftekSDKDemo.exe
redist	Redistribution packages for x86 and x64 based systems.
documentation.pdf	This document
LICENSE.TXT	License document

3.1.1 Completing the Installation

Registration of dll files is only necessary in the following circumstances:

- The OCX interface (SoftekBarcode.ocx) to the toolkit is to be used.
- The COM interface (SoftekATL.dll) to the toolkit is to be used.
- The target system is running a 64-bit version of Windows and Adobe PDF documents are to be processed.

Note: If the win32 dll, .net or Java interfaces are to be used then no registration is necessary.

The dll files can be registered as follows:

1. In the x86-folder, right click on REGISTER.BAT and select *Run as Administrator*
2. If running on an x64 based system then do the same in the x64 folder

No further steps are necessary though you may wish to modify the systems PATH environment variable or copy some of the dll files to the windows system folder so that they can be accessed by applications. Please refer to the section on [Production Systems](#) for more details.

3.1.2 Demonstration Application

The toolkit can be demonstrated by running the application [SofttekSDKDemo.exe](#) (located in both the x86 and x64 folders).

3.1.3 Evaluation License Keys

30-day evaluation [license keys](#) can be obtained from sales@bardecode.com . Most interfaces can be used without a valid license key, though a pop up dialog box will be displayed when a barcode is read.

3.1.4 Sample Projects

The install set includes a number of sample projects. All of the sample projects directly under the projects folder know the relative location of the x86 and x64 dll files and should need no configuration in order to work, except for the COM and OCX based projects which will require registration of certain files (see [Completing the Installation](#) above).

The following table shows the sample projects available in the install set:

VB.Net using .net	Visual Studio 2005/2008/2010, Visual Basic using the .Net interface
VB.Net using com	Visual Studio 2005/2008/2010, Visual Basic using the COM interface (*)
VB.Net using win32 dll	Visual Studio 2005/2008/2010, Visual Basic using the win32 DLL interface
VC# using .net	Visual Studio 2005/2008/2010, Visual C# using the .Net interface
VC++ using .net	Visual Studio 2005/2008/2010, Visual C++ using the .Net interface
VC++ using ocx	Visual Studio 2005/2008/2010, Visual C++ using the OCX interface (*)
VC++ using win32 dll	Visual Studio 2005/2008/2010, Visual C++ using the win32 DLL interface
Other/ASP	ASP example using the COM interface (*)
Other/Visual Basic 6	Visual Basic 6 using the OCX interface (*)
Other/Visual Studio 2003 using VB.Net	Visual Studio 2003 using the .Net interface
Other/Contrib/Borland C++ using DLL	Borland C++ Builder using the win32 DLL interface
Other/Contrib/Delphi	Delphi 2009 using the win32 DLL interface

* Requires registration of certain dll files – please refer to [Completing the Installation](#) for further details.

3.2 Redistribution and Installation on Production Systems

There are 3 approaches that can be taken to installation on a production system:

1. Use a separate folder for the DLL files
2. Install the dll files into the windows system folders
3. Install a sub-set of the dll files into the same folder as the application

Each approach has its advantages, depending on the type of interface to be used, how the interfaces are used within the applications and whether or not 64-bit systems should be supported.

3.2.1 Using a Separate Folder for the DLL files

Storing all the dll files for the toolkit in a folder with a structure similar to the installation folder (as created by the self extracting installer) has the advantage of isolating the toolkit files from other dll files and can allow different versions of the toolkit to be used on the same system. For example, you may have a Softek sub-folder under an application, and in that folder have x86 and x64 sub-folders containing the necessary DLL files.

3.2.1.1 Win32 dll

The simplest approach is to add either the x86 or x64 folder to the system PATH so that applications can find the SoftekBarcode.dll file as needed. This works well so long as there isn't another version of the dll in the windows system folder. In this case it's still possible to load the correct version of the dll by using the LoadLibrary windows function (C++ projects may need to set the build option to delay loading the dll). Note that the dll will load other dll files from the same folder whatever the value of the PATH. This is the method used in the sample projects.

3.2.1.2 OCX and COM

In the case of the OCX and COM interfaces, there is no work to do other than to register the appropriate dll files. Please refer to [Completing the Installation](#) for more details.

3.2.1.3 .net

This approach is not applicable to the .net1 interface (implemented by SoftekBarcodeLib.dll) or the .net 2+ interface as implemented by SoftekBarcodeLib2.dll. It is however, well suited to the .net 2+ interface as implemented by SoftekBarcodeLib3.dll...

The BarcodeReader class constructor can take the location of the installation folder as an argument when the class is created. For example:

```
barcode = New SoftekBarcodeLib3.BarcodeReader("path\to\toolkit")
```

Note that the *path/to/toolkit* may be relative, so if you have a folder called Softek under the application folder and in that folder have x86 and x64 sub-folders, then you would create an instance as follows:

```
barcode = New SoftekBarcodeLib3.BarcodeReader("Softek")
```

The component will then load the correct supporting dll files from either the x86 or x64 folders. This is the method used in the sample projects. If the class is created without specifying the installation folder then either the x86 or x64 folders should be added to the system PATH according to the system architecture.

3.2.1.4 Java

The simplest approach is to add either the x86 or x64 folder to the system PATH so that Java can find the barcode_jni.dll and SoftekBarcode.dll files.

3.2.1.5 PDF File Support

No configuration is necessary on x86 based systems. For x64 based systems please run REGISTER.BAT as the administrator in the x86-folder.

3.2.2 Installation in Windows System Folders

Installing the dll files into the Windows System Folders has the advantage of making them immediately available to all applications but does have the potential to interfere with other applications that use the toolkit. The *redist* folder contains install sets for both x86 and x64 based systems (*softek_barcode_redist_x86.exe* and *softek_barcode_redist_x64.exe*). If PDF documents need to be processed on an x64 based system then please install both the x86 and x64 redistributable packages. Note that the DLL files used in these installers are identical to those found in the x86 and x64 folders.

It is also possible to work out which files need to be installed for a given interface with reference to the tables in [Appendix C](#).

3.2.3 Installation in the Same Folder as Application

With this approach, only the specific DLL files necessary for the interface being used are installed into the same folder as the application. The tables in [Appendix C](#) detail the dependencies between the DLL files for the various interfaces. Note that this approach does not work on x64 systems if PDF File support is required.

For example, if the .net interface implemented by SoftekBarcodeLib3.dll is being used, and PDF File support is not required, then the only files required are SoftekBarcodeLib3.dll and SoftekBarcode.dll. In the application the barcode object should be created without giving a path to the installation folder (e.g `barcode = New SoftekBarcodeLib3.BarcodeReader()`).

3.2.3.1 PDF File Support

On an x86 based system, copy the following files from the x86-folder to the same folder as the application:

SoftekBarcodePDF.dll, Cimage.dll, Convert.dll, Encryptpdf.dll, Pdf2image.dll and Pdf2Tif.dll

This method is not possible for an x64 based system.

3.3 Upgrading

3.3.1 License Key

Version 7.4.1 of the Softek Barcode Reader Toolkit for Windows is the first version of the toolkit to require a license key to be set at run-time. When upgrading to this version you will need to modify your code to set the [license key](#) property prior to calling the ScanBarCode method.

3.3.2 DLL Dependencies

Please note that all components of this version have been built using Visual Studio 2008 and will therefore have different requirements to previous versions. Please refer to the table in [Appendix C](#) for further details.

Also, note that if one DLL file is updated then so must all other DLL files. For example, this version of SoftekBarcode.dll will not function with previous versions of SoftekBarcodePDF.dll.

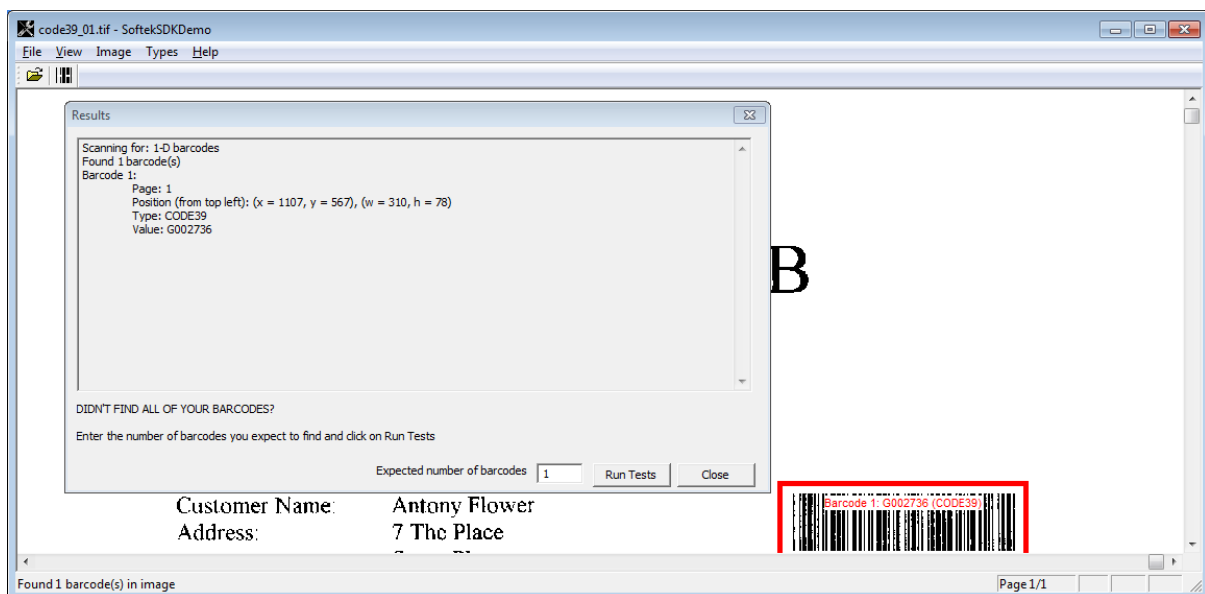
4 Evaluating the SDK

4.1 Using the SoftekSDKDemo application

If you are evaluating the SDK then the first thing you might wish to do is test out your images with the SoftekSDKDemo application. This application displays image files, allows you to alter the settings in the toolkit and displays details of any barcodes found in an image.

To start the demonstration application, navigate to the x86-folder and double click on SoftekSDKDemo.exe. There is also a 64-bit version in the x64 folder. When the application starts, a file browser window will open and you should select an image file that you wish to test with the SDK and click *Open*. The first page of the image will then be displayed.

Pressing the *Enter* key will scan the displayed page for common 1-D barcodes and display the results...



If the application wasn't able to find your barcode in the image then enter the number of barcodes you expected to find and click *Run Tests* – the application will then try a number of different settings and generate a report.

Click on *Close* to dismiss the results window. The results can be viewed at any time by clicking on the *View* menu and selecting *Results*.

Settings for the SDK can be altered by clicking on the *File* menu and selecting *Settings* or using the short-cut Ctrl-S...

The label against a setting generally indicates the associated property and where this is not the case the help text underneath should explain. For example, "Read Code 39" will set the property ReadCode39.

It's also possible to scan an entire document for barcodes (rather than just the current page) by selecting the "Scan all pages for barcodes" option from the File menu.

Image Display Options

Use the Image menu to move between pages of an image, change the size of the image on the screen or rotate the image. The menu can also be accessed by right clicking anywhere on the image.

4.2 Integration into applications

The download also includes everything needed to integrate the SDK into an application. The projects folder contains sample projects for Visual Studio and other environments. These are a good starting point before moving on to integrate the SDK with an existing application. Temporary license keys can be obtained from sales@bardecode.com – without a key the SDK will display a pop-up box each time it scans an image for a barcode. Should you require any assistance with the integration process then please contact support@bardecode.com.

5 Licensing

A license is required for each computer where any of the toolkit dll files will be installed – either for development or run-time use.

In this section:

- *Toolkit* refers to the Softek Barcode Reader Toolkit for Windows.
- *Server version(s) of Windows* means Windows Server 2003, Windows Server 2008 or any future version of Windows that includes the word “Server” in the title or is marketed as a server operating system.
- *Desktop version(s) of Windows* means any version of Windows that is not a *Server version*, such as Windows XP, Windows Vista or Windows 7.

There are deals available for unlimited site and distribution licenses. The Site and Distribution licenses allow for unlimited use of the barcode reader toolkit in return for a one-off payment. Note that Site and Distribution Licenses are available in Desktop and Server versions.

All of the licenses are available either with or without support for Adobe PDF documents.

5.1 License Keys

License keys are supplied when a license is purchased or requested for evaluation and are applied by setting the LicenseKey property before the ScanBarCode method is called. For example:

```
barcode.SetLicenseKey("MY LICENSE STRING")  
barcode.ScanBarCode("input.tif")  
etc etc
```

5.2 Desktop Developer License

This license is required for each Desktop version of Windows that will be used for development with the Toolkit, and is available in packs of 1, 4 and 8. Note that this license will not function on Server versions of Windows.

5.3 Desktop Run-Time License

This license is for each Desktop version of Windows that will make run-time use of the Toolkit, and is available in packs of 1, 4 and 8. Note that this license will not function on Server versions of Windows.

5.4 Server License

A Server license is required for any installation of the Toolkit on a Server version of Windows.

5.5 Desktop Distribution License

A Desktop Distribution License allows unlimited run-time use of the Toolkit on Desktop versions of Windows, in return for a one-time payment. Note that this license will not function on Server versions of Windows.

5.6 Desktop and Server Distribution License

A Desktop and Server Distribution License allows unlimited run-time use of the Toolkit in return for a one-time payment. Note that this license will work on both Desktop and Server versions of Windows.

5.7 Site License

A site license allows unlimited use of the Toolkit at a single physical location i.e. unlimited Desktop Developer, unlimited Desktop Run-Time and unlimited Server licenses.

5.8 Examples

1. A software company has 1 developer and needs run-time licenses for 4 Desktop versions of Windows.

Licenses required: 1 Desktop Developer License + 4 Desktop Run-Time Licenses

2. A software company has 1 developer and needs to deploy a license on a fax server.

Licenses required: 1 Desktop Developer License + 1 Server License

3. A software company wish to add barcode recognition to their applications. They have a team of 4 developers who will need to work with the toolkit and would like to to distribute run-time licenses on Desktop versions of Windows without paying any further fees.

Licenses required: 4 Desktop Developer License + Unlimited Desktop Distribution License

4. A company wishes to develop and deploy the barcode toolkit within their own organization. They are based at a single location and will require a large number of run-time licenses for both Desktop and Server versions of Windows.

License required: 1 Site License

5. Sames a example 4 above, but the company needs to deploy their application at a number of locations within the organization. There is a development team of 8 people.

License required: 8 Desktop Developer License + 1 Desktop and Server Distribution License.

6 Contact Information

6.1 Sales

Softek Software's online store can be found at the following URL:

<http://www.bardecode.com/en/purchase.html>

For all sales enquiries please contact sales@bardecode.com or call +44 1296 663 633. Softek Software can provide quotations on request and can also accept payment by wire transfer or check.

6.2 Support

Softek Software offers free pre-sales support for the Barcode Reader Toolkit and all licenses include 12 months support and upgrade cover.

Softek Software online knowledge base can be found at the following URL:

<http://www.bardecode.com/en/knowledge-base.html>

For all support enquiries please contact support@bardecode.com

7 Interfaces to the Toolkit

7.1 Windows DLL

The Windows DLL interface is provided by the file SoftekBarcode.dll, which can be installed as a stand-alone component. The first call to the dll should be to create a handle to the barcode reader sdk, using the function `mtCreateBarcodeInstance`. Further calls to the dll will then take the handle as the first parameter. There is no need to register the dll on installation.

Example:

```
hBarcode = mtCreateBarcodeInstance()  
nBarcodes = mtScanBarCode(hBarcode, "input.tif")  
...process the result in some way (see below)  
mtDestroyBarcodeInstance(hBarcode)
```

Note that the names of the above functions start with "mt" and are safe to use in multi-threaded applications.

For each function in the DLL whose name begins with "mt" there is an equivalent function beginning with "st" (except for `mtCreateBarcodeInstance` and `mtDestroyBarcodeInstance`); these functions require no `hBarcode` handle but are not recommended for multi-threaded applications, though the above code would be simplified to the single line:

```
nBarcodes = stScanBarCode("input.tif")
```

Properties of the toolkit are accessed using `Set` and `Get` functions in the DLL:

```
mtSetPropertyName(hBarcode, value)  
value = mtGetPropertyName(hBarcode)
```

e.g.

```
mtSetReadCode39(hBarcode, 1)  
value = mtGetReadCode39(hBarcode)
```

7.1.1 Handling String Values

Strings passed to and returned from the DLL are 'C' style strings i.e. pointers to an array of bytes terminated by a byte with value 0 (null). This means that Visual Basic applications need to convert the returned values as follows:

VB6 applications can use a function such as `nullTrim`:

```
nBarCodes = mtScanBarCode(hBarcode, "input.tif")  
  
For i = 1 To nBarCodes  
    strBarcode = nullTrim(mtGetBarString(hBarcode, i))  
Next i
```

```

Function nullTrim(s As String) As String
' Trim the string to the null character
Dim n As Integer
n = InStr(s, Chr$(0))
If (n > 0) Then
    nullTrim = Trim(Left(s, n - 1))
Else
    nullTrim = Trim(s)
End If
End Function

```

VB.Net applications should use code similar to:

```

strBarcode = System.Runtime.InteropServices.Marshal.PtrToStringAnsi(mtGetBarString (hBarcode,
i)))

```

7.1.2 Distribution

Please note that distribution of this file is strictly subject to license.

```

x86:          x86/SoftekBarcode.dll
x64:          x64/SoftekBarcode.dll

```

Registration: no

Dependencies:

- None

See also:

- [Appendix C: Installation Files](#)

7.2 OCX/ActiveX

The ocx interface is provided by the file SoftekBarcode.ocx, which in turn wraps around the SoftekBarcode.dll interface. The ocx is typically used in VB6 projects where it provides access to the toolkit through an object and string handling is also simpler.

VB6 Example:

```

nBarCodes = SoftekBarcode1.ScanBarCode(Path)
For i = 1 To nBarCodes
    value = SoftekBarcode1.GetBarString(i)
Next i

```

7.2.1 Distribution

Please note that distribution of this file is strictly subject to license.

```

x86:          x86/SoftekBarcode.ocx
x64:          N/A

```

Registration: yes

Dependencies:

- [SoftekBarcode.dll](#)

Notes:

There is no x64 version of the ocx interface

See also:

- [Appendix C: Installation Files](#)

7.3 COM Object

The com object is provided by the file SoftekATL.dll, which in turn wraps around the SoftekBarcode.dll interface. The com object is versatile way to access the toolkit from VB6, VB.Net, VC++, C# and ASP.

There are some differences in the way in which the com object is called. The return values from some of the methods are stored in properties, for example:

```
oBarcode = New SoftekATL.CBarcode
oBarcode.ScanBarCode(Path.ToString)
nBarCodes = oBarcode.BarCodeCount
```

And the page number for the nth barcode is obtained in the following way:

```
nPage = oBarcode.BarStringPage(i)
```

In an ASP page you can create the barcode object in the following way:

```
Set barcode = Server.CreateObject("SoftekATL.Barcode")
```

7.3.1 Distribution

Please note that distribution of this file is strictly subject to license.

x86: x86/SoftekATL.dll

x64: x64/SoftekATL.dll

Registration: yes

Dependencies:

- [SoftekBarcode.dll](#)

See also:

- [Appendix C: Installation Files](#)

7.4 .Net 2+ Components

There are 2 components that can be used for .applications using the .net framework 2.0 or higher: SoftekBarcodeLib2.dll and SoftekBarcodeLib3.dll. The comparison chart below compares the 2 components:

	SoftekBarcodeLib2.dll	SoftekBarcodeLib3.dll
Stand-alone	Yes (except when processing PDF files)	No – acts as a wrapper to Softekbarcode.dll
Architecture independent	No – different versions are needed for x86 and x64 based systems (more...)	Yes – though still needs to wrap around correct version of SoftekBarcode.dll (more...)
Speed	Slower	Faster
Supported Image Formats	Uses .Net framework to load images which can be restricting if using color TIF files.	Supports wider range of image formats.
Requires Microsoft Visual C++ 2008 SP1 Redistributable Package	Yes	No

These components can be used in .net applications developed with Visual Studio 2005 or later. To use the component a reference should be added in the project to the SoftekbarcodeLib2.dll or SoftekBarcodeLib3.dll files and the barcode object can then be created in the following way:

For SoftekBarcodeLib2.dll:

```
barcode = New SoftekBarcodeLib2.BarcodeReader
nBarCodes = barcode.ScanBarCode(Path)
For i = 1 To nBarCodes
    value = barcode.GetBarString(i)
Next i
barcode.ReleaseResources()
```

And for SoftekBarcodeLib3.dll:

Either:

```
barcode = New SoftekBarcodeLib3.BarcodeReader
nBarCodes = barcode.ScanBarCode(Path)
For i = 1 To nBarCodes
    value = barcode.GetBarString(i)
Next i
```

...which assumes that dll files needed by the component are in the windows system folder.

Or:

```
barcode = New SoftekBarcodeLib3.BarcodeReader(pathToSoftekBarcodeSDK)
nBarCodes = barcode.ScanBarCode(Path)
For i = 1 To nBarCodes
```

```
value = barcode.GetBarString(i)
```

```
Next i
```

...where *pathToSoftekBarcodeSDK* is the path to a folder storing the x86 and x64 dll files in folders of the same name (i.e. it should have 2 sub-folders named x86 and x64, storing the appropriate dll files).

7.4.1 Distribution of SoftekBarcodeLib2.dll

Please note that distribution of this file is strictly subject to license.

SoftekBarcode

x86: x86/SoftekBarcodeLib2.dll

x64: x64/SoftekBarcodeLib2.dll

Registration: no

Dependencies:

- Visual C++ 2008 SP1 Redistributable Package
- Requires SoftekBarcode.dll if documents are to split into PDF format.

See also:

- [Using the .net components on x86 and x64 systems](#)
- [Appendix C: Installation Files](#)

7.4.2 Distribution of SoftekBarcodeLib3.dll

Please note that distribution of this file is strictly subject to license.

SoftekBarcode

AnyCPU anycpu/SoftekBarcodeLib3.dll

Registration: no

Dependencies:

- SoftekBarcode.dll

See also:

- [Using the .net components on x86 and x64 systems](#)
- [Appendix C: Installation Files](#)

7.5 .Net 1 Component

The .net 1 interface is provided by SoftekBarcodeLib.dll, which can be installed as a stand-alone component. This component can be used in .net applications developed with Visual Studio 2003. To

use the component a reference should be added in the project to the SoftekbarcodeLib.dll file and the barcode object can then be created in the following way:

```
barcode = New SoftekBarcodeLib.BarcodeReader
nBarCodes = barcode.ScanBarCode(Path)
For i = 1 To nBarCodes
    value = barcode.GetBarString(i)
Next i
```

NOTE: The .Net 1 component is not able to process PDF Documents.

7.5.1 Distribution

Please note that distribution of this file is strictly subject to license.

x86: x86/SoftekBarcodeLib.dll
x64: Not available

Registration: no

Dependencies:

- None

See also:

- [Appendix C: Installation Files](#)

7.6 Java Class

The Java interface is an interface to SoftekBarcode.dll through a barcode class (Softek/Barcode.class) and a Java Native Interface (bardecode_jni.dll).

An instance of the Barcode class can be created in the following way:

```
Softek.Barcode barcode = new Softek.Barcode();
```

...and then used to read barcodes as follows:

```
n = barcode.ScanBarCode("image.tif");
for (i = 0; i < n; i++)
{
    value = barcode.GetBarString(i + 1);
}
```

7.6.1 Distribution

Please note that distribution of this file is strictly subject to license.

Any cpu: java/Softek/Barcode.class

x86: x86/bardecode_jni.dll
x64: x64/bardecode_jni.dll

Registration: no

Dependencies:

- [SoftekBarcode.dll](#)

See also:

- [Appendix C: Installation Files](#)

8 Using the .net components on x86 and x64 systems

One of the major differences between the SoftekBarcodeLib2.dll and SoftekBarcodeLib3.dll components is in deployment.

8.1 SoftekBarcodeLib2.dll

There are separate versions of SoftekBarcodeLib2.dll for x86 and x64 based system, so care needs to be taken to ensure that applications reference the correct version in any given installation. There are two approaches that can be taken to using this component in Visual Basic .Net and Visual C# projects. In Visual C++ projects the x86 and x64 versions of the .net assembly may need to be switched manually.

8.1.1 Approach 1: Build for “Any CPU”

If you want to build your application for “Any CPU” and ensure that you reference the correct .net assembly on either an x86 or x64 based system then it is recommended that you do the following:

1. On an x86 based system install the x86 version of SoftekBarcodeLib2.dll in C:\windows\system32
2. On an x64 based system install the x86 version of SoftekBarcodeLib2.dll in C:\windows\SysWOW64 and the x64 version in C:\windows\system32
3. In your application, set the reference path to search c:\windows\SysNative first, followed by c:\windows\System32 second. This will ensure that the correct component is loaded for the correct architecture.

8.1.2 Approach 2: Build for “x86” or “x64”

This approach means that you need to target a specific architecture but has the advantage of working better with the publish wizard in visual studio.

1. Change the configuration for your project to either x86 or x64.
2. Add a reference to the appropriate version of SoftekBarcodeLib2.dll from the toolkit in either the x86 or x64 folders.
3. Close the project
4. Open the .vbproj or .csproj file with notepad, search for SoftekBarcodeLib2.dll and edit the HintPath, replacing x86 or x64 with \$(Platform).

Example change to HintPath:

Before:

```
<HintPath>..\x86\SoftekBarcodeLib2.dll</HintPath>
```

After:

```
<HintPath>..\$(Platform)\SoftekBarcodeLib2.dll</HintPath>
```

When you next open the project you should be able to switch between x86 and x64 targets and build/publish your application for either target.

8.2 SoftekBarcodeLib3.dll

The SoftekBarcodeLib3.dll component is architecture independent though it does rely on the SoftekBarcode.dll file – which has separate x86 and x64 versions. Again, there are 2 approaches to using the component on x86 and x64 systems:

8.2.1 Approach 1: Install SoftekBarcode.dll etc in the Windows system folders

This approach relies on Windows to load the correct version of SoftekBarcode.dll appropriate to the system architecture:

	Target folder for x86/SoftekBarcode.dll file	Target folder for x64/SoftekBarcode.dll file
x86 based system	windows\system32	N/A
x64 based system	windows\SysWOW64	windows\system32

And create the SoftekBarcodeLib3.BarcodeReader class as follows:

```
barcode = New SoftekBarcodeLib3.BarcodeReader
```

Please refer to [Installation Files](#) for more details.

8.2.2 Approach 2: Store SoftekBarcode.dll etc in a specific folder

This approach allows you to store all the dll files for the toolkit in a single folder, with a structure identical to the install set supplied by Softek. When an instance of the BarcodeReader class is created the path to this folder must be specified:

```
barcode = New SoftekBarcodeLib3.BarcodeReader(pathToSoftekBarcodeSDK)
```

pathToSoftekBarcodeSDK should contain 2 sub-folders named x86 and x64, holding all the supporting dll files for the toolkit.

This approach also allows different versions of the toolkit engine to be used in isolation.

8.3 Handling PDF Documents in ASP on x64 Systems

This section describes a problem that can be seen on ASP applications that handle PDF documents on x64 systems.

Symptom:

ScanBarCode returns -1 and GetLastError returns 3001.

Description:

CoCreateInstance is failing and is probably returning E_ACCESSDENIED. This is because IIS_IUSRS does not have permission for Local Launch or Local Activation of either the PDF2Image or PDF2TIFCom COM servers which are used by the SDK to convert PDF documents.

Solution:

Grant access for the IIS_IUSRS user to launch and activate the above COM servers:

Run dcomcnfg.exe

From Console Root, expand Component Services/Computers/My Computer/DCOM Config

Scroll down to locate PDF2Image.CPDF2Image, right click on the item and select Properties.

Click on the Security tab

In Launch and Activate Permissions select Customize and click on Edit.

If IIS_IUSRS is not in the list then click on Add and enter IIS_IUSRS before clicking on OK.

Grant IIS_IUSRS permission for Local Launch and Local Activation.

Click on OK to exit from Launch and Activate Permissions and then click on OK to exit from Properties.

Repeat for PDF2TIFCom.CPDF2TIFCom

9 Supported Image Formats

9.1 TIFF

The following compressions are supported in the DLL, .Net (SoftekBarcodeLib3.dll), COM, OCX and Java interfaces:

- Uncompressed
- LZW (Lempel-Ziv-Welch)
- Packbits (run length encoding)
- Jpeg
- ZIP
- CCITT Fax 3/Group 3
- CCITT Fax 4/Group 4

The .Net interfaces provided by SoftekBarcodeLib.dll and SoftekBarcodeLib2.dll use the .Net framework to load image files into memory and generally support a more limited set of TIF compressions.

9.2 BMP

The SDK supports single plane BMP files in either 1-bit (black and white), 8-bit (gray scale), 24 bit (color) or 32-bit (color) format.

9.3 Adobe PDF

Please see [Reading Barcodes from Adobe PDF Documents](#)

9.4 Other formats

The SDK also support images in Jpeg, GIF and PNG format.

10 Supported Barcode Formats

10.1 1-D Barcode Formats

The following 1-D barcode formats are supported by the SDK (with corresponding properties given in brackets):

- Codabar also known as Code 2 of 7, Codeabar, Ames Code, NW-7 and Monarch ([ReadCodabar](#))
- Code 128 Symbol Sets A, B and C ([ReadCode128](#))
- Code 128 Short Format ([ReadShortCode128](#))
- Code 2 of 5 Datalogic ([ReadCode25ni](#))
- Code 2 of 5 Iata1 ([ReadCode25ni](#))
- Code 2 of 5 Iata2 ([ReadCode25ni](#))
- Code 2 of 5 Industrial ([ReadCode25ni](#))
- Code 2 of 5 Interleaved ([ReadCode25](#))
- Code 2 of 5 Matrix ([ReadCode25ni](#))
- Code 3 of 9 ([ReadCode39](#))
- Code 3 of 9 Extended ([ReadCode39](#) and [ExtendedCode39](#))
- Code 93 ([ReadCode93](#))
- EAN-8, European Article Number/International Article Number ([ReadEAN8](#))
- EAN-13 and UPC-A, European Article Number/International Article Number ([ReadEAN13](#))
- GS1-128, UCC-128, EAN-128 ([ReadCode128](#))
- GS1-Databar (please see [2-D section](#) below)
- Patch Code Symbols ([ReadPatchCodes](#))
- UPC-A, Universal Product Code ([ReadEAN13](#) and [ReadUPCA](#))
- UPC-E, Universal Product Code ([ReadUPCE](#))

10.2 2-D and Stacked Barcode Formats

The following 2-D and stacked barcode formats are also supported:

- QR-Code ([ReadQRCode](#))
- Data Matrix ECC200 sizes 8x8 to 144x144 ([ReadDataMatrix](#))
- GS1-Databar or Reduced Space Symbology. Omnidirectional, Stacked Omnidirectional, Expanded, Expanded Stacked and Limited ([ReadDatabar](#))
- Micro-PDF-417 ([ReadMicroPDF417](#))
- PDF-417, Portable Data File ([ReadPDF417](#))

11 Reading Barcodes from Adobe PDF Documents

It is possible to read barcodes from PDF documents by purchasing a license to the PDF Extension to the Softek Barcode Reader Toolkit. After purchasing the license you will receive a new [license key](#) for the toolkit which will enable the processing of PDF documents using the [ScanBarcode](#) function.

The processing of a PDF document is no different to the processing of any other file format. For example:

```
n = barcode.ScanBarcode("input.pdf")
if (n > 0)
    value = barcode.GetBarString(1)
```

By default the toolkit assumes that it will be processing image only PDF documents, e.g, scanned images. Making this assumption allows the toolkit to quickly strip images out of documents at the same size and resolution as the original. If the toolkit is unable to extract any images using this method then it reverts to the method used in earlier versions of the toolkit whereby the PDF document is rendered into image format.

The following properties of the toolkit are specific to the processing of PDF documents:

[PdfImageOnly](#)

Defaults to true and indicates that the toolkit should consider the input PDF documents to be scanned images. When set to false the toolkit will render a PDF document into image format.

[PdfImageExtractOptions](#)

A mask that controls the way in which images are extracted from PDF documents when PdfImageOnly is set to True.

[PdfBpp](#)

Controls the color depth of an image rendered from a PDF document. Has no effect on the color depth of images extracted from PDF documents.

[PdfDpi](#)

Controls the resolution of an image rendered from a PDF document. Has no effect on the color depth of images extracted from PDF documents.

Information handling PDF documents in ASP on x64 systems can be found [here](#).

12 Understanding Confidence Levels for Barcode Reading

As the toolkit scans the images in a document, it assigns a score to each barcode-like pattern it finds. The score for a very clear barcode will typically be in the region 20 to 100, depending on the size of the barcode and the settings used within the sdk. The PrefOccurrence and MinOccurrence properties specify the scores for the toolkit to use when deciding which barcodes to report and which to ignore. The PrefOccurrence property is the preferred occurrence for a barcode and defaults to value of 5. Any barcode with a score greater than or equal to PrefOccurrence will be reported by the toolkit. Some documents contain difficult to read barcodes with very low scores - lower than PrefOccurrence. In this case (where all of the barcodes have scores lower than PrefOccurrence) the toolkit will report the barcode with the highest score - so long as this figure is greater than or equal to MinOccurrence (minimum occurrence).

Consider the case of a 3 page document with barcodes on each page:

Page 1: Barcode 0001 with score 12

Page 2: Barcode 0002 with score 4

Page 3: Barcode 0003 with score 7

With default settings except for MultipleRead = true, the toolkit will only report the barcodes on pages 1 and 3 because the barcode on page 2 has a score less than PrefOccurrence (5).

If we now set PageNo = 2 (so we only scan page 2) then the barcode "0002" is reported, because no other barcodes were found with a score \geq PrefOccurrence (5) and the score for this barcode is \geq MinOccurrence (2).

With PageNo back to 0 (scan all pages) and PrefOccurrence = 4 the toolkit will report all 3 barcodes because all the scores are now \geq PrefOccurrence (4).

In conclusion, the purpose of MinOccurrence is to allow the capture of poor quality barcodes whilst minimising the number of false positive readings in documents containing good quality barcodes.

13 Reading Barcodes from Color Images

In general, it is easier to read a barcode from a higher resolution bi-tonal image than from a lower resolution color image. Color images often have low contrast levels between the black and white bars and can often be degraded by compressions such as jpeg. This can make it difficult for a barcode reader to determine the relative widths of the black bars and white spaces, which can make it especially hard to decode barcode types such as Code 128 or UPC/EAN.

There are 2 key properties involved when scanning a color image for a barcode - [ColorThreshold](#) and [ColorProcessingLevel](#). ColorThreshold can be used to manually set a threshold at which a pixel value is considered black or white, but it is normally left at the default value of 0, which allows the toolkit to assess the threshold level for itself through the ColorProcessingLevel property. This can vary from 0 to 5, with higher values yielding best results but taking longer than lower values.

Note that for QR-Code recognition a ColorProcessingLevel of 1 is recommended for the best balance of speed to accuracy.

14 Splitting Documents According to Barcode Position

The TifSplit feature of the toolkit allows you to use barcodes as document separators in both TIF and PDF documents. The input file is scanned for barcodes and then split into a number of smaller documents.

There are 2 properties that control how the input file is split:

Setting [TifSplitPath](#) turns the feature on and controls where the new documents will be created.

The path can contain the tokens %d and %s:

- %d is replaced by a 1-based sequence number
- %s is replaced by the value of the barcode separator

e.g c:\tmp\Output%d.tif will create files Output1.tif, Output2.tif etc

The format of the output document is determined by the file extension used for the TifSplitPath property. TIF files may be split into either TIF or PDF format where as PDF documents may only be split into PDF format.

[TifSplitMode](#) controls how the input pages are copied to the output documents.

- Mode=0 creates output files that contain a barcode on page 1 (the first output file will always start with page 1 of the input file).
- Mode=1 creates output files that contain a barcode on the last page (the last output file will always end with the last page of the input file).
- Mode=2 creates output files that contain no barcodes. A new output document is started each time the software finds a barcode in the input file.

Example:

Suppose there is a 6 page TIF file with barcodes on pages 2 and 5. The barcode on page 2 has the value "AAAAAA" and the barcode on page 5 has the value "BBBBBB"

```
barcode.TifSplitPath = "C:\tmp\Output%s_%d.tif"  
barcode.TifSplitMode = 0  
barcode.ScanBarCode(InputPath)
```

...will create 3 output files. Output_1.tif will contain page 1, OutputAAAAAA_2.tif will contain pages 2, 3 and 4, and OutputBBBBBB_3.tif will contain pages 5 and 6.

```
barcode.TifSplitPath = "C:\tmp\Output%s_%d.tif"  
barcode.TifSplitMode = 1  
barcode.ScanBarCode(InputPath)
```

...will create 3 output files. OutputAAAAAA_1.tif will contain pages 1 and 2, OutputBBBBBB_2.tif will contain pages 3, 4 and 5, and Output_3.tif will contain page 6.

```
barcode.TifSplitPath = "C:\tmp\Output%s_%d.tif"  
barcode.TifSplitMode = 2  
barcode.ScanBarCode(InputPath)
```

...will create 3 output files. Output_1.tif will contain page 1, OutputAAAAAA_2.tif will contain pages 3 and 4, and OutputBBBBBB_3.tif will contain page 6.

15 Tips on Reading Barcodes

Some images may require non-default values for certain properties. This section describes some common issues with barcode images and suggests some possible solutions. It may be the case that no single set of properties is able to process all documents in a batch, if so then it's worth considering using [xml settings](#).

15.1 Skewed Barcodes



By default, the toolkit does not read barcodes that are badly skewed as in the above example.

Tip: try setting [SkewTolerance](#) to a value between 1 and 5. You may also need to decrease the value of [SkewLineJump](#) to a value such as 1 though this will slow down the recognition speed.

15.2 Badly defined edges to bars



In these cases the bars have joined together in certain places, which makes it difficult for the toolkit to separate the bars and determine their relative sizes.

Tip: Try setting [NoiseReduction](#) to a value between 10 and 20.

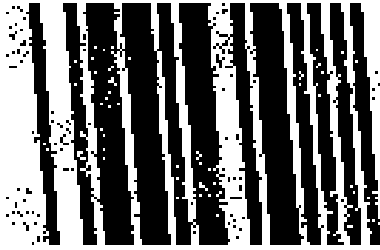
15.3 Noisy background to the image



Similar to the above problem except the entire image will typically contain black dots in the background.

Tip: Try setting [NoiseReduction](#) to a value between 5 and 10.

15.4 White speckles in the black bars



In these cases the image will typically contain black dots in the background and white dots in the black bars.

Tip: Try setting [NoiseReduction](#) to a value between 5 and 10 and [Despeckle](#) to True.

Black speckles in the spaces between bars

White speckles in the bars

Lines and other marks close to the left or right hand end of the barcode



Most barcodes should have a quiet zone around the barcode, to distinguish the barcode from the rest of the image – however, as in the above example it's not unusual for barcodes to be printed in boxes which may result in the barcode being ignored by the toolkit.

Tip: Measure the distance in pixels between the barcode and the edge of the box and try setting [QuietZoneSize](#) to a value slightly less than this distance. Typically this might be a value between 10 and 20. Use values less than 5 with care because they can result in false positive readings for certain barcode types.

16 Reading Barcodes from Bitmaps Held in Memory

The toolkit is capable of reading barcode values directly from bitmaps held in memory through the functions [ScanBarcodeFromBitmap](#) and [ScanBarcodeFromDIB](#).

[ScanBarcodeFromBitmap](#) takes a handle to a bitmap object and in the case of the .net interface can also accept a System.Drawing.Bitmap object.

[ScanBarcodeFromDIB](#) takes a handle to a device independent bitmap.

A handle to a bitmap (HBITMAP) can be created by using the Microsoft GDI function CreateBitmap:

```
HBITMAP CreateBitmap(  
    int nWidth,  
    int nHeight,  
    UINT cPlanes,  
    UINT cBitsPerPel,  
    CONST VOID* lpvBits  
);
```

Notes:

- nWidth should be divisible by 4
- cPlanes should be 1
- cBitsPerPel should be either 1, 8, 24 or 32. A value of 8 indicates a gray scale image. A value of 1 indicates a bi-tonal image where a bit value of 0 represents a black pixel unless the [Photometric](#) property is modified.

A DIB handle can be created by using the GDI function CreateDIBSection.

17 Using XML Files to store sets of properties

It's possible that an application may need to try more than one set of properties on a set of images in order to maximize the read rate. One way to do this is to call ScanBarCode repeatedly, adjusting the settings between each call. Another way is to store the groups of settings in an xml file and call [LoadXMLSettings](#) prior to calling ScanBarCode. This causes ScanBarCode to load each page into a memory bitmap and then try each set of properties until a barcode is found – at which point the next page is loaded and the process repeated. The [LoadXMLSettings](#) reference page contains detailed information of the format of the xml file.

18 Unicode File Paths

Versions 7.4.1.7 and later contain improved support for Unicode file paths – or more accurately, file paths that contain non-ascii characters.

Support in the various interfaces to the toolkit can be detailed as follows (excluding file paths for PDF documents, which are handled below):

There is full support for Unicode paths in the .net interfaces (SoftekBarcodeLib2 and SoftekBarcodeLib3).

The win32 DLL interface can handle files paths encoded using UTF-8 if the FilePathEncoding property has been set to a value of 1. The same applies to the COM and OCX interfaces because they act as wrappers around the win32 DLL.

The Java interface has full support for Unicode file paths.

18.1 PDF Documents

The component that processes PDF documents can currently only accept ASCII file paths, however, if the file system supports short path names then the toolkit will convert the Unicode path to its short form and use that to load the document.

If short paths are not available then the toolkit will attempt to load the document using an ASCII conversion of the path. This is often successful if the path contains Latin characters but will not work for other symbol sets such as Chinese.

19 Appendix A: Methods Reference

19.1 List of methods

<u>CreateBarcodeInstance</u>	create a instance of the barcode toolkit (win32 dll only)
<u>DestroyBarcodeInstance</u>	destroy an instance of the barcode toolkit (win32 dll only)
<u>ExportXMLSettings</u>	save settings to an xml file
<u>GetBarString</u>	get a barcode value
<u>GetBarStringPos</u>	get the position of a barcode
<u>GetBarStringType</u>	get the type of a barcode
<u>GetBarStringDirection</u>	get the orientation of a barcode
<u>GetLastError</u>	get the last error number for the toolkit
<u>GetLastWinError</u>	get the last windows error number
<u>LoadXMLSettings</u>	load settings from an xml file
<u>ProcessXML</u>	process files and folders specified in an xml file
<u>SaveResults</u>	save the results of barcode reading to an xml or csv file
<u>ScanBarCode</u>	scan an image file for barcodes
<u>ScanBarCodeFromBitmap</u>	scan a bitmap held in memory for barcodes
<u>ScanBarCodeFromDIB</u>	scan a bitmap held in memory for barcodes
<u>SetScanRect</u>	specify the portion of an image to scan for barcodes

19.2 CreateBarcodeInstance

Syntax

mtCreateBarcodeInstance()

Return Value

Handle to an instance of the barcode toolkit (dll only).

Remarks

Create an instance of the barcode toolkit and return a handle that may be used with other functions in the win32 dll interface. Please note that this function is not applicable to other interfaces for the toolkit. This function only exists in the dll interface to the SDK.

Example:

```
Dim hBarcode As System.IntPtr
hBarcode = mtCreateBarcodeInstance()
mtSetReadCode39(hBarcode, True)
```

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtCreateBarcodeInstance Lib "Softekbarcode" () As System.IntPtr
```

Visual C++:

```
extern "C" {
HANDLE __stdcall mtCreateBarcodeInstance();
}
```

See also:

[DestroyBarcodeInstance](#)

19.3 DestroyBarcodeInstance

Syntax

mtDestroyBarcodeInstance(hBarcode)

Parameters

hBarcode Handle to an instance of the barcode toolkit.

Return Value

1/True on success and 0/False on failure.

Remarks

DestroyBarcodeInstance destroys an instance of the barcode toolkit and releases any resources used by the toolkit. This function only exists in the dll interface to the SDK.

Example:

```
Dim hBarcode As System.IntPtr  
hBarcode = mtCreateBarcodeInstance()  
mtDestroyBarcodeInstance(hBarcode)
```

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtDestroyBarcodeInstance Lib "Softtekbarcode" (ByVal hBarcode As  
System.IntPtr) As Short
```

Visual C++:

```
extern "C" {  
short __stdcall mtDestroyBarcodeInstance(HANDLE hBarcode) ;  
}
```

See also:

[CreateBarcodeInstance](#)

19.5 ExportXMLSettings

Synax

.net/com/java/ocx: Object.**ExportXMLSettings**(filePath)
dll: **mtExportXMLSettings**(hBarcode, filePath)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
filePath Path to the xml file to be created.

Return Value

1/True on success and 0/False on failure.

Remarks

ExportXMLSettings saves the current property values to an XML format file. The settings could be modified and loaded again using [LoadXMLSettings](#).

The default set of properties is as follows:

```
<xml version='1.0' encoding='iso-8859-1'>
  <SoftekBarcode>
    <Properties>
      <AllowDuplicateValues>1</AllowDuplicateValues>
      <BitmapResolution>200</BitmapResolution>
      <CodabarMaxVariance>20</CodabarMaxVariance>
      <Code25Checksum>0</Code25Checksum>
      <Code25MinOccurrenceLength>5</Code25MinOccurrenceLength>
      <Code39Checksum>0</Code39Checksum>
      <Code39NeedStartStop>1</Code39NeedStartStop>
      <ColorChunks>1</ColorChunks>
      <ColorProcessingLevel>2</ColorProcessingLevel>
      <ColorThreshold>0</ColorThreshold>
      <ConvertUPCEToEAN13>1</ConvertUPCEToEAN13>
      <Despeckle>0</Despeckle>
      <Encoding>0</Encoding>
      <ErrorCorrection>0</ErrorCorrection>
      <ExtendedCode39>0</ExtendedCode39>
      <GammaCorrection>100</GammaCorrection>
      <LineJump>1</LineJump>
      <MaxLength>999</MaxLength>
      <MedianFilter>0</MedianFilter>
      <MinSeparation>180</MinSeparation>
      <MinLength>4</MinLength>
```

```

<MinOccurrence>2</MinOccurrence>
<MinSpaceBarWidth>0</MinSpaceBarWidth>
<MultipleRead>0</MultipleRead>
<NoiseReduction>0</NoiseReduction>
<PageNo>0</PageNo>
<PatchCodeMinOccurrence>30</PatchCodeMinOccurrence>
<Pattern></Pattern>
<Pdf417Debug>0</Pdf417Debug>
<PdfBpp>8</PdfBpp>
<PdfDpi>300</PdfDpi>
<Photometric>0</Photometric>
<PrefOccurrence>5</PrefOccurrence>
<QuietZoneSize>0</QuietZoneSize>
<ReadCodabar>1</ReadCodabar>
<ReadCode128>1</ReadCode128>
<ReadCode25ni>0</ReadCode25ni>
<ReadCode25>1</ReadCode25>
<ReadCode39>1</ReadCode39>
<ReadCode93>0</ReadCode93>
<ReadDatabar>0</ReadDatabar>
<ReadDataMatrix>0</ReadDataMatrix>
<ReadEAN8>1</ReadEAN8>
<ReadEAN13>1</ReadEAN13>
<ReadMicroPDF417>0</ReadMicroPDF417>
<ReadNumeric>0</ReadNumeric>
<ReadPatchCodes>0</ReadPatchCodes>
<ReadPDF417>0</ReadPDF417>
<ReadQRCode>0</ReadQRCode>
<ReadShortCode128>0</ReadShortCode128>
<ReadUPCA>0</ReadUPCA>
<ReadUPCE>1</ReadUPCE>
<ScanDirection>15</ScanDirection>
<ShortCode128MinLength>2</ShortCode128MinLength>
<SkewLineJump>9</SkewLineJump>
<SkewTolerance>0</SkewTolerance>
<ShowCheckDigit>0</ShowCheckDigit>
<ShowCodabarStartStop>1</ShowCodabarStartStop>
<SkewSpeed>3</SkewSpeed>
<UseOldCode128Algorithm>0</UseOldCode128Algorithm>
<UseRunCache>1</UseRunCache>
<UseOverSampling>0</UseOverSampling>
<WeightLongerBarcodes>1</WeightLongerBarcodes>
</Properties>
</SoftekBarcode>
</xml>

```

Win32 dll declarations:**VB.Net:**

```
Private Declare Function mtExportXMLSettings Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr, ByVal strFile As String) As Short
```

Visual C++:

```
extern "C" {  
int __stdcall stExportXMLSettings(LPCSTR filePath);  
}
```

19.6 GetBarString

Syntax

.net/java/ocx: Object.**GetBarString**(n)
com: Object.**BarString**(n)
dll: **mtGetBarString** (hBarcode, n)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
n 1-based index to barcode to be queried.

Return Value

GetBarString retrieves a barcode string that has been detected by the [ScanBarCode](#) method. In .net applications, use System.Runtime.InteropServices.Marshal.PtrToStringAnsi to convert the IntPtr returned by mtGetBarString to a String.

Remarks

- Check digit characters are only output if the [ShowCheckDigit](#) property is set to True.
- UPC-E barcodes are a zero-suppressed version of EAN-13 and as such are converted to EAN-13 format unless the [ConvertUPCEToEAN13](#) property is set to False.
- Code 39 barcodes are not returned with the start/stop * characters.
- Codabar barcode value are always returned with the start/stop character pair, which can be either a/t, b/n, c/* or d/n.
- The following strings can be returned for patch codes - "Type I", "Type II", "Type III", "Type IV", "Type VI" and "Type T".

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtGetBarString Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,
ByVal nBarCode As Short) As System.IntPtr
```

Visual C++:

```
extern "C" {
LPCSTR __stdcall mtGetBarString(HANDLE hBarcode, short index);
}
```

19.7 GetBarStringPos, GetBarStringRect, BarStringPage, BarStringTopLeftX etc

Syntax

.net:

System.Drawing.Rectangle Object.**GetBarStringRect**(n)
Object.**GetBarStringPage**(n)

com:

Object.**BarStringPage**(n)
Object.**BarStringTopLeftX**(n)
Object.**BarStringTopLeftY**(n)
Object.**BarStringBottomRightX**(n)
Object.**BarStringBottomRightY**(n)

ocx:

pageNo = Object.**GetBarStringPos**(n, &TopLeftX &TopLeftY, &BottomRightX, &BottomRightY)

dll:

pageNo = **mtGetBarStringPos**(hBarcode, n, &TopLeftX, &TopLeftY, &BottomRightX, &BottomRightY)

Parameters

hBarcode	Handle to an instance of the barcode toolkit (dll only).
n	1-based index to barcode to be queried.
TopLeftX	variable to receive x coordinate of the top left hand corner of rectangle
TopLeftY	variable to receive y coordinate of the top left hand corner of rectangle
BottomRightX	variable to receive x coordinate of the bottom right hand corner of rectangle
BottomRightY	variable to receive y coordinate of the bottom right hand corner of rectangle

Return value

GetBarStringRect returns a System.Drawing.Rectangle object.

GetBarStringPage, GetBarStringPos, mtGetBarStringPos and BarStringPage return the page number (indexed from 1) for the barcode.

BarStringTopLeftX etc return the coordinates of the bounding rectangle.

Remarks

GetBarStringPos and the related group of functions are used to obtain the page number and bounding rectangle of a barcode. The methods available and the way in which they are called differ from interface to interface. Note that the bounding rectangle only covers the readable area of the barcode and if MultipleRead is set to false it will only cover a portion of the barcode.

In the case of the dll and ocx interfaces, the GetBarStringPos function returns the page number.

Win32 dll declarations:**VB.Net:**

```
Private Declare Function mtGetBarStringPos Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,
ByVal nBarCode As Short, ByRef TopLeftX As Integer, ByRef TopLeftY As Integer, ByRef BottomRightX
As Integer, ByRef BottomRightY As Integer) As Short
```

Visual C++:

```
short __stdcall mtGetBarStringPos(HANDLE hBarcode, short nBarCode, long FAR* pTopLeftX, long
FAR* pTopLeftY, long FAR* pBotRightX, long FAR* pBotRightY);
```

19.8 GetBarStringType

Syntax

.net/java/ocx: Object.**GetBarStringType**(n)
com: Object.**BarStringType**(n)
dll: **mtGetBarStringType**(hBarcode, n)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
n 1-based index to barcode to be queried.

Return Value

GetBarStringType returns a string representing the type of barcode.

In .net applications, use System.Runtime.InteropServices.Marshal.PtrToStringAnsi to convert the IntPtr returned by mtGetBarString to a String.

Remarks

Call GetBarStringType after calling ScanBarCode to determine the types of barcode found in a document. Examples of values returned by the function are "CODE39" and "CODE128".

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtGetBarString Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,  
ByVal nBarCode As Short) As System.IntPtr
```

Visual C++:

```
extern "C" {  
LPCSTR __stdcall mtGetBarString(HANDLE hBarcode, short index);  
}
```

19.9 GetBarStringDirection

Syntax

.net/java/ocx: Object.**GetBarStringDirection**(n)
com: Object.**BarStringDirection**(n)
dll: **mtGetBarStringDirection**(hBarcode, n)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
n 1-based index to barcode to be queried.

Return Value

1 = Left to Right

2 = Top to Bottom

4 = Right to Left

8 = Bottom to Top

16 = Top Left to Bottom Right

32 = Top Right to Bottom Left

64 = Bottom Right to Top Left

128 = Bottom Left to Top Right

Remarks:

Call GetBarStringDirection after calling ScanBarCode to determine the orientations of barcode found in a document. Note that this function will always return 1 for a QR-Code.

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtGetBarStringDirection Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr, ByVal nBarCode As Short) As Short
```

Visual C++:

```
extern "C" {  
short __stdcall mtGetBarStringDirection(HANDLE hBarcode, short nBarCode) ;  
}
```

19.10 GetLastError

Syntax

.net/java/ocx: **Object.GetLastError()**
com: **Object.LastError()**
dll: **mtGetLastError()**

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).

Return Value

Please refer to the following table for an explanation of the possible error numbers:

1000	Failed to load SoftekBarcodePDF.dll
1001	Failed to find a function in SoftekBarcodePDF.dll
1002	Failed to get the path for the temp file folder
1003	Failed to generate a temporary name
1004	Failed to generate a temp file name
1005	Failed to load SoftekBarcodePDF.dll
1006	Failed to find a function in SoftekBarcodePDF.dll
1007	Failed to load SoftekBarcodePDF.dll
1008	Failed to find a function in SoftekBarcodePDF.dll
1009	Failed to open a BMP file
1010	Failed to read BMP file header
1011	Failed to read BMP file info
1012	Memory allocation problem when loading a bitmap
1013	Memory allocation problem when loading a bitmap
1014	Error creating bitmap handle
1015	Failed to create handle for URL
1016	Failed to open requested URL
1017	Failed to open temporary file for contents of URL
1018	File open failed.
1019	Null dib handle supplied to mtScanBarCodeFromDIB
1020	Failed to find a function in SoftekBarcodePDF.dll
1021	Failed to load SoftekBarcodePDF.dll
2000	Failed to load pdf2image.dll
2001	PDF conversion returned zero or negative page count
2002	Failed to load pdf2image.dll
2003	Failed to load a function in pdf2image.dll
2004	PDF conversion returned null handle
2005	Failed to load pdf2tif.dll
2006	Failed to find a function in pdf2image.dll
2007	Failed to find a function in pdf2image.dll
2008	Requested page number is out of range
2009	Requested page number is out of range
2010	PDF conversion failed
2011	PDF conversion failed
2012	Failed to find a function in pdf2image.dll
2013	Failed to find a function in pdf2image.dll
2014	Failed to load file as a PDF document - probably a format error
2015	Failed to generate full path name
2016	No access to document
3000	Failed to connect to COM server PDF2ImageCOM.exe
3001	Failed to connect to COM server PDF2ImageCOM.exe
3002	Get page count function failed
3003	Get page box function failed

```
3004 Internal error in PDFToImageDrawToHDC
4000 Failed to connect to COM server PDF2TIFCOM.exe
4001 Failed to connect to COM server PDF2TIFCOM.exe
```

Remarks:

Call GetLastError to retrieve the last internal error number for the toolkit .

Win32 dll declarations:**VB.Net:**

```
Private Declare Function mtGetLastError Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr) As Short
```

Visual C++:

```
extern "C" {
short __stdcall mtGetLastError(HANDLE hBarcode) ;
}
```

19.11 GetLastWinError

Syntax

.net/java/ocx: Object.**GetLastWinError()**
com: Object.**LastWinError()**
dll: **mtGetLastWinError()**

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).

Return Value

The last windows error number, as recorded at the time of the last toolkit error.

Remarks:

Call GetLastWinError to retrieve the windows error number.

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtGetLastWinError Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr)
As Short
```

Visual C++:

```
extern "C" {
short __stdcall mtGetLastWinError(HANDLE hBarcode) ;
}
```

19.12 LoadXMLSettings

Syntax

Loading from xml file:

```
.net/java:    Object.LoadXMLSettings(file)
com/ocx:     Object.LoadXMLSettings(file, silent)
dll:        mtLoadXMLSettings(hbarcode, file, silent)
```

Loading from string:

```
.net/java:    Object.LoadXMLSettings(string)
com/ocx:     Object.LoadXMLSettings(string, silent)
dll:        mtLoadXMLSettings(hbarcode, string, silent)
```

Parameters

hBarcode	Handle to an instance of the barcode toolkit (dll only).
file	Path to the xml file to be loaded.
string	String holding the xml property values.
silent	Integer or boolean value. 1/True will suppress error messages.

Return Value

1/True on success and 0/False on failure.

Remarks

Load settings from the specified XML file or string. This method can also be used to define sets of properties to be applied sequentially to an image until a barcode is located. It's also possible to specify properties that only apply to particular pages of an image. The silent parameter controls whether message message boxes will be displayed if there is a parsing error.

Note that when loading from a string the XML data can be in the abbreviated form:

```
"<PropertyName>Value</PropertyName>"
```

For the com object, use the XMLRetval property to obtain the return value for the function.

The simplest format for the xml is as follows:

```
<xml>
  <SoftekBarcode>
    <Properties>
      <PropertyName>PropertyValue</PropertyName>
    </Properties>
  </SoftekBarcode>
</xml>
```

Example:

Set MedianFilter to 1 (True) and set ReadCode25 to 0 (False).

```
<xml>
  <SoftekBarcode>
    <Properties>
      <MedianFilter>1</MedianFilter>
      <ReadCode25>0</ReadCode25>
    </Properties>
  </SoftekBarcode>
</xml>
```

The xml file can also be used to apply sets of properties sequentially to an image until a barcode is found:

This example first applies default values to an image. If no barcode is found then a MedianFilter is applied.

```
<xml>
  <SoftekBarcode>
    <Properties>
  </Properties>
    <Properties>
      <MedianFilter>1</MedianFilter>
    </Properties>
  </SoftekBarcode>
</xml>
```

This example restricts the search to the first 3 pages of an image, until a barcode is located:

```
<xml>
  <SoftekBarcode>
    <Properties>
      <PageNo>1</PageNo>
    </Properties>
    <Properties>
      <PageNo>2</PageNo>
    </Properties>
    <Properties>
      <PageNo>3</PageNo>
    </Properties>
  </SoftekBarcode>
</xml>
```

The default set of properties can be found in the manual page for [ExportXMLSettings](#).

Win32 dll declarations:

VB.Net:

Private Declare Function mtLoadXMLSettings Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr, ByVal strFile As String, ByVal silent As Boolean) As Short

Visual C++:

```
extern "C" {  
int __stdcall mtLoadXMLSettings(HANDLE hBarcode, LPCSTR filePath, unsigned char silent) ;  
}
```

See Also

[ExportXMLSettings](#)

[ProcessXML](#)

19.13 ProcessXML

Syntax

.net: Integer Object. **ProcessXML** (inputFile, outputFile)
com: Object. **ProcessXML** (inputFile, outputFile, silent)
ocx: BOOL Object. **ProcessXML** (inputFile, outputFile, silent)
dll: BOOL **mtProcessXML** (hbarcode, inputFile, outputFile, silent)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
inputFile Path to the xml file to be processed.
ouputFile Path to the xml or csv file to be created.
silent Integer or boolean value. 1/True will suppress error messages.

Return Value

1/True on success and 0/False on failure. For the COM object, use the XMLRetval property to obtain the return value for the function.

Remarks

ProcessXML takes property values, file names and folder names from the inputFile and creates outputFile in either XML or CSV file format.

The property values defined in the XML file follow the specification as defined in the manual page for [LoadXMLSettings](#). The files and folder to process are defined in the following way....

To process a file:

```
<Process>  
  <File>/path/to/image.tif</File>  
</Process>
```

To process the images in a folder:

```
<Process>  
  <Folder>/path/to/folder</Folder>  
</Process>
```

Both the File and Folder tags accept an optional id attribute. This can be used in the output to identify the image file or folder.

Example:

```
<xml>  
  <SoftekBarcode>  
    <Properties>  
  </Properties>  
  <Properties>
```

```
<MedianFilter>1</MedianFilter>
</Properties>
<Process>
  <File id=1001>C:\tmp\image1.tif</File>
  <File id=1002>C:\tmp\image2.tif</File>
  <File id=1003>C:\tmp\image3.tif</File>
  <Folder id=9001>C:\tmp\images</Folder>
</Process>
</SoftekBarcode>
</xml>
```

See the manual page for [SaveResults](#) for the output file format.

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtProcessXML Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,
ByVal strInputFile As String, ByVal strOutputFile As String) As Short
```

Visual C++:

```
extern "C" {
int __stdcall mtProcessXML(HANDLE hBarcode, LPCSTR inputFilePath, LPCSTR outputFilePath,
unsigned char silent);
}
```

19.14 SaveResults

Overview

.net/ocx: Object.**SaveResults** (file)
com: Object. **SaveXMLResults** (file)
dll: **mtSaveResults** (hbarcode, file)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
file Path to the xml or csv file to be created.

Return Value

1/True on success and 0/False on failure.

Remarks

SaveResults saves the current set of results to an XML or CSV file.

The outputFile format is determined by the file extension (csv or xml).

The CSV fields are as follows:

Folder - location of image file

File Name

Count - number of barcodes found in image

Index - Index for this bar code (1,2,3 etc or -1 is no barcode found).

Error - Error number

ID - as specified in an xml input file for image or folder (see [ProcessXML](#))

Value - value of barcode

Type - type of barcode

Hits - hit count for this barcode

Page - page number in image

Direction - ScanDirection mask value for this barcode

TopLeftX - x position for top left

TopLeftY - y position for top left

BottomRightX - x position for bottom right

BottomRightY - y position for bottom right

The XML output format contains the same information as the CSV format, but arranged in the following way:

```
<xml version='1.0' encoding='iso-8859-1'>
  <SoftekBarcode>
    <Result>
      <Folder>location of image file</Folder>
      <FileName>name of file</FileName>
      <Count>number of barcodes</Count>
```

```

<Error>error number</Error>
<ID>id as specified in an xml input file for image or folder (see ProcessXML)</ID>
Repeated for each barcode found in image
<Barcode>
  <Value>value of barcode</Value>
  <Type>type of barcode</Type>
  <Hits>score</Hits>
  <Page>page number</Page>
  <Direction>ScanDirection mask value for this barcode</Direction>
  <TopLeftX>x position for top left</TopLeftX>
  <TopLeftY>y position for top left</TopLeftY>
  <BottomRightX>x position for bottom right</BottomRightX>
  <BottomRightY>y position for bottom right</BottomRightY>
</Barcode>
</Result>
</SoftekBarcode>
</xml>

```

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtSaveResults Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,
ByVal strOutputFile As String) As Short
```

Visual C++:

```
extern "C" {
int __stdcall mtSaveResults(HANDLE hBarcode, LPCSTR filePath);
}
```

19.15 ScanBarCode

Syntax

.net/com/java/ocx: Object.ScanBarCode(file)
dll: mtScanBarCode(hBarcode, file)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
file Path to the file containing the image to be scanned for barcodes.

Return Value

-1 Error opening file
-2 BMP file is multi-plane
-3 Invalid number of bits per sample
-4 Memory allocation error
-5 Invalid tif photometric property
-6,-7,-8 Invalid license key.

Remarks

Scan the specified [image file](#) for bar code strings and return the number of bar codes found in the file. Note that the function will stop when the first barcode is found in a document unless the [MultipleRead](#) property is set to True.

Further information for errors can be found by calling the [GetLastError](#) and [GetLastWinError](#) functions.

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtScanBarCode Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,
ByVal strFile As String) As Short
```

Visual C++:

```
extern "C" {
short __stdcall mtScanBarCode(HANDLE hBarcode, LPCTSTR file);
}
```

19.16 ScanBarcodeFromBitmap

Syntax

.net: Object.ScanBarcodeFromBitmap(bitmap)
.net/com/ocx: Object.ScanBarcodeFromBitmap(hBitmap)
dll: mtScanBarcodeFromBitmap(hBarcode, hBitmap)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
bitmap System.Drawing.Bitmap object
hBitmap handle (HBITMAP) to a bitmap.

Return Value

-1 N/A
-2 Bitmap is multi-plane
-3 Invalid number of bits per sample
-4 Memory allocation error
-5 N/A
-6,-7,-8 Invalid license key.

Remarks

Scan the specified device dependent bitmap for bar code strings and return the number of bar codes found. The image must be single plane. Note that hBitmap is a HANDLE to a BITMAP, not the address of a BITMAP structure. The Windows GDI function, CreateBitmapIndirect can be used to create an HBITMAP from a BITMAP structure. Note that the function will stop when the first barcode is found in a document unless the [MultipleRead](#) property is set to True.

The managed component supports 2 forms of this function:

ScanBarcodeFromBitmap(IntPtr hBitmap)

ScanBarcodeFromBitmap(System.Drawing.Bitmap bitmap) (Managed Component Only)

Scan the specified managed bitmap object for bar code strings and return the number of bar codes found.

See also: [ScanBarcodeFromDIB](#)

[Photometric](#)

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtScanBarcodeFromBitmap Lib "SofttekBarcode" (ByVal hBarcode As System.IntPtr, ByVal hBitmap As IntPtr) As Short
```

Visual C++:

```
extern "C" {  
short __stdcall mtScanBarCodeFromBitmap(HANDLE hBarcode, long hBitmap);  
}
```

19.17 ScanBarCodeFromDIB

Syntax

.net/com/ocx: Object.ScanBarCodeFromDIB(hDIB)
dll: mtScanBarCodeFromDIB(hBarcode, hDIB)

Parameters

hBarcode Handle to an instance of the barcode toolkit (dll only).
hDIB Memory object – see Remarks below.

Return Value

-1 N/A
-2 DIB is multi-plane
-3 Invalid number of bits per sample
-4 Memory allocation error
-5 N/A
-6,-7,-8 Invalid license key.

Remarks

Scan the specified device independent bitmap for bar code strings and return the number of bar codes found. The image must be single plane. The hDIB parameter is a handle to a memory object which has the same format as a BMP file not including the BITMAPFILEHEADER. Note that the function will stop when the first barcode is found in a document unless the [MultipleRead](#) property is set to True.

See also [ScanBarCodeFromBitmap](#)

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtScanBarCodeFromDIB Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr, ByVal hDIB As IntPtr) As Short
```

Visual C++:

```
extern "C" {  
short __stdcall mtScanBarCodeFromDIB(HANDLE hBarcode, long hDIB);  
}
```

19.18 SetScanRect

Syntax

.net/com/java/ocx: SetScanRect(tlx, tly, brx, bry, mode)
dll: mtSetScanRect(hBarcode, tlx, tly, brx, bry, mode)

Parameters

hBarcode	Handle to an instance of the barcode toolkit (dll only).
tlx	x coordinate of top left hand corner
tly	y coordinate of top left hand corner
brx	x coordinate of bottom right hand corner
bry	y coordinate of bottom right hand corner
mode	mapping mode (see below)

Return Value

1/True on success and 0/False on failure.

Remarks

SetScanRect specifies the bounding rectangle in the image that should be searched for barcodes. To clear the rectangle and search the entire image set the rectangle to (-1, -1, -1, -1). The top left hand corner of an image is (0,0).

The mapping mode can have the following values:

0 = All measurements are in pixels.

1 = All measurements are a percentage of the width or height of the image.

Win32 dll declarations:

VB.Net:

```
Private Declare Function mtSetScanRect Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr,
ByVal TopLeftX As Integer, ByVal TopLeftY As Integer, ByVal BottomRightX As Integer, ByVal
BottomRightY As Integer, ByVal MappingMode As Short) As Boolean
```

Visual C++:

```
extern "C" {
BOOL __stdcall mtSetScanRect(HANDLE hBarcode, long TopLeftX, long TopLeftY, long BottomRightX,
long BottomRightY, short MappingMode);
}
```

20 Appendix B: Properties Reference

(*) indicates an advanced parameter than can only be set using [LoadXMLSettings](#)

AllowDuplicateValues	allow duplicate barcode values on the same page
BitmapResolution	set the resolution of a bitmap
CodabarMaxVariance (*)	max width variance for codabar characters
Code25Checksum	handle final character of code 25 barcode as checksum
Code39Checksum	handle final character of code 39 barcode as checksum
Code39NeedStartStop	expect start/stop characters with code 39 barcodes
ColorChunks (*)	divide scan lines into sections for threshold levels
ColorProcessingLevel	control the time spent processing a color image
ColorThreshold	set the color threshold level for a color image
ConvertUPCEToEAN13	automatically convert UPC-E format to EAN-13
DatabarOptions	set options for GS1 Databar barcodes
Despeckle	remove speckled marks from an image before scanning
Encoding	set the encoding method for barcode values
ErrorCorrection	attempt to correct errors
ExtendedCode39	assume extended code 39 barcode format
FilePathEncoding	specify how file paths are encoded
GammaCorrection	set gamma correction level for color images
LicenseKey	set the license key
LineJump	control the frequency of line sampling
MaxLength	set the maximum length for a barcode
MedianFilter	perform a median filter on the image before scanning
MinLength	set the minimum length for a barcode
MinOccurrence	specify the lowest permitted score for a barcode
MinSeparation	minimum distance between barcodes of same value
MinSpaceBarWidth	minimum size of a space between bars

<u>MultipleRead</u>	scan for more than one barcode
<u>NoiseReduction</u>	perform noise reduction before scanning
<u>PageNo</u>	set the page number to scan in a multi-page image
<u>PatchCodeMinOccurrence</u>	minimum score for a Patch Code barcode.
<u>Pattern</u>	only report barcodes that fit the specified pattern
<u>Pdf417Debug</u> (*)	enable debug mode for PDF-417 barcodes
<u>PdfBpp</u>	load pdf documents at specified bits-per-pixel
<u>PdfDpi</u>	load pdf documents at specified dots-per-inch
<u>PdfImageExtractOptions</u>	mask to control how images are extracted from pdf files
<u>PdfImageOnly</u>	PDF documents will only contain images
<u>PdfImageRasterOptions</u>	mask to control how pdf files are rasterized
<u>Photometric</u>	set photometric interpretation for bi-tonal bitmaps
<u>PrefOccurrence</u>	specify the preferred score for a barcode
<u>QuietZoneSize</u>	set the size of the quiet zone around a barcode
<u>ReadCodabar</u>	scan for codabar barcodes
<u>ReadCode128</u>	scan for code-128 barcodes
<u>ReadCode25</u>	scan for code-25 barcodes
<u>ReadCode25ni</u>	scan for non-interleaved code-25 barcodes
<u>ReadCode39</u>	scan for code-39 barcodes
<u>ReadCode93</u>	scan for code-93 barcodes
<u>ReadDatabar</u>	scan for databar barcodes
<u>ReadDataMatrix</u>	scan for datamatrix barcodes
<u>ReadEAN13</u>	scan for ean-13 barcodes
<u>ReadEAN8</u>	scan for ean-8 barcodes
<u>ReadMicroPDF417</u>	scan for micro pdf-417 barcodes
<u>ReadNumeric</u>	only read numeric barcodes
<u>ReadPatchCodes</u>	scan for patch codes

ReadPDF417	scan for pdf-417 barcodes
ReadQRCode	scan for QR-codes.
ReadShortCode128	scan for short code-128 barcodes
ReadUPCA	scan for upc-a barcodes
ReadUPCE	scan for upc-e barcodes
ScanDirection	specify the orientations in which to scan
ShortCode128MinLength	set the minimum length for a short code-128 barcode
ShowCodabarStartStop (*)	include codabar start/stop characters
ShowCheckDigit	display check digits where possible
SkewLineJump	frequency of line sampling for skewed barcodes
SkewTolerance	maximum tolerance for skewed barcodes
TifSplitMode	control the way that tif and pdf documents are split
TifSplitPath	specify the output path for splitting tif and pdf documents
UseOldCode128Algorithm (*)	use the old code-128 detection method
UseOverSampling	process multiple scan lines at the same time
UseRunCache (*)	use a memory cache for run-length information
WeightLongerBarcodes	accept lower scores for longer barcodes

(*) indicates an advanced parameter than can only be set using [LoadXMLSettings](#).

20.1 Setting and Getting Property Values

All properties, except those marked as “Advanced” can be set and read using the following methods. All properties (including those marked as “Advanced”) can also be set using [LoadXMLSettings](#).

Setting a property value using the .net/com or java interfaces:

```
Object.PropertyName = Value
```

Getting a property value using the .net/com or java interfaces:

```
Value = Object.PropertyName
```

Setting a property value using the ocx interface:

```
Object.SetPropertyName(Value)
```

Getting a property value using the ocx interface:

```
Value = Object.GetPropertyName()
```

Setting a property value using the dll interface:

```
mtSetPropertyName(hBarcode, Value)
```

Getting a property value using the ocx interface:

```
Value = mtGetPropertyName(hBarcode)
```

Win32 dll declarations:

If the property is of type BOOL:

VB.Net:

```
Private Declare Function mtGetPropertyName Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr) As Boolean
Private Declare Function mtSetPropertyName Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr , ByVal newValue As Boolean) As Boolean
```

Visual C++:

```
extern "C" {
BOOL __stdcall mtGetPropertyName (HANDLE hBarcode);
BOOL __stdcall mtSetPropertyName (HANDLE hBarcode , BOOL bNewValue);
}
```

If the property is of type SHORT:

VB.Net:

```
Private Declare Function mtGetPropertyNames Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr) As Short
Private Declare Function mtSetPropertyName Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr, ByVal newValue As Short) As Short
```

Visual C++:

```
extern "C" {
short __stdcall mtGetPropertyNames (HANDLE hBarcode);
short __stdcall mtSetPropertyName (HANDLE hBarcode, short bNewValue);
}
```

If the property is of type STRING:

VB.Net:

```
Private Declare Function mtGetPropertyNames Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr) As String
Private Declare Function mtSetPropertyName Lib "SoftekBarcode" (ByVal hBarcode As System.IntPtr, ByVal newValue As String) As String
```

Visual C++:

```
extern "C" {
LPCSTR __stdcall mtGetPropertyNames (HANDLE hBarcode);
LPCSTR __stdcall mtSetPropertyName (HANDLE hBarcode, LPCSTR bNewValue);
}
```

20.2 AllowDuplicateValues

Overview

The AllowDuplicateValues can be used to stop the toolkit from reporting duplicate barcodes on the same page in an image. This can be useful for images where the middle section of a barcode is badly damaged or missing. With the property set to TRUE the toolkit may report that there are 2 barcodes of the same type and value. With the property set to FALSE it would assume that the 2 barcodes were part of a single barcode and set the bounding rectangle accordingly.

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

[MinSeparation](#)

20.3 BitmapResolution

Overview

BitmapResolution is the resolution of the bitmap to be scanned in [ScanBarCodeFromBitmap](#), in dots per inch. This value only effects the expected size of the quiet area around a barcode and for most images can be left to the default value.

Type: SHORT
Default value: 200

See also: [Setting and Getting Property Values](#)

20.4 CodabarMaxVariance

Overview

CodabarMaxVariance is the maximum percentage variance that a character in a codabar barcode can have from the average for that barcode.

Type: SHORT
Default value: 20

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

See also: [ReadCodabar](#)

20.5 Code25Checksum

Overview

When True the toolkit will only report Code 25 barcodes where the last character is a valid checksum for the rest of the barcode. The toolkit expects a Code 25 checksum to be calculated using the following method:

Sum all of the even positioned characters (the right hand message character is always even), and multiply by 3.

Sum all the odd positioned characters.

Sum the totals from steps 1 and 2.

The checksum is the smallest number that when added to this sum results in a multiple of 10.

If the resulting number of characters is odd and you are using Interleaved Code 2 of 5 then add a leading 0 to the message data.

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.6 Code39Checksum

Overview

When True the toolkit will only report Code 39 barcodes where the last character is a valid checksum for the rest of the barcode. The toolkit expects a Code 39 checksum to be calculated using modulus-43.

The following table shows the character and value used for the calculation...

Char	Value	Char	Value	Char	Value	Char	Value
0	0	B	11	M	22	X	33
1	1	C	12	N	23	Y	34
2	2	D	13	O	24	Z	35
3	3	E	14	P	25	-	36
4	4	F	15	Q	26	.	37
5	5	G	16	R	27	space	38
6	6	H	17	S	28	\$	39
7	7	I	18	T	29	/	40
8	8	J	19	U	30	+	41
9	9	K	20	V	31	%	42
A	10	L	21	W	32		

e.g

Data = 12345ABCDE+

Sum of values: $1 + 2 + 3 + 4 + 5 + 10 + 11 + 12 + 13 + 14 + 41 = 116$

$116 / 43 = 2 \text{ rem } 30$, so U is the check digit.

Data and check digit = 12345ABCDE+U

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

[ReadCode39](#)

[Code39NeedStartStop](#)

[ExtendedCode39](#)

20.7 Code39NeedStartStop

Overview

When set to TRUE the toolkit will only report Code 39 barcodes that start and end with a * character.

Setting this property to FALSE is not recommended for the following reasons:

It is not a valid Code 39 barcode without the start and stop * character.

Without a start/stop * character, a Code 39 barcode reads with 2 different values, left to right, and right to left. The toolkit will report it as 2 different barcodes unless the scan direction is restricted to one direction only.

The probability of a false positive reading is increased significantly by setting this property to FALSE.

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

20.8 ColorChunks

Overview

ColorChunks specifies how many sections a scan line of an image should be broken into when calculating threshold levels for black and white pixels.

Type: SHORT

Default value: 1

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

See also: [ColorProcessingLevel](#)

20.9 ColorProcessingLevel

Overview

The `ColorProcessingLevel` property controls the amount of processing time spent reading barcode values from color images. Values range from 0 to 5, with a default of 2. A low value will process color images faster but accuracy and read-rate levels will be lower than if a high value is used.

Please note that setting the [ColorThreshold](#) property to a non-zero value effectively sets `ColorProcessingLevel` to 0.

Type: SHORT

Default value: 2

See also: [Setting and Getting Property Values](#)

[ColorChunks](#)

[ColorThreshold](#)

20.10 ColorThreshold

Overview

ColorThreshold is the color value used by the control to decide whether a pixel should be considered to be black or white. The value should be in the range 0 to 255.

Please note that if this property is set to a non-zero value than [ColorProcessingLevel](#) is effectively set to a value of 0. It is recommended to set this property to 0 and control the accuracy of reading from color images through the [ColorProcessingLevel](#) property.

Type: SHORT

Default value: 0

See also: [Setting and Getting Property Values](#)

[ColorProcessingLevel](#)

20.11 ConvertUPCEToEAN13

Overview

A UPC-E barcode is actually an EAN-13/UPC-A barcode that has had certain digits removed to create an 8 digit number. Only certain EAN-13/UPC-A barcodes can go through this process. For example, the UPC-A barcode "023456000073 " can be suppressed to the UPC-E value "02345673" and restored to it's original value by the barcode reader. The Softek barcode Reader SDK can interpret a UPC-E barcode in either format via the `ConvertUPCEToEAN13` property.

When set to TRUE the toolkit will convert type UPC-E barcodes into EAN-13 format.

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

[ReadUPCE](#)

[ReadEAN13](#)

[ReadUPCA](#)

20.12 DatabarOptions

Overview

The DatabarOptions property can be used to set various options for GS1 Databar recognition. All options are turned on by default, but some applications may find it useful to disable certain features for performance reasons.

The property works as a mask and can be constructed from the following values:

- 1 Read the supplementary 2-D portion if indicated by the linkage flag.
- 2 Read RSS-14 barcodes
- 4 Read RSS-14 Stacked barcodes
- 8 Read RSS-Limited barcodes
- 16 Read RSS-Expanded barcodes
- 32 Read RSS-Expanded Stacked barcodes

Type: SHORT

Default value: 255

See also: [Setting and Getting Property Values](#)

[ReadDatabar](#)

20.13 Despeckle

Overview

If the **Despeckle** property is set to TRUE and the [NoiseReduction](#) property is none zero, then the toolkit removes white speckles inside the bars of a barcode before removing black marks from the spaces between bars.

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.14 Encoding

Overview

The Encoding property controls the format in which the toolkit returns strings for barcode types that use full symbol sets such as PDF-417.

The property can take any of the following values:

- 0 Raw , with null characters suppressed.
- 1 Quoted printable
- 2 Unicode
- 3 UTF-8

Type: SHORT

Default Value: 0

See also: [Setting and Getting Property Values](#)

20.15 ErrorCorrection

Overview

Some barcodes cannot be read because the process of scanning or faxing has split or merged bars together. When ErrorCorrection is set to True to toolkit will, where possible, make a best guess at such barcodes.

Note that this property currently only applies to Code 39 and Code 39 Extended barcodes.

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

[ReadCode39](#)

20.16 ExtendedCode39

Overview

A Code 39 barcode can be used to represent the entire ASCII-128 symbol set by using 2 normal Code 39 characters to represent one character in the ASCII-128 symbol set. A barcode reader cannot distinguish between normal and extended Code 39 barcodes and so the ExtendedCode39 property

must be set to TRUE when reading barcodes encoded using the extended symbol set. Note that the [ReadCode39](#) property must also be set to TRUE.

If the toolkit is unable to decode the string in the extended symbol set then it is left as a normal Code 39 barcode.

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

 [ReadCode39](#)

20.17 FilePathEncoding

Overview

FilePathEncoding specifies how file paths are encoded when passed to the win32 DLL interface. A value of 0 indicates ASCII encoding and 1 indicates UTF-8 encoding. Wrapper interfaces such as Java and SoftekBarcodeLib3.dll set this property to 1 by default where as the default for the win32 DLL is 0.

Type: SHORT
Default value: (see above)

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

20.18 GammaCorrection

Overview

If GammaCorrection is set to a value other than 100 then the toolkit will apply gamma correction to a color image. The amount of gamma correction is equal to $\text{GammaCorrection} / 100$. For example, to achieve a gamma correction of 0.5 the property should be set to a value of 50.

Type: SHORT
Default value: 100

See also: [Setting and Getting Property Values](#)

20.19 LicenseKey

Overview

Use the LicenseKey property to set your license key prior to calling the [ScanBarCode](#), [ScanBarCodeFromBitmap](#) or [ScanBarCodeFromDIB](#) functions. With no license key the .net interface will return all barcode values as "Please contact sales@bardecode.com for a trial license string" and other interfaces will display a pop up box that the user will need to click on to continue.

Type: STRING

Default value: ""

See also: [Setting and Getting Property Values](#)

20.20 LineJump

Overview

The LineJump property controls the frequency with which the toolkit samples scan lines as it moves through an image. Increasing the value of the LineJump property will increase the speed at which an image is processed but may decrease the read rate. The [SkewLineJump](#) property is used in a similar way when searching for skewed barcodes.

Type: SHORT

Default value: 1

See also: [Setting and Getting Property Values](#)

[SkewLineJump](#)

20.21 MaxLength

Overview

MaxLength defines the largest length for a barcode string, including checksum characters.

Type: SHORT

Default value: 999

See also: [Setting and Getting Property Values](#)

[MinLength](#)

20.22 MedianFilter

Overview

When TRUE the toolkit will apply a median filter to the image before checking for barcodes. This is a useful option for high resolution images that contain speckles of black and white. It is not recommended for images where the black bars or white spaces are less than 2 pixels wide.

Type: BOOL

Default Value: FALSE

See also: [Setting and Getting Property Values](#)

20.23 MinLength

Overview

MinLength defines the smallest length for a barcode string, including checksum characters.

Type: SHORT

Default value: 4

See also: [Setting and Getting Property Values](#)

[MaxLength](#)

20.24 MinOccurrence

Overview

Please refer to [PrefOccurrence](#) for more information.

Type: SHORT

Default value: 2

See also: [Setting and Getting Property Values](#)

20.25 MinSeparation

Overview

MinSeparation defines the minimum distance between barcodes of identical value and vertical alignment in 1/300th of an inch. If the distance between two barcodes of same value and on the same alignment is less than MinSeparation then the toolkit assumes that it is a single barcode that has been split into 2 parts by a problem in the scanning process.

Type: SHORT

Default value: 180

See also: [Setting and Getting Property Values](#)

20.26 MinSpaceBarWidth

Overview

MinSpaceBarWidth is the minimum acceptable size for a space between the bars in a barcode. When set to a value of 0 the toolkit will automatically select the best value. Spaces that are smaller than the value used are ignored.

Type: SHORT

Default value: 0

See also: [Setting and Getting Property Values](#)

20.27 MultipleRead

Overview

Normally the toolkit stops at the first positive match for a barcode. When **MultipleRead** is TRUE the toolkit will check the entire image for barcode strings and record each positive match.

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.28 NoiseReduction

Overview

If the **NoiseReduction** property is none zero then the toolkit will run an image through a noise reduction filter before scanning for barcodes. The filter removes marks from an image that are unlikely to be part of a barcode. A larger value for NoiseReduction will remove larger marks from the image, but may also destroy vital barcode information. A typical value for **NoiseReduction** is 10.

Type: SHORT

Default value: 0

See also: [Setting and Getting Property Values](#)

[Despeckle](#)

20.29 PageNo

Overview

PageNo is a 1 based index that specifies the page to be scanned in an image. A value of zero indicates that every page will be scanned

Type: SHORT

Default value: 0

See also: [Setting and Getting Property Values](#)

20.30 PatchCodeMinOccurrence

Overview

Please refer to [PrefOccurrence](#) for more information.

Type: SHORT
Default value: 30

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

20.31 Pattern

Overview

The Pattern property is a regular expression that each barcode found in an image is compared against. The toolkit will only return barcodes that match the pattern.

The toolkit use POSIX extended regular expression syntax.

Examples:

"ABCDEF" will match all barcodes containing "ABCDEF" (e.g "XYZABCDEFXYZ").

"ABC[0-9]+"

"^ABC[0-9]+\$" will match barcodes that only consist of "ABC" followed by one or more digits (e.g "ABC12345").

Note that if a Code 39 barcode uses a checksum character and the Pattern property is used to specify the entire string (ie. the last character of the pattern is \$) then the Code39Checksum property must also be set to True.

Type: STRING
Default value: NULL

See also: [Setting and Getting Property Values](#)

[ReadNumeric](#)

20.32 Pdf417Debug

Overview

Output information about the structure and cluster values of the barcode. The debug information is returned in place of the normal barcode value (see GetBarString).

The fields of the string are as follows:

Error correction status – true or false

Number of data columns

Number of rows

Error correction level
Number of unknown cluster values
And for each cluster: cluster value (score)

Type: BOOL
Default value: FALSE

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

20.33 PdfBpp

Overview

PdfBpp is the number of bits-per-pixel to be used when converting a pdf file into image format, before scanning for a barcode. The value should be 1, 8 or 24. Has no effect when PdfImageOnly is set to True.

Type: SHORT
Default value: 8

See also: [Setting and Getting Property Values](#)

[PdfDpi](#)

[PdfImageOnly](#)

20.34 PdfDpi

Overview

PdfDpi is the number of dots-per-inch to be used when converting a pdf file into image format, before scanning for a barcode. Has no effect when PdfImageOnly is set to True.

Type: SHORT
Default value: 300

See also: [Setting and Getting Property Values](#)

[PdfBpp](#)

[PdfImageOnly](#)

20.35 PdfImageExtractOptions

Overview

PdfImageExtractOptions is a mask that controls the way in which images are extracted from PDF documents when PdfImageOnly is set to TRUE.

The values for the mask are as follows:

- 1 Enable fast extraction
- 2 Auto-invert black and white images
- 4 Auto-merge strips
- 8 Auto-correct photometric values in black and white images

PDF documents process at around twice the speed when fast extraction is enabled, but it is possible that some black-and-white images will extract with incorrect photometric values (a negative image, even when auto-invert is enabled); to compensate for this the SDK will use the dominant background color to determine the correct value when the “auto-correct photometric” flag is set. All of the flags are enabled by default.

Type: SHORT

Default value: 15

Note: This property is an **Advanced Property** and must be set using [LoadXMLSettings](#).

See also: [PdfImageOnly](#)

20.36 PdfImageOnly

Overview

PdfImageOnly indicates that the PDF documents to be processed by the toolkit are images. Image only PDF documents can be processed faster than other types (containing text, vector graphics etc). If the number of images extracted from a PDF document does not match the number of pages in the document then the document is re-processed as if PdfImageOnly was set to false.

Type: BOOL

Default value: True

Note: In some interfaces this property must be set as an **Advanced Property** using [LoadXMLSettings](#).

See also: [PdfImageExtractOptions](#)

[Setting and Getting Property Values](#)

20.37 PdfImageRasterOptions

Overview

PdfImageRasterOptions is a mask that controls the way in which PDF files are rasterized. Note that this option is only used when PdfImageOnly = false or when image extraction has failed.

The values for the mask are as follows:

- 1 Use alternative pdf-to-tif conversion function.
- 2 Always use pdf-to-tif conversion rather than loading the rasterized image directly into memory.

Type: SHORT
 Default value: 0

Note: This property is an **Advanced Property** and must be set using [LoadXMLSettings](#)

See also: [PdfImageOnly](#)

20.38 Photometric

Overview

The Photometric property determines how the toolkit interprets a pixel value in a bi-tonal bitmap passed to the [ScanBarcodeFromBitmap](#) method.

	Pixel Value = 0	Pixel Value = 1
Photometric = 0	Black	White
Photometric = 1	White	Black

This property is not used with the ScanBarcode or ScanBarcodeFromDIB methods.

Type: SHORT
 Default value: 0

See also: [Setting and Getting Property Values](#)
[ScanBarcodeFromBitmap](#)

20.39 PrefOccurrence

Overview

As the SDK scans an image it assigns a score to each barcode candidate. At the end of a scan, any candidates with a score \geq PrefOccurrence are reported by the SDK. If no candidate meets this criteria then the SDK selects the candidate with the highest score and reports this barcode if it has a score \geq [MinOccurrence](#). Note that Patch Codes are only ever reported if the score is \geq [PatchCodeMinoccurrence](#).

Type: SHORT
 Default value: 5

See also: [Setting and Getting Property Values](#)

[MinOccurrence](#)

[PatchCodeMinOccurrence](#)

20.40 QuietZoneSize

Overview

When the toolkit checks for a barcode on a scan line in an image, it ignores those parts of the line that are not preceded by the number of white pixels specified by QuietZoneSize. When the property has a value of 0 then the quiet zone is calculated $1/10^{\text{th}}$ of the value of the image resolution (e.g. 10 pixels in a 100 dpi image).

Type: SHORT

Default value: 0

See also: [Setting and Getting Property Values](#)

20.41 ReadCodabar

Overview

When set to TRUE the toolkit will search for codabar barcodes and the string returned by GetBarStringType will be set to "CODABAR".

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

[CodabarMaxVariance](#)

20.42 ReadCode128

Overview

When set to TRUE the toolkit will search for type 128 barcodes and the string returned by GetBarStringType will be set to CODE128.

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

[ReadShortCode128](#)

[UseOldCode128Algorithm](#)

20.43 ReadCode25

Overview

When set to TRUE the toolkit will search for type 2 of 5 interleaved barcodes and the string returned by GetBarStringType will be set to "CODE25".

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

[ReadCode25ni](#)

[Code25Checksum](#)

20.44 ReadCode25ni

Overview

When set to TRUE the toolkit will search for type 2 of 5 non-interleaved barcodes in the following formats:

- Code 2 of 5 Datalogic
- Code 2 of 5 Iata1
- Code 2 of 5 Iata2
- Code 2 of 5 Industrial
- Code 2 of 5 Interleaved
- Code 2 of 5 Matrix

The string returned by GetBarStringType will be set to "CODE25".

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.45 ReadCode39

Overview

When set to TRUE the toolkit will search for type 39 barcodes and the string returned by GetBarStringType will be set to "CODE39".

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

[Code39Checksum](#)

[Code39NeedStartStop](#)

[ExtendedCode39](#)

20.46 ReadCode93

Overview

When set to TRUE the toolkit will search for type 93 barcodes and the string returned by GetBarStringType will be set to "CODE93".

Type: BOOL

Default value: FALSE

Note: In some interfaces this property must be set as an **Advanced Property** using [LoadXMLSettings](#)

20.47 ReadDatabar

Overview

When set to TRUE the toolkit will search for GS1 Databar barcodes and the string returned by GetBarStringType will be set to "DATABAR". The following types of GS1 Databar are supported:

RSS-14

RSS-14 Truncated

RSS-14 Stacked

RSS-14 Stacked Omnidirectional

RSS Limited

RSS Expanded

RSS Expanded Stacked

Please note the the bounding rectangle for stacked versions of the barcode currently only includes either the top-most or bottom-most element of the stack.

Reading supplementary data

Some GS1 Databar barcodes encode supplementary data in the form of a micro-PDF-417 barcode above the linear portion of the barcode. To read the supplementary portion set

[ReadMicroPDF417](#) to True and ensure that [DatabarOptions](#) includes the option to read supplementary barcodes.

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

[ReadMicroPDF417](#)

[DatabarOptions](#)

20.48 ReadDataMatrix

Overview

When set to TRUE the toolkit will search for DataMatrix (ECC 200) barcodes and the string returned by GetBarStringType will be set to "DATAMATRIX".

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.49 ReadEAN13

Overview

When set to TRUE the toolkit will search for EAN-13 type barcodes and the string returned by GetBarStringType will be set to "EAN13".

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

20.50 ReadEAN8

Overview

When set to TRUE the toolkit will search for EAN-8 type barcodes and the string returned by GetBarStringType will be set to "EAN8".

Type: BOOL

Default value: TRUE

See also: [Setting and Getting Property Values](#)

20.51 ReadMicroPDF417

Overview

When set to TRUE the toolkit will search for micro-PDF-417 barcodes and the string returned by GetBarStringType will be set to "PDF417".

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.52 ReadNumeric

Overview

When True the toolkit will only report numeric barcodes. Note that this the same as setting the Pattern property to the value "[0-9]+\$".

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.53 ReadPatchCodes

Overview

When set to TRUE the toolkit will search for patch code barcodes and the string returned by GetBarString will be set to PATCH.

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.54 ReadPDF417

Overview

When set to TRUE the toolkit will search for PDF-417 barcodes and the string returned by GetBarString will be set to "PDF417".

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.55 ReadQRCode

Overview

When set to TRUE the toolkit will search for QR-Codes and the string returned by GetBarString will be set to "QRCODE".

- All version sizes are supported, however best results will always be obtained when using smaller version sizes with maximum error correction.
- The Kanji symbol set is not currently supported.
- Skewed QR-Codes can be read using the default value for [SkewTolerance](#). Changing the value of [SkewTolerance](#) will have no effect on the scanning of QR-Codes.
- For best results set [ColorProcessingLevel](#) to a value of 1.
- [GetBarStringDirection](#) will always return a value of 1 for QR-Codes.
- If only QR-Codes need to be recognized then set ScanDirection to a value of 1.

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.56 ReadShortCode128

Overview

When set to TRUE the toolkit will search for Code 128 barcodes of symbol set C, without the normal start and stop characters. The barcode type for these barcodes is set to "SHORTCODE128".

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.57 ReadUPCA

Overview

When set to TRUE the toolkit will search for UPC-A type barcodes.

UPC-A barcodes are a subset of EAN-13. For example, the UPC-A barcode "016000336100" is the same as the EAN-13 barcode "0016000336100". In fact, UPC-A barcodes are the sub-set of EAN-13 barcodes that start with a 0. The ReadEAN13 property controls whether any barcodes of type EAN-13 are recognized - and this includes UPC-A, whether or not ReadUPCA is set to true. The effect of the ReadUPCA flag is to control whether an EAN-13 barcode that starts with a 0 is returned as a 12 digit UPC-A or as a 13 digit EAN-13 barcode. The string returned by GetBarString will be set to either "UPCA" or "EAN13".

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.58 ReadUPCE

Overview

When set to TRUE the toolkit will search for UPC-E type barcodes.

A UPC-E barcode is actually an EAN-13/UPC-A barcode that has had certain digits removed to create an 8 digit number. Only certain EAN-13/UPC-A barcodes can go through this process.

For example, the UPC-A barcode "023456000073" can be suppressed to the UPC-E value "02345673" and restored to its original value by the barcode reader. The Softek barcode Reader SDK can interpret a UPC-E barcode in either format via the [ConvertUPCETOEAN13](#) property. The string returned by GetBarStringType will be set to "UPCE".

Type: BOOL
Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.59 ScanDirection

Overview

ScanDirection is a mask that controls the directions in which the barcode reader will look for barcodes in an image, and is built from the following values:

1 = Left to Right

2 = Top to Bottom

4 = Right To Left

8 = Bottom to Top

For example, a value of 5 (1 + 4) means that the reader will look for barcode from left to right and right to left.

Note: This property replaces the [Rotation](#) property used in previous versions.

Type: SHORT
Default value: 15

See also: [Setting and Getting Property Values](#)

20.60 ShortCode128MinLength

Overview

ShortCode128MinLength defines the smallest length for a barcode string, including checksum characters.

Type: SHORT

Default value: 2

See also: [Setting and Getting Property Values](#)

20.61 ShowCodabarStartStop

Overview

Include the start and stop characters when returning the value of a codabar barcode.

Type: BOOL

Default value: TRUE

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

20.62 ShowCheckDigit

Overview

When set to TRUE the OCX will include the barcode check digit in the returned string.

Note: This property only applies to barcode types with built in check digits (e.g Code 128).

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.63 SkewLineJump

Overview

SkewLineJump works in a similar way to the [LineJump](#) property, but only effects the phase of the scanning process concerned with searching for skewed barcodes. It can be useful to set the 2 properties to different values for reasons of performance.

Type: SHORT

Default value: 9

See also: [Setting and Getting Property Values](#)

20.64 SkewTolerance

Overview

SkewTolerance controls the maximum angle from the horizontal or vertical at which a barcode will be recognised by the toolkit. The table below shows the possible values for this property along with the approximate maximum angles:

0 = up to 5 degrees

1 = 13 degrees

2 = 21 degrees

3 = 29 degrees

4 = 37 degrees

5 = 45 degrees

Type: SHORT

Default value: 0

See also: [Setting and Getting Property Values](#)

20.65 TifSplitMode

Overview

TifSplitMode controls the way in which multi-page TIF files are split into sub-files when the TifSplitPath property is set. A value of 0 means that the sub-files will each start on a page with a barcode (or page 1). A value of 1 means that each sub-file will terminate on a page with a barcode (or the last page) and a value of 2 means that each sub-file consists of the pages between (but not including) the barcodes.

For example, a 6 page TIF file with barcodes on pages 2 and 5 will split as follows:

TifSplitMode = 0: First sub-file contains page 1. Second sub-file contains pages 2, 3 and 4. Third sub-file contains pages 5 and 6.

TifSplitMode = 1: First sub-file contains pages 1 and 2. Second sub-file contains pages 3, 4 and 5. Third sub-file contains page 6.

TifSplitMode = 2: First sub-file contains page 1. Second sub-file contains pages 3 and 4. Third sub-file contains page 6.

Type: SHORT
Default value: 0

See also: [Setting and Getting Property Values](#)

20.66 TifSplitPath

Overview

TifSplitPath is the file path template for splitting multi-page TIF files into sub-files in either TIF or PDF format. The template can include the following conversion specifications:

%d replaced by a sub-file index number (starting at 1)
%b replaced by the barcode found in the sub-file (modes 0 and 1) or the last barcode found before the sub-file (mode 2). If no relevant barcode is available then an empty string is used.

If the template does not include %d or %s, or %s then a %d is assumed before the file extension. This case also applies if %s is used on it's own and there is no relevant barcode value available.

NOTE: The .Net 1 interface cannot output files in PDF format and the .Net 2/3.5 interface requires the PDF Extension to output files in PDF format.

The mode for splitting the TIF file is controlled by the TifSplitMode property.

e.g c:\tmp\output%d_%s.tif

or c:\tmp\output%d_%s.pdf

Type: STRING
Default value: NULL

See also: [Setting and Getting Property Values](#)

20.67 UseOldCode128Algorithm

Overview

Use the Code 128 detection algorithm as used in earlier versions of the toolkit (pre version 7.3.1).

Type: BOOL
Default value: FALSE

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

20.68 UseOverSampling

Overview

When UseOverSampling is TRUE the barcode reader samples 3 lines at a time (skipping 2 lines between each sample) and takes the average pixel value. This is useful for images containing both black and white speckles.

Type: BOOL

Default value: FALSE

See also: [Setting and Getting Property Values](#)

20.69 UseRunCache

Overview

Use a memory cache for run-length information derived from an image.

Type: BOOL

Default value: TRUE

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

20.70 WeightLongerBarcodes

Overview

When WeightLongerBarcodes is TRUE the barcode reader will weight the counts used with the [PrefOccurrence](#) and [MinOccurrence](#) properties according to the length and type of the barcode in question. Barcode types using built in checksums are favoured above barcode types with no checksum.

Type: BOOL

Default value: TRUE

Note: This is an **Advanced property** and can only be set using [LoadXMLSettings](#)

21 Appendix C: Installation Files

This section shows the dll files required for the various interfaces to the toolkit. Note that one approach is to leave all the dll files in the folder created by the install set and either modify the system PATH or specify the location of the files in the application. The section on installation on [Production Systems](#) contains details of the options.

21.1 x86 Applications

The following table shows the files necessary to support x86 based applications. Note that all of these files can either be left in the original installation folder or put in one of the following target folders:

- Windows system32 folder on x86 based systems
- Windows SysWow64 folder on x64 based systems
- Any folder on the applications path (and on the reference path for .net applications). On x64 systems take care to ensure that x86 and x64 applications find the correct versions of the dll files in their paths.

File	Win32 DLL	.Net 2.0+ Lib2.dll	.Net 2.0+ Lib3.dll	.Net 1	Java	ActiveX/OCX	COM	Adobe PDF
x86/bardecode_jni.dll					✓			
x86/cimage.dll								✓
x86/convert.dll								✓
x86/encryptpdf.dll								✓
x86/pdf2image.dll								✓
x86/pdf2tif.dll								✓
x86/pdf2img.ini								✓
x86/SoftekATL.dll							🚩	
x86/SoftekBarcode.dll	✓	✓🚩	✓		✓	✓	✓	
x86/SoftekBarcode.ocx						🚩		
x86/SoftekBarcodeLib2.dll		✓🚩						
anycpu/SoftekBarcodeLib3.dll			✓					
x86/SoftekBarcodeLib.dll				🚩				
x86/SoftekBarcodePDF.dll								✓

Where Lib2.dll refers to SoftekBarcodeLib2.dll and Lib3.dll refers to SoftekBarcodeLib3.dll



Register with regsvr32.

Not compatible with the PDF Extension to the toolkit.

Requires install Microsoft Visual C++ 2008 Redistributable Package SP1 for x86 systems

Required only if documents are to be split into PDF format.

21.2 x64 Applications

The following table shows the files that need to be installed to support x64 based applications. One approach is to leave all the dll files in the original installation folder, but if this isn't appropriate then the target folder for the x64 files can be either the windows system32 folder or any folder on the

applications path (and on the reference path for .net applications). The target folder for the x86 files (used to support Adobe PDF documents) can be any folder except the system32 folder (the SysWow64 folder is often used).

File	Win32 DLL	.Net 2.0+ Lib2.dll	.Net 2.0+ Lib3.dll	Java	COM	Adobe PDF
x64/bardecode_jni.dll				✓		
x64/msvcr90.dll	✓		✓	✓	✓	
x64/pdf2image.dll						✓
x64/pdf2tif.dll						✓
x64/SoftekATL.dll					✓ 	
x64/SoftekBarcode.dll	✓	✓ 	✓	✓	✓	
x64/SoftekBarcodeLib2.dll		✓ 				
anycpu/ SoftekBarcodeLib3.dll			✓			
x64/ SoftekBarcodePDF.dll						✓
x86/cimage.dll						✓
x86/convert.dll						✓
x86/encryptpdf.dll						✓
x86/pdf2image.dll						✓
x86/PDF2ImageCOM.exe						✓ 
x86/pdf2tif.dll						✓
x86/Pdf2tifcom.exe						✓ 
x86/pdf2img.ini						✓

Where Lib2.dll refers to SoftekBarcodeLib2.dll and Lib3.dll refers to SoftekBarcodeLib3.dll



Register with regsvr32.

Register by running “PDF2ImageCOM.exe /regserver” or “Pdf2tifcom.exe /regserver”

Requires install Microsoft Visual C++ 2008 Redistributable Package SP1 for x64 systems

Required only if documents are to be split into PDF format.

Note: There is no .Net 1 component available for 64-bit systems.

21.3 “Any CPU” Applications

For the purposes of the installation of the toolkit, applications compiled for “Any CPU” in Visual Studio will require installation files for the native system i.e x64 files on an x64 system and x86 files on an x86 system.

See also: [Using the .net components on x86 and x64 systems](#)

22 Appendix D: Release Notes

22.1 Version 7.5.1.1

Addition of support for QR-Codes.

22.2 Version 7.4.2.1

Enhanced error information

New functions have been added to the toolkit to retrieve error codes that may assist with the detection and correction of problems. The `GetLastError` function returns the last internal error code for the toolkit and `GetLastWinError` returns the last windows error number.

Suppression of Dialog Messages

The SDK will now suppress all dialog boxes under its control when the process is not visible (e.g. when being run as part of a service). This will prevent processes from hanging.

Partial reads on PDF-417 barcodes

It was possible for certain PDF-417 barcodes to give partial and incorrect readings. This has been corrected. The maximum length for a PDF-417 barcode has now been increased from 2K to 8K (when represented in quoted printable format).

2-D barcodes at coarse fax resolution

Previous versions of the toolkit were not able to decode 2-D barcodes from coarse resolution faxes because the vertical and horizontal resolutions were different. This has been corrected.

Improvements to the SoftekSDKDemo application

The SoftekSDKDemo application will now retain settings in the registry. It is also possible to reset the settings back to default and import and export settings from/to an xml file.

22.3 Version 7.4.2.2

64-bit version of SoftekBarcodeLib2.dll now correctly packaged as a strongly named assembly.

Potential buffer over-run in the Code-128 module fixed.

22.4 Version 7.4.2.3

Support for Desktop specific licensing added to the toolkit.

22.5 Version 7.4.1

22.5.1 Version 7.4.1.1

1. License Key

The most important change in Version 7.4.1 is the requirement to set a license key prior to calling the `ScanBarCode`, `ScanBarCodeFromBitmap` or `ScanBarCodeFromDIB` functions. This will simplify the distribution of updates to the toolkit and protect the rights of licensees.

2. New .net component

This version also includes a new interface to the toolkit for .net applications. The SoftekBarcodeLib3.dll component wraps around the SoftekBarcode.dll and can process TIF documents at 3 times the speed of the SoftekBarcodeLib2.dll component.

3. Simplified installation

All the files for the SDK are now installed into a single folder by running a self extracting installer. No dll files need to be registered unless the COM or OCX interfaces are to be used and so the new version may be used in isolation to previous versions of the toolkit. The install set also includes the dll files for the PDF Extension, with the license key determining whether or not barcode can be read from PDF documents.

4. PDF Processing

Image only PDF documents can now be processed at up to 3X the speed of previous versions. When the PdfImageOnly property is set to True (the default value) the toolkit will assume that all PDF documents are image only (e.g, scanned documents) and use a faster conversion method than would be needed for other types of PDF document.

5. PDF Processing on x64 Systems

This version has resolved a number of issues with processing PDF documents on x64 based systems.

6. Replacement of ImageReader and DeveloperCenter with SoftekSDKDemo

The demonstration applications used in previous versions of the toolkit have been replaced with the SoftekSDKDemo application, which comes in versions for both x86 and x64 based systems.

7. Improvements to barcode recognition

Improvements have been made to the Code 39, Code 128, GS1-Databar and Datamatrix modules.

8. Code 93 Support

Support for Code 93 barcodes has also been added. Please refer to the [manual page](#) for further details.

9. TIF Resolution

If a TIF document uses a resolution of 1 dpi then the value will be ignored and a default value of 200 dpi used instead.

10. DLL Dependencies

The only DLL file that requires installation of the Microsoft Visual C++ Redistributable Package (2008 SP1) is SoftekbarcodeLib2.dll.

22.5.2 Version 7.4.1.2

A PInvoke error when using the SoftekBarcodeLib3.dll under .Net 4 has been fixed.

22.5.3 Version 7.4.1.3

The SetScanRect method in SoftekBarcodeLib3.dll now functions correctly.

Options for extracting images from PDF documents can now be controlled through the [PdfImageExtractOptions](#) advanced property.

A typing error in the product names for the redistribution packages has been corrected.

22.5.4 Version 7.4.1.4

The LicenseKey property added to Java class

An error in the PDF-417 decode algorithm has been corrected.

The datamatrix module has been changed to better handle the effects of perspective in a photograph.

A correction has been made to the way BMP files are loaded by the win32 dll.

22.5.5 Version 7.4.1.5

PdfImageRasterOptions mask property added to allow control over rasterization of PDF documents.

Options dialog in SoftekSDKDemo has been split into tabs.

22.5.6 Version 7.4.1.6

Updated versions of pdf2tif.dll, convert.dll and encrypt.dll added to the install set

22.5.7 Version 7.4.1.7

Added [FilePathEncoding](#) property.

Support for [file paths containing non-ascii characters](#) has been improved.

22.6 Version 7.3.1

1. GS-1 Databar

GS-1 Databar support has been added to toolkit. This includes RSS-14, RSS-14 Stacked, RSS Truncated, RSS Limited, RSS Expanded and RSS Expanded Stacked. There is also support for supplemental information encoded in micro-PDF-417 barcodes.

2. Color Images

A new property called ColorProcessLevel has been added to the toolkit. This property controls the amount of time the toolkit will spend processing a color image and should make it unnecessary to set the ColorThreshold property to a value other than zero. If you are upgrading from a previous version of the toolkit and set ColorThreshold to a non-zero value in your code then you will need to change the value to 0 in order to take advantage of the new property.

3. Micro-PDF-417

A memory violation error has been fixed in the micro PDF-417 module.

4. Size of color PDF files

The compression used when outputting PDF documents as part of the Tiff Split process has been changed from zip to jpeg, which yields significantly smaller files.

5. Java Interface

The properties; ReadMicroPDF417, ReadDatabar and ReadDataMatrix have been added to the java interface.

6. SkewTolerance Range

Range checks have been added for the value of the SkewTolerance property.

7. SofttekBarcodeLib2.ScanBarcodeFromBitmap empty string problem

It was possible for the ScanBarcodeFromBitmap method in the .net 2 interface to return a value of 1, and yet GetBarString would return an empty string. This has been fixed and the barcode value is now returned.

8. Median Filter memory leak

A memory leak has been fixed when MedianFiler is set to True.

9. Double read on skewed DataMatrix barcodes.

An error whereby it was possible to get a double read for slightly skewed datamatrix barcodes has been fixed.

10. Empty values for Datamatrix

An error whereby the toolkit returned empty datamatrix barcode values for some images has been fixed.

22.7 Version 7.2.1

1. Datamatrix Barcode Support

Support has been added for Datamatrix ECC 200 barcodes. To enable the reading of datamatrix barcodes please set the ReadDataMatrix property to True.

2. Memory leak fix

Previous versions of the toolkit contained a memory leak if SkewTolerance > 0 and UseOverSampling was set to True or 1.

3. Support for 2 and 3 digit Code 25 barcodes.

If MinLength is set to an appropriate value then the toolkit will recognize 2 and 3 digit Code-25 barcodes.

4. Potential memory over-run when reading micro-PDF417 barcodes fixed.

5. Output of split files in PDF Format

A potential memory exception has been fixed when export TIF split pages in PDF format.

6. Reading non-TIF files from URL's

A bug which meant that the toolkit always assumed a file read from a URL was in TIF format has been fixed.

22.8 Version 7.1.4

1. Tiff-split in PDF Format

The tiff-split feature of the toolkit is now able to output in both tiff and pdf format. Note that the .Net component requires installation of the PDF Extension to do this where as the DLL, OCX and COM interfaces do not. If the PDF Extension is installed then the toolkit is also able to split a PDF file into smaller parts. Note that PDF files are rasterized before being split.

2. Java Interface Extended

The Java interface has been extended to include tiff-split functionality.

3. PdfDpi default value changed

The default value of the PdfDpi property (the resolution at which PDF files are rasterized) has been changed from 200 to 300.

4. Patch Code Recognition

Previous versions of the toolkit would output false positive results if ReadPatchCodes was set to True and MinOccurrence and PrefOccurrence were left at default values, but version 7.1.4 imposes a minimum hit count for patch codes of 30. This value can only be changed via the LoadXMLSettings method/function by setting the property PatchCodeMinOccurrence.

5. PDF-417 Recognition

Improvements have been made to PDF-417 recognition, which allow the toolkit to cope with larger variations in the width of columns.

22.9 Version 7.1.3

1. Improved support for the EAN-13 family of barcodes.

The EAN-13 module has been improved to allow a greater variance in module width.

2. Extended support for Code 2 of 5 barcodes.

Support added for the following variants of the Code 25 family: Datalogic, Matrix, Industrial and IATA via the ReadCode25ni setting.

3. Improved detection of false positive results when using SkewSettings > 0.

In previous versions it was possible to get partial reads of barcode such as Code 2 of 5 when using SkewSetting values > 0. This version detects and removes such errors.

4. Encoding property added to control format in which barcodes are returned:

0 (default) = raw (with null characters suppressed)

1 = Quoted printable

2 = Unicode

3 = UTF-8

Note that this property mainly applies to PDF-417 barcodes.

5. Sorting of barcode results

Previous versions of the toolkit have generally returned barcodes ordered bottom up and left to right - with some exceptions. This version now checks the ordering to ensure that barcodes are returned in the expected order.

6. Bug fix in SaveResults method in the .net 2.0 component.

In previous versions this function only saved the results for the last page in a multi-page TIF document.

7. The properties PdfBpp, PdfDpi, PageNo and Photometric can be exported and imported using the XML interface.

8. Added support for micro-PDF-417 barcodes via the ReadMicroPDF417 property.

9. Added support for Short Code 128 barcodes via the ReadShortCode128 property. Another property called ShortCode128MinLength has also been added to control the minimum allowed length for this type of barcode. The default value is 2.

10. Improved recognition of long Code 128 and Code 39 barcodes.

22.10 Version 7.1.2b

1. EAN-13/UPC-A Barcodes

Correction to potential parity checking problem.

22.11 Version 7.1.2

1. Code 39 Module

The Code 39 module has been improved to handle barcodes with split bars.

2. Java Interface

SetScanRect, GetBarStringRect and GetBarStringPage methods added to the jni interface. Please see the README.txt file in the java folder for further details.

3. Tiff Split

The TiffSplitPath property can now include %s as well as %d. The %s code will be replaced by the value of the barcode in the sub-file.

4. JPG Loading Errors

The handling of error conditions when loading a jpg file has been improved.

22.12 Version 7.1.0a

Modified DeveloperCentre application to automatically switch to XML settings if no barcode found.

22.13 Version 7.1.0

Median Filter, XML and Java Interfaces.....

1. Median Filter

The Median Filter option is another method of cleaning noisy images. It's more versatile than the noise reduction filters because it isn't restricted to a single orientation, although it isn't recommended for low resolution images.

2. XML Interface Added

2.1 Using an XML File to Control Properties

It is now possible to load settings for the toolkit from an XML file. What's more, it is also possible to define groups of settings, to be applied successively to an image until a barcode is found. Groups of settings can also be targeted at particular pages in an image.

See the manual pages for LoadXMLSettings and ExportXMLSettings for more details.

2.2 Exporting Results to an XML or CSV File

The SaveResults method can be used to export barcode values and locations to an XML or CSV file.

See the manual page for SaveResults for more details.

2.3 Using an XML File to Specify Files or Folders to be Processed.

An XML file can also be used to specify files and folders for the toolkit to scan, with the results written to another XML file or a CSV file.

Note that this does not currently support PDF files.

See the manual page for ProcessXML for more details.

3. Java Interface

The toolkit is now provided with a Java class, which can be found in the java folder under the installation folder. Please see the README file in the Java folder for more details on the supported properties and methods.

4. Code 25 character width tolerance

The allowed variation in width for Code 25 characters in barcodes of more than 6 characters has been increased.

5. Extra wide spaces

The toolkit now handles barcodes with an exceptionally high ratio between the first black bar and first white space.

6. Auto Color Threshold Calculation Improved

The automatic calculation of the value for color threshold has been improved.

7. PDF-417 Recognition

An improvement has been made to the PDF-417 algorithm, which allows it to recognise much lower resolution images than was previously possible. A logic error has also been fixed in the decoding algorithm for byte compacted data.

8. Code 39 Recognition

A small improvement has been made to the Code 39 algorithm to handle barcodes with a low ratio between the wide and narrow bars.

22.14 Version 7.0.10

1. Code 128 recognition.

A small change to help read barcodes where the size of the black bars have been shrunk by the scanning process.

22.15 Version 7.0.9

1. Auto-calculation of Color Threshold

The algorithm to calculate the color threshold value has been improved to handle a greater variety of color images. The new method also recognizes color images that only use 2 unique colors (i.e. black and white encoded as color).

2. Code 128 recognition

The variation in character width for a code 128 barcode has been increased to allow better detection in color images. A restriction in character width variation is still used for reasons of performance.

3. BMP files using less than 24 bpp and with no image size specified in the header.

If a BMP file with less than 24 bits per pixel contains no image size in the header then the toolkit will now calculate the expected image size from the width and height.

4. Memory leak when Using Noise Reduction on Color Images

If a color bitmap was processed with the NoiseReduction property set to a value great than zero and ScanDirection exclusively portrait or landscape, then a memory leak occurred. The leak has now been fixed.

22.16 Version 7.0.8

1. A logic error in TifSplitMode has been fixed.

If TifSplitMode was set to zero then the second barcode in the image would appear on the last page of the first file, rather than the first page of the second file.

22.17 Version 7.0.7

1. Code 128 recognition

The variation in character width for a code 128 barcode has been increased.

22.18 Version 7.0.6

1. Photometric property added to the Managed Components

22.19 Version 7.0.5

1. The following combination of settings in versions 7.0.3 and 7.0.4 was causing the ScanBarCode function to return a value of 0, but has been corrected in this version.

PageNo != 0

MultipleRead = True

ScanDirection = 1, 4 or 5

2. The recognition of PDF-417 Barcode with small column widths has been improved.

3. The default value for the ReadPDF417 property has been changed from TRUE to FALSE

22.20 Version 7.0.4

1. Automatic setting of ColorThreshold

If the ColorThreshold property is set to a value of zero then the toolkit will automatically calculate a value. This enables the toolkit to read barcodes from very dark or very light images.

2. Code 128 Recognition

The tolerance level for differences in the width of Code 128 characters has been increased.

22.21 Version 7.0.3

1. PNG Image Support

The Windows DLL, OCX and COM interfaces now all support the PNG file format.

2. PDF-417 Recognition Improved

The PDF-417 module has been improved to handle images with the following defects:

a. The scanning process has reduced the size of the black bars.

- b. Part of the end pattern is missing.
- c. The column widths vary through the image.
- d. The bases of some columns are missing.

3. Code 39 Error Correction/Oversample Bug

A buffer over-run which occurred on some images where Code 39 error correction and oversampling had both been set has been fixed.

22.22 Version 7.0.2

1. Splitting TIF files according to the location of the barcodes in the image.

A new mode has been added, which will split a TIF file and throw away the pages containing a barcode. For example, if an image consists of 10 pages, with barcodes on pages 1, 4 and 6 then the toolkit will create 3 new files. The first file will contain pages 2 and 3. The second will contain page 5 and the third will contain pages 7, 8, 9 and 10.

Set TifSplitMode to a value of 2 to use this mode.

2. Code 128 recognition improved.

During the scanning process it is possible for the black bars to increase in size and for the white spaces to shrink. The toolkit has always allowed for this problem, but it meant that the Code 128 module sometimes mis-read a character value and so failed to return a value for the barcode. This has been corrected and some barcodes that were not detected will now read OK.

22.23 Version 7.0.1a

1. PDF-417

PDF-417 reading capability has been added to the toolkit (please see note above). Control over PDF-417 reading is via the ReadPDF417 property, which by default is set to False.

2. Code 25 Module modified to handle differences in character widths in high resolution images.

3. MaxLength increased from 99 to 999.

4. Code 39 checksum calculation fixed.

22.24 Version 6.2.1a

1. Version numbers brought into line on the dll and ocx files.

2. Code 25 module - maximum ratio value increased from 3 to 4.

22.25 Version 6.2.1

1. Code 39, Code 25 and Codabar Recognition

The rules for the above barcode symbologies have been tightened to reduce the possibility of a false positive reading.

22.26 Version 6.2.0d

1. Code 39 Recognition

Further enhancements have been made to Code 39 recognition:

If the Code39Checksum property is set to True then the toolkit will only report Code 39 barcodes where the last character is a valid checksum for the rest of the string.

If the ExtendedCode39 property is set to True then the toolkit will attempt to interpret the barcode in the Code 39 Extended symbol set.

2. Code 25 Recognition

The Code 25 recognition module has been modified to allow for barcodes where the width of a module for a bar may be different to that of a space.

3. Regular Expression Matching

The Pattern property now allows applications to specify a regular expression that all reported barcodes must match against. For example, Pattern = "^ABC[0-9]+XYZ\$" would match barcodes similar to "ABC123456XYZ".

If the related property called Numeric is set to True then it is the equivalent of setting Pattern to "[0-9]+\$".

4. Improved Code 128 Recognition

The Code 128 recognition module has been improved to cope with barcodes where the bar widths have been distorted by the process of scanning or faxing.

5. Error Correction

If the ErrorCorrection property is set to True then the toolkit will attempt to find the best match for a barcode character where it is practical to do so. This currently only applies to Code 39 barcodes, but will be extended to other types in the future.

6. Minimum Space Bar Width

The MinSpaceBarWidth property specifies the smallest width of a space in a barcode. Any spaces smaller than this width will be ignored by the toolkit. This is very useful when a scanned image contains lots of white dots in the black bars. Setting the property to a value of 2 or 3 can often result in a reliable read for such images.

7. Control over PDF Conversion

Where the PDF Extension to the toolkit is installed, it is now possible to control the resolution and color depth of the image created from the PDF file, and used to locate the barcodes. This allows applications to control the speed and accuracy of the process. The PdfBpp property controls the number of bits-per-pixel of the converted image (1, 8 or 24) and PdfDpi controls the number of dots per inch of the converted image. Lower values for either property lead to an increase in speed.

8. Default value of LineJump property has been changed from 9 to 1.

9. Improvement in speed

Optimization in the Code 25, Code 39 and Code 128 module has given a significant improvement in speed.

22.27 Version 6.1.1

1. PDF Conversion

The ScanBarcode method now uses the PDF Extension to convert the pdf document at 8 bits per pixel.

2. ZIP Compression

The toolkit now supports TIF documents that use ZIP compression.

3. ScanDirection Mask Change

The managed component interface has been modified to interpret the ScanDirection mask in the same way as the standard windows dll. This may mean that some applications will need changing if they use the managed component and set a non-default value for the mask. If in doubt then please contact support@bardecode.com for further information.

4. Percentage mapping Mode for SetScanRect

A new mapping mode is now available for SetScanRect. If a value of 1 is used for the mapping mode then the units are treated as a percentage of the width or height of the image.

5. SoftekATL COM Object Trial Version

The trial version of the SoftelATL COM Object has been modified to prevent the display of the trial version dialog box, which caused problems for web servers. The new trial version replaces the last 3 characters of the barcode with a * character.

6. HBITMAP and DIB Support in the Managed Component

The managed component interface now supports the ScanBarcodeFromBitmap and ScanBarcodeFromDIB methods.

7. UPC-E Recognition

A bug that prevented the recognition of valid UPC-E barcodes has been fixed.

8. Images with Small Height

The toolkit now correctly handles images with a height of less than 4 pixels.

9. Old JPEG Compression

If a TIF file contains pages that use old jpeg compression then the return value will either be the number of barcodes found on other pages or, if no other barcodes were found, a value of -5.

10. Code 39 and Code 25 Recognition

The algorithms for Code 39 and Code 25 have been improved to cope better with poor quality barcodes.

11. Memory Leak in the .Net Component

A memory leak has been fixed in the .Net component. The memory was leaked each time an instance of the component was destroyed.

22.28 Version 6.1.0

1. PDF File Format

When used in conjunction with the "SofttekBarcode Toolkit PDF Extension" library, this version of the toolkit can read barcodes from PDF files.

2. Improvement in Performance

The time it takes for the toolkit to load TIF files has been significantly improved by code optimization. This means that barcodes can be read from TIF files at much higher speeds than before (does not apply to the managed component).

3. Code 25 Non-Interleaved Recognition available across all interfaces

The ReadCode25ni property has now been added to all interfaces. The default for this property is False.

4. Photometric Property Added

The Photometric property is used with the ScanBarcodeFromBitmap method to specify the bit value representing Black in a 2 color bitmap. Default of 0.

5. AllowDuplicateValues Property Added

The AllowDuplicateValues property controls whether the toolkit allows barcodes with identical values on the same page to be ignored. Default of True

6. Properties of Managed Component not used when calling ScanBarcodeFromBitmap method.

Any properties set in the managed component were previously being ignored when calling the ScanBarcodeFromBitmap method. The method has now been modified to fix the settings each time it is called.

7. Softtek COM Object Memory Problem Fixed

A problem in the COM object interface to the toolkit has now been fixed. This caused the BSTR values returned from certain functions to become corrupt.

8. Full Multi-Page Support for Managed Component

The Managed Component now provides full multi-page image support. The default for the PageNo property is now zero - which means that all page of an image will be checked for barcodes.

22.29 Version 6.0.10

1. GIF file support for DLL, OCX and COM Interfaces.

This also includes support for multi-page GIF files.

2. Multi-page Image Support in the Managed Component Interface

The .Net managed Component now supports barcode reading from multi-page image files. It still differs from the regular DLL though because it will only search the page in the image specified by the PageNo property. If PageNo is set to zero then the component will throw an exception when the image is loaded.

3. Access Violation Problems with OverSampling

When OverSampling was used with certain images the barcode toolkit read 1 scan line beyond the length of the image and caused an access violation. This has now been corrected.

4. ColorThreshold level problem resolved

A problem with setting high values for ColorThreshold has been resolved.

22.30 Version 6.0.9

1. The ScanBarcodeFromBitmap function has been fixed so that the HBITMAP handle isn't deleted.
2. Problem Reading from 32 bits-per-pixel bitmap handles.

The function that converts 32-bit color images to black and white used an incorrect value for the color threshold, which meant that perfectly good barcodes were not detected by the software.

3. The ScanBarcodeFromBitmap function has been changed to read the bitmap in the same direction as the toolkit reads other image formats. This means that any rectangle specified by the SetScanRect method should now be measured in pixels from the top left hand corner, rather than the bottom left hand corner.

22.31 Version 6.0.8

1. Memory leak fixed in ScanBarcodeFromBitmap

22.32 Version 6.0.7

1. Access Violation in ScanBarcodeFromBitmap

A small section of code that tried to determine if a DIB handle had been passed to ScanBarcodeFromBitmap has been removed because it was causing access violations. DIB handles should be passed to the ScanBarcodeFromDIB function.

2. The 100% managed component is now "strongly named".

22.33 Version 6.0.6

1. Support for JPEG compressed TIF files.
2. Bug fixed in the splitting of multi-page tif files.

Description: It was possible for barcodes in a multi-page TIF file to be repeated. This has been fixed.

3. libbarcode.jpg file is no longer required.

Description: The functionality for reading JPG files has been moved inside the SoftekBarcode.dll file.

4. Non-Interleaved Code-25 Capability

The functions `stSetReadCode25ni` and `stGetReadCode25no` have been added to the DLL interface. These functions have the same calling convention as `stSetReadCode25` and `stGetReadCode25` and control whether the toolkit searches for non-interleaved Code 25 barcodes. The default is for the toolkit not to search for these barcodes.

22.34 Version 6.0.4

1. Exception Error in Dot Net Component

Description: Certain image files caused an exception error in the dot net component, especially when using options such as noise reduction or over sampling.

Solution: `SoftekBarcodeLib.dll` has been modified to interpret the byte width of image scan lines correctly.

2. Reading Barcodes from DIB Handles

A new function called `ReadBarCodeFromDIB` has been added to the toolkit. The old function called `ReadBarCodeFromBitmap` will now try to determine if the handle is for a BITMAP or a DIB and call the `ReadBarCodeFromDIB` function if necessary.

3. Reading Skewed Barcodes.

The 45 degree angle mask values for the `ScanDirection` mask are no longer used. The reading of skewed barcodes is now controlled by the `SkewTolerance` property which ranges in value from 0 (no check for skewed barcodes) to 5 (all angles considered). The default value is 0.

22.35 Version 6.0.3

1. Code 39 barcodes with length less than 4 characters would not read.

Solution: The toolkit will now read Code 39 barcodes of any length. Remember though that the purpose of the `MinLength` setting is to cut down on the chance of a false positive reading.

2. Photometric Interpretation of Monochrome BMP files - when using the .Net component.

The corresponding fix from version 6.0.2 has been applied to the .Net Component.

22.36 Version 6.0.2

1. Photometric Interpretation of Monochrome BMP files

Description: BMP files that used a zero bit to represent a white pixel were being displayed in negative by the image viewer and interpreted as a negative image by the barcode reader.

Solution: SoftekBarcode.dll has been modified to read the color map correctly.

2. Some BMP files would read OK on Windows 2000 but would fail to read on Windows XP.

Description: The bmWidthBytes member of the BITMAP structure is sometimes mis-reported by Windows XP. The value is often rounded up to an integer divisible by 4. This leads to the barcode toolkit loading the image into memory incorrectly and either missing the barcode, getting the position wrong or reporting multiple occurrences of a barcode.

Solution: The length of the bitmap data is now checked to make sure that it matches the reported size of bmWidthBytes. The fix is in SoftekBarcode.dll.

The photometric interpretation on monochrome bitmap files is now checked before processing. Files that represented white with a zero bit were being read as a negative image

22.37 Version 6.0.1

The main change to the barcode toolkit for version 6 is the ability to work in a multi-threaded environment. A managed component has been added to the set of interfaces, and allows easy deployment of the toolkit within the .Net framework.

The algorithms for the Code 3 of 9 and Code 2 of 5 Interleaved barcode types have been re-written to provide a better success rate with low resolution images.

It's also worth noting why version 5 was never released. Just as version 5 got to the pre-release stage, the need for a thread safe version of the toolkit was identified and was given top priority.

Summary of changes:

1. Support for the Codabar symbology.

The toolkit now supports recognition of the Codabar barcode type. The ReadCodabar property has been added, with a default value of True.

2. Improved recognition algorithms for Code 3 of 9 and Code 2 of 5 Interleaved.

Both of the algorithms for detecting Code 3 of 9 and Code 2 of 5 Interleaved barcodes have been improved. This particularly applies to low resolution images where there may only be minor differences between the wide and narrow bars.

3. 100% Managed .Net Component added to the range of interfaces.

The managed component is implemented by the file SoftekBarcodeLib.dll. This interface supports most of the properties and methods of the toolkit with the exception of the following:

ScanBarCodeFromBitmap

TifSplitPath

TifSplitMode

BitmapResolution

The component uses the .Net class for loading images and so supports a greater range of image types (e.g PNG, GIF) in addition to the usual set.

4. UPC-E to EAN-13 Conversion

UPC-E is a zero suppressed version of UPC-A/EAN-13 and the default in version 4 was to restore the suppressed zero's. The option has been added in version 6 to leave the barcode as it is printed, via the ConvertUPCEToEAN13 property. The default value is True - which means that the zero's will be restored (the behaviour in version 4).

5. Retrieve the Direction of a Barcode in an Image

A new method has been added to retrieve the orientation of a barcode. The GetBarStringDirection method returns a value that, if used for the ScanDirection property, would detect the barcode. Some barcodes can be read from left to right or right to left. In these cases the direction returned is the first match found.

6. Code39Draw control

The Code39Draw control has been fixed so that it prints out the same size as it displays on the screen.

7. 45 degree angle scanning

4 new scan directions are available in the toolkit. The default value for the ScanDirection mask is still set to 15, which covers left-right and top-down directions. Please refer to the manual for full details of the new values for the mask.