

RasterEdge .NET Imaging SDK

Developer’s Guide

RasterEdge.DocImagingSDK 9.5.0

2015-7-2

**Table of Contents**

**Introduction ...............................................................................................................**

**About RasterEdge Document Imaging .................................................................................7**

**Deploy RasterEdge Imaging ................................................................................................8**

**Online Demo ......................................................................................................................8**

**Where to seek help ............................................................................................................8**

**Install SDK .................................................................................................................**

**Install ................................................................................................................................9**

**F&Q ..................................................................................................................................9**

**License ..................................................................................................................... 12**

**Trial License ................................................................................................................... 12**

**Purchase License .............................................................................................................. 12**

Web Server License.............................................................................................................................. 12

**XDoc. HTML5 Viewer .............................................................................................. 13**

**Overview ......................................................................................................................... 13**

**Tutorial ........................................................................................................................... 13**

F&Q ..................................................................................................................................................... 13

**WinForm Document Viewer ..................................................................................... 15**

WinViewer Overview .......................................................................................................................... 15

Feature List ...................................................................................................................................... 15

Control APIs. ................................................................................................................................... 15

Add a WinViewer to Your Project....................................................................................................... 16

Setting Up Your Project .................................................................................................................. 16

Open File from Specified Path ........................................................................................................ 17

Draw Specified Annotation on Page ............................................................................................... 19

Customize WinViewer Control............................................................................................................ 20

Customize Toolbar .......................................................................................................................... 20

Customize Annotation Style ............................................................................................................ 20

Customize the Way Open and Store Files ....................................................................................... 22

Customize the Drop-down List ....................................................................................................... 23

**XDoc.Converter ........................................................................................................ 24**

**Overview ......................................................................................................................... 24**

**Tutorial ........................................................................................................................... 24**

**XDoc.PDF ................................................................................................................ 25**

**Overview ......................................................................................................................... 25**

**Tutorial ........................................................................................................................... 25**

**Programming with RasterEdge Document Imaging SDK .......................................... 26**

**Overview ......................................................................................................................... 26**

**Register Process .............................................................................................................. 28**

**TIFF ............................................................................................................................... 29**

TIFF File Overview ............................................................................................................................. 29

Requirements ....................................................................................................................................... 29

TIFF Programming Classes ................................................................................................................. 29

Construct, Load and Save a TIFFDocument. ....................................................................................... 30

Load or Create a TIFFDocument .................................................................................................... 30

Save a TIFF File .............................................................................................................................. 31

Clear Temp files generated in the process ........................................................................................... 31

Get preview of Tiff Document. ........................................................................................................... 31

TIFF Rendering and Conversion ......................................................................................................... 32

TIFF Annotation .................................................................................................................................. 32

*Editing TIFF Metadata* ........................................................................................................................ 33

Add, Delete, Update and Find a TIFF Field .................................................................................... 33

Add, Delete, Update and Find a EXIF Field ................................................................................... 34

Add, Delete, Update and Find a IPTC Field .................................................................................... 34

Get the XML Information ............................................................................................................... 35

Editing TIFF Document ....................................................................................................................... 35

Re-Order Pages of TIFF File ........................................................................................................... 35

Swap two pages of TIFF File ......................................................................................................... 36

Insert Pages to TIFF File ................................................................................................................. 36

Delete Pages from TIFF File ........................................................................................................... 36

Split TIFF File ................................................................................................................................. 37

Combine TIFF File .......................................................................................................................... 37

Generate Barcode on TIFF .................................................................................................................. 37

Read Barcode Information from TIFF ................................................................................................. 38

How to ................................................................................................................................................. 38

How to View Thumbnail Image of a TIFF Document .................................................................... 38

How to Generate Magnified Images of TIFFDocument. ................................................................. 39

How to Add Watermark onto TIFF Document................................................................................ 40

**Word ............................................................................................................................... 42**

Introduction to Word Functions. .......................................................................................................... 42

.docx and .doc file formats. .................................................................................................................. 42

Load and Save Word Document .......................................................................................................... 43

About Word Programming Classes ................................................................................................. 43

Word Document Object. .................................................................................................................. 43

Load Word document ...................................................................................................................... 43

Save Word Document ..................................................................................................................... 44

Get Preview of Word Document......................................................................................................... 44

Annotations on Word Document ......................................................................................................... 44

Word Rendering and Conversion......................................................................................................... 45

Create Barcode in Word ...................................................................................................................... 45

Read Barcode from Word .................................................................................................................... 46

How to’s .............................................................................................................................................. 46

How to: Create Thumbnail of DOCXDocument ............................................................................. 47

**Excel ............................................................................................................................... 48**

Introduction to Excel Functions. .......................................................................................................... 48

.xlsx and .xls file formats ..................................................................................................................... 48

About Excel Programming Classes ..................................................................................................... 48

Excel Document Object ....................................................................................................................... 49

Load Excel Document ......................................................................................................................... 49

Save Excel Document .......................................................................................................................... 50

Get preview of Excel Document ......................................................................................................... 50

Annotations on Excel Document ......................................................................................................... 50

Excel Rendering and Conversion......................................................................................................... 51

Create Barcode in Excel ...................................................................................................................... 51

Read Barcode from Excel .................................................................................................................... 52

How to’s .............................................................................................................................................. 52

How to: Create Thumbnail of Excel. ............................................................................................... 52

**PowerPoint...................................................................................................................... 54**

Introduction to PowerPoint Functions ................................................................................................. 54

.pptx and.ppt file formats ..................................................................................................................... 54

About PowerPoint Programming Classes ............................................................................................ 55

PowerPoint Document Object ............................................................................................................. 55

Load PowerPoint Document ................................................................................................................ 55

Save PowerPoint Document ................................................................................................................ 56

Get Preview of PowerPoint Document ............................................................................................... 56

Annotations on PowerPoint Document. ............................................................................................... 56

PowerPoint Rendering and Conversion ............................................................................................... 57

Create Barcode in PowerPoint ............................................................................................................. 57

Read Barcode from PowerPoint .......................................................................................................... 58

How to’s .............................................................................................................................................. 58

How to: Create Thumbnail of PowerPoint Document ..................................................................... 58

**DICOM ........................................................................................................................... 60**

DICOM Overview ............................................................................................................................... 60

Programming with DICOM ................................................................................................................. 60

Load DICOM File. ............................................................................................................................... 60

DICOM Rendering and Conversion .................................................................................................... 60

**JBIG2 ............................................................................................................................. 61**

JBIG2 Codec Overview ....................................................................................................................... 61

Feature List .......................................................................................................................................... 61

How to Decode an JBIG2 Image ......................................................................................................... 61

**JPEG 2000 ...................................................................................................................... 62**

JPEG 2000 Codec Overview ............................................................................................................... 62

Feature List .......................................................................................................................................... 62

How to Decode a JPEG 2000 Image. ................................................................................................... 62

**RasterEdge OCR ............................................................................................................ 63**

**RasterEdge RasterImage ................................................................................................ 64**

RasterImage Overview ........................................................................................................................ 64

Requirements ....................................................................................................................................... 64

About RasterImage Programming Classes .......................................................................................... 64

Convert Image ..................................................................................................................................... 64

Load Image .......................................................................................................................................... 65

Save Image .......................................................................................................................................... 66

Image Process ...................................................................................................................................... 67

**Programming with Images ........................................................................................ 69**

**Overview ......................................................................................................................... 69**

**Image Concept ................................................................................................................ 69**

REImage, the Core Programming Class for Images ............................................................................ 69

Image Data ........................................................................................................................................... 70

Image Compressions. ........................................................................................................................... 70

Why Compression ........................................................................................................................... 70

Types of Compression ..................................................................................................................... 70

Image Codecs....................................................................................................................................... 71

Supported Formats .......................................................................................................................... 71

**REImage the Core Image Class in RasterEdge Imaging SDK ......................................... 71**

Introduction ......................................................................................................................................... 71

Requirements ....................................................................................................................................... 72

How to Create REImage ...................................................................................................................... 72

Create REImage from Image File, Stream & Byte Array. ............................................................... 72

Create Image from Bitmap .............................................................................................................. 73

Annotate on REImage. ......................................................................................................................... 73

Save REImage ..................................................................................................................................... 74

**ImageProcessing ............................................................................................................. 74**

**Annotations ..................................................................................................................... 75**

Introduction to Annotations ................................................................................................................. 75

Requirements ....................................................................................................................................... 76

Generate an AnnotationObject ............................................................................................................. 76

**How to Generate an Annotation Object Programmatically ............................................. 76**

Burn Annotation to Document or Image.............................................................................................. 77

How to Burn Annotation Object to Document (PDF, TIFF, WORD, EXECEL, PPT) ................... 77

How to Burn Annotations on Images .............................................................................................. 78

Annotations on ASP.NET DocumentViewer or Windows Form DocumentViewer ........................... 79

Annotation Assemblies ........................................................................................................................ 79

**Metadata ......................................................................................................................... 81**

Introduction to Metadata. ..................................................................................................................... 81

Supported Metadata Types .............................................................................................................. 81

Image Formats Supporting Metadata .............................................................................................. 82

EXIF Metadata..................................................................................................................................... 82

Parse Exif Metadata from TIFF File. ............................................................................................... 82

Embed Exif to TIFF ........................................................................................................................ 82

Parse & Update Exif from File ........................................................................................................ 82

XMP .................................................................................................................................................... 83

**Barcode Read .................................................................................................................. 84**

How to’s .............................................................................................................................................. 84

How to: Read Barcode from Image ................................................................................................. 84

How to: Read Barcode from Document .......................................................................................... 85

Advanced ReaderSettings .................................................................................................................... 87

**Barcode Create ............................................................................................................... 88**

How to’s .............................................................................................................................................. 88

How to: Draw Barcode on Image .................................................................................................... 88

How to: Create Barcode and Save as Image .................................................................................... 90

Advanced Settings ........................................................................................................................... 91

**TWAIN Scanning. ........................................................................................................... 99**

TWAIN Scanning Overview ............................................................................................................... 99

Acquisition ...................................................................................................................................... 99

TWAINDevice ................................................................................................................................ 99

Getting Started with RETwain ............................................................................................................. 99

Setting Up Events .......................................................................................................................... 100

Getting and Setting Properties ....................................................................................................... 100

How to’s ........................................................................................................................................... 101

How to Do Console Based Scanning ............................................................................................. 101

How to Scan Many Pages into a PDF or TIFF file ........................................................................ 102

**Licensing RasterEdge Imaging. ..................................................................................... 105**

Purchasing License ............................................................................................................................ 105

**Introduction**

**About RasterEdge Document Imaging**

RasterEdge Document Imaging is a powerful document imaging SDK and controls. With easy to

use document imaging APIs, user can implement functions of loading, saving, converting,

annotating and editing documents and images files. Supported Document formats include TIFF,

PDF, Microsoft Word, Excel, PowerPoint and DICOM. Png, Jpeg, gif, Bitmap among other

commonly used image formats are supported as well. The toolkit also includes a Windows

Forms control library, a Zero footprint ASP.NET AJAX-Enabled Server-Side document Viewer, and

a Twain library for Twain scanning. Annotation module is embedded in both Windows Form and

ASP.NET Web Form controls. Please try our online demo at http://www.rasteredge.com/dotnet-

imaging/web-viewer-demo/.

The product feature includes:

 Create document object from file, stream or byte array.

 Create PDF or TIFF using image source obtained from file, data base or scanning process.

 Load and parse Microsoft Word, Excel, and PowerPoint from file, stream

 Display PDF, TIFF, Word, Excel and PowerPoint with customized options.

 Batch conversion from supported documents to various image formats, such as JPEG,

PNG, and GIF.

 Convert (print) TIFF, PDF, Microsoft Word, Excel, PowerPoint, and DICOM into PDF file

format.

 Generate thumbnails image of document by setting zoom factor or resolution.

 Draw and burn annotations on documents and images.

 Advanced Windows Form control to load, display, annotate, OCR and save above

document formats.

 Zero footprint WebForms document viewer controls to load, display, annotate, convert,

search and save above document formats with featured AJAX technology.

 Add or scan barcode image from documents.

 Add image (logo or watermark) to specified page in document.

 Document processing like inserting, deleting, and reordering PDF document pages or

TIFF pages in multipage TIFF.

 APIs to combine, split, extract TIFF and PDF documents or pages and enable batch

operations on document collection.

**Deploy RasterEdge Imaging**

Because RasterEgde Imaging toolkit is a collection of assemblies, you need to add reference to

specific assemblies in your project to use the APIs provided.

To add reference to RasterEdge Imaging assemblies in your project using Visual Studio:

a. In the Solution Explorer, right-click on **References**, and click **Add Reference**. You download

from our server.

b. Click the browse menu tab to locate RasterEdge.DocImageSDK8.x\Bin

c. Choose dlls that you want to use in your project.

d. Add using statement at the top of the code, for example

using RasterEdge.Imaging.Basic;

**Online Demo**

Using RasterEdge document sdk, you can build a project to enable document viewing,

annotating, converting, and saving. We develop an online document viewer program to

illustrate these features.

You cannot only try using our demo documents but upload your own documents as well.

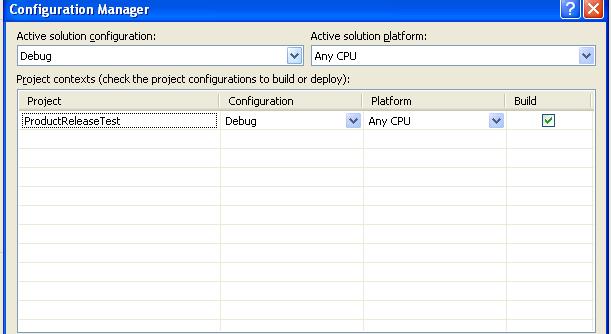
See the demo at

http://www.rasteredge.com/dotnet-imaging/web-viewer-demo/

**Where to seek help**

If you have any question, problems or concerns about RasterEdge products, please contact us

atsupport@rasteredge.com or you can leave a message at our forum at forum.rasteredge.com



**Install SDK**

**Install**

RasterEdge SDK is designed to be easy to use. So no installation process is needed.

**F&Q**

1. Q: error: Cannot find RasterEdge name space occurs when adding reference

A: right click on projects, select properties, under application tab, change target framework

to frameworks other than Client Profile version. Rebuild you application

2. Q: Exception of assembly discripency occurred.

A: Please check if your project setting matches the dll you referenced. If you are using an

x64 operating system and choose AnyCPU as target platform, you should reference x64 dlls

in the SDK Package.

3. Q: Exception of registered error thrown in run time. E.g. Image encoder TIFF is not

registered

A:

 First please make sure that you download the latest version of the SDK at

RasterEdge.com



 Then please add the line of code WorkRegistry.Reset()before you use RasterEdge dlls

APIs, This code register all RasterEdge dlls referenced to your projects. This code need

only to run only once.

 Please make sure that you have already referenced correct dlls to your project. For

example if you want to convert word to pdf, you should add both pdf and word dlls to

your project.

 If the problem persists, please check if the dll in your bin folder is blocked by your OS.

Right click on the dlls, and select properties, then click unblock button.

After unblock all of the dlls you used, make sure to add reference to the unblocked ones

and rebuild the project.

 If possible, move your project to hard drive other than C drive where you might not

have the authority to read and write assembly files.

If the problem still yet solved, please see the following:

If your project is a console or winform application. Make sure that the dlls you referenced

are in the same folder as the .exe application.

Or if it is an asp.net project, please make sure that the dlls you referenced are retrievable in

the Bin folder of your asp.net project. Namely the dlls should be located under the path

which is the result of this statement.

string projectName = HttpRuntime.AppDomainAppPath.Replace("\\", "/") + "/Bin/";

If there is no such path/directory existed, please create one and copy paste dlls accordingly.

If you have any other questions or concerns about RasterEdge product, you can contact us

atsupport@rasteredge.com.

4. Q: Error load program with incorrect format

A: This is often caused by incompatibility between platform target property and the dlls

build. Right click on projects and select properties. In build tab select proper platform target.



5. Q: In Asp.net project when testing web viewer control, get the error:

*Could not load file or assembly 'RasterEdge.Imaging.Annotation' or one of its dependencies.*

*An attempt was made to load a program with an incorrect format.*

A: This is because Visual Studio uses a 32-bit web server internally. You can first debug with

x86 webviewer dlls in Visual Studio which mimic IIS environment. Once done debugging with

x86 dlls in Visual Studio, replace the x86withx64 version of dlls and deploy the project to the

IIS on the servers.

6. Q: When testing web viewer control, I can’t upload document successfully

A: Don’t add reference to SVGWebViewer.dll to your project. Make sure the Default page to

start is the Default.aspx. Append*/Default.aspx* in the address bar if necessary

**License**

**Trial License**

For users to better evaluate our Document Imaging toolkit, activation of trail license is not

needed at user side. The trial license will be automatically expired 45 days after your first trial of

the SDK functions. You can then request an extension of another 30 days by contacting us at

*support@rasteredge.com.*

**Purchase License**

After you purchase the Developer License, we will send you a set of activated assemblies.

*For more information, please contact us at support@rasteredge.com.*

**Web Server License**

If you want to deploy your projects which use our dlls on a server, you should buy a server

license.

*For more information, please contact us at support@rasteredge.com.*

**XDoc. HTML5 Viewer**

**Overview**

RasterEdge XDoc.HTML5 Viewer adopts the latest HTML5 technology and owns strong

compatibility with most modern web browser environments. It is a JavaScript based image

viewing control that can be created on the client side without additional add-ins and

communicates directly with a WebHandler on the service side.

Product Features

 Allow to view, annotate, save, OCR, search and convert various document file formats.

 Flexible annotations enable collaboration at ease.

 Zero footprint viewer no addition plug-in is needed.

 Supported Tiff, PDF, Office Word, Excel, PPT, Dicom among other document and image

formats.

 Wide range of web browsers support including IE9+, FireFox, Chrome, Safari, and Opera

 Secured document displaying mechanism. The original document stay safe behind

protection in server, and only shallow copy of the documents is transmitted on the

network for displaying purpose.

 Cache is implemented to avoid repeated process and transmission of original document

files therefore maximizes the throughput of the system. The cached files are cleaned

automatically when not needed.

 White label webviewer control, seamless integration to your own project.

**Tutorial**

A complete example of XDoc.HTML5 Viewer is included in RasterEdge.DocImageSDK demo

projects in download package. You can create web document viewer with your own style. Please

see detailed tutorial from: http://www.rasteredge.com/how-to-csharp/xdoc/html5-viewer/.

**F&Q**

 **When configure IIS to run 500.19 error occurs, it may be caused by:**

1. Not registered the .net framework to the iis. (One of reasons: install a .net framework

before the installation of iis.)

2. The site configured in IIS has no permission to operate. (Modify permission)

Here are some solutions:

 In the command line input

cd C:\Windows\Microsoft.NET\Framework64\v2.0.50727

aspnet\_regiis.exe –i

 Right-click the correspond site-> Edit Permissions -> Security -> Group or user

names -> Edit -> Add -> Add Everyone usersgiven Full Control permissions.

 **If the uploading document is not successful, please pay attention to the following:**

1. Make sure the default page is Default.aspx. If not sure, in the address bar, add/Default.

aspx.

2. Don’t add SVGWebViewer dll to the project.

 **Q: In Asp.net project when testing web viewer control, I get the error:**

**Could not load file or assembly 'RasterEdge.Imaging.Annotation' or one of its dependenci**

**es. An attempt was made to load a program with an incorrect format.**

A: This is because Visual Studio uses a 32-bit web server internally. You can firstly debug

with x86 webviewer dlls in Visual Studio mimic IIS environment. Once done debugging with

x86 dlls, replace x86 dlls with x64 version of dlls to run in real IIS on server.

**WinForm Document Viewer**

**WinViewer Overview**

The WinViewer is a .NET imaging control used for Windows Forms applications, providing

professional document solutions for users and making images & documents viewing,

manipulation, annotation and saving an easy task. The WinViewer Control supports most

common document and image formats (PDF, MS Word, Excel, Ppt, Tiff, Png;).

A complete example of WinViewerDemo is included in the RasterEdge.DocImageSDK demo

projects in download package. This complete example is implementing default RasterEdge style,

if you want to create WinViewerDemo with your own style, please see our “Add a WinViewer to

Your Project” below this part for a step-by-step tutorial.

**Feature List**

 Perfectly work with Microsoft Visual Studio 2005 and above version

 Create high-quality document images. Repaint at zoom operations to maintain high

fidelity document imaging

 Automatically Generate thumbnails for document and provide navigation using

thumbnails viewer.

 Easy to view, edit, annotate and save document

 Zoom, pan, or select document page with mouse to fit width, height, or both

 Support tiff, pdf, office word, excel, ppt, medical image (dicom) and other commonly

used image formats.

**Control APIs**

The WinViewer control provides developers many APIs to be invoked. The following will give you

a broad overview of these functions.

**APIs**

**Description**

LoadFromFile(String

Accept the path of the file that you want to open.

filePath)

SaveFile(String filePath)

Save the file to a specified path.

UpPage

Scroll to previous visible page in the currently open document.

DownPage

Scroll to next visible page in the currently open document.

ZoomIn

Increase current zoom percentage of the WinViewer.

ZoomOut

Decrease current zoom percentage of the WinViewer.

FitWidth

Reset size of the currently displayed page.

FitHeight

ShowOneToOne

AddPage

Prior to the currently displayed page to add a blank page.

DeletePage

Delete the currently displayed page.

Roate90

Rotate the currently displayed page 90 degrees clockwise.

Rotate180

Rotate the currently displayed page 180 degrees clockwise.

Rotate270

Rotate the currently displayed page 90 degrees

counterclockwise.

DrawText, DrawFreehand

DrawLine, DrawPolygonLines

DrawFilledRectangle,

DrawRectangle,

Draw the specified type annotation on the page.

DrawHighLight

DrawEllipse, DrawPolygon

DrawRubberStamp

BurnAnnotation

Burn all annotations to the page.

DeleteAnnotation

Delete all selected annotations.

**Add a WinViewer to Your Project**

This part will guide you to add WinViewer control to your project and invoke APIs of this control.

See detailed steps below:

1. Setting Up Your Project

2. Open File From Specified Path

3. Draw Specified Type Annotation on Page

**Setting Up Your Project**

 In Visual Studio, create a new Windows Forms project called WinViewerDemo. If you already

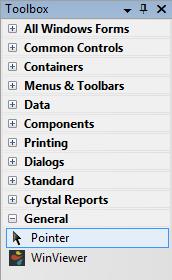
have a Windows Forms Project, this step can be omitted.

 Add control. Right-click on the Toolbox, select “Choose Items;”, locate the

**RasterEdge.DocImagSDK6.3\bin** in download package, and browse to find and select

**RasterEdge.Imaging.WinformsControl.DocumentViewer.dll**, then WinViewer Control will

appear in Toolbox. See as below:



Once

you

do

this,

you

will

find

a

new

reference

called

RasterEdge.Imaging.WinformsControl.DocumentViewer.dll has been added to your project.

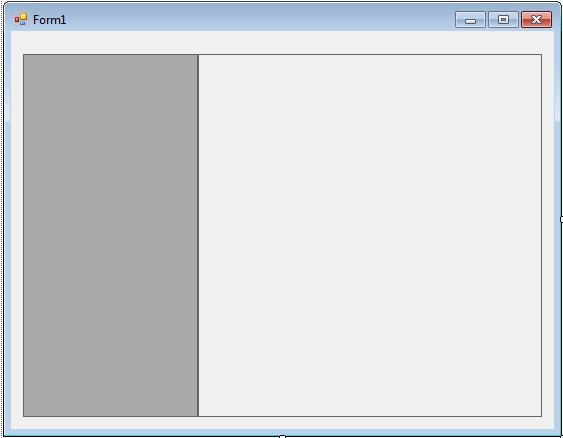
**Open File from Specified Path**

There may be other elements on your form, but in the following tutorial, we will take a blank

form as an example.

Drag a WinViewer control onto your WinForms application, and the form you created should

look similar, in the design-time view, to the screen capture shown below.



Now, you only need several lines of C# code to open a document.

The following code presents an open file dialog. You can select a file to be loaded into the

WinViewer control. If the file format is not supported by WinViewer control, there will prompt a

window “cannot open your file”.

private void OpenFile\_Click(objectsender, EventArgse)

{

OpenFileDialogofd = new OpenFileDialog();

ofd.Filter = "(\*.\*)|\*.\*";

ofd.Multiselect = false;

if (ofd.ShowDialog() == DialogResult.OK)

{

this.winViewer1.LoadFromFile(ofd.FileName);

}

}

**Draw Specified Annotation on Page**

Call a single method for each kind of annotation that you want to support from your toolbar

button’s OnClick events. Resizing, burning, deleting and moving are all built in.

private void HighLight\_Click(objectsender, EventArgse)

{

this.winViewer1.DrawHighLight();

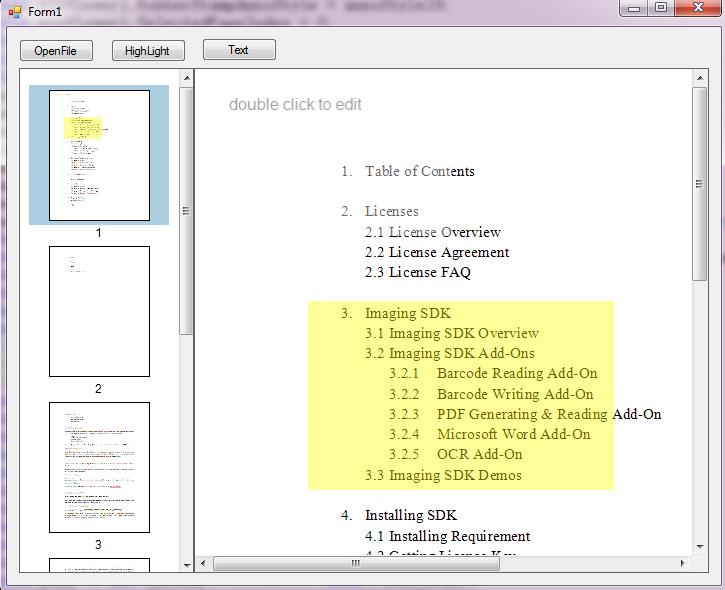
}

private void Text\_Click(objectsender, EventArgse)

{

this.winViewer1.DrawText();

}



Right-clicking on the created annotation, you will see the option to burn or delete. If you right-

click on the thumbnail, there is the option to “Add new page” or “delete page”. Using this

WinViewer, there’s necessary for you to add some buttons to complete these operations.

**Customize WinViewer Control**

**Customize Toolbar**

It’s easy to see from the above example. You can customize your toolbar, and the only thing you

need to do is call the appropriate API.

**Customize Annotation Style**

RasterEdge gives you default Annotation Style, but allows you to modify the style according to

your preferences.

Here we take the styles modification of HighLight annotation and Text annotation as examples.

private void HighLight\_Click(objectsender, EventArgse)

{

this.winViewer1.HighLightAnnoStyle.FillColor = Color.Red;

this.winViewer1.HighLightAnnoStyle.Transparency = 0.2f;

this.winViewer1.DrawHighLight();

}

private void Text\_Click(objectsender, EventArgse)

{

this.winViewer1.TextAnnoStyle.FillColor = Color.Yellow;

this.winViewer1.TextAnnoStyle.AnnoText = "double click to edit your content";

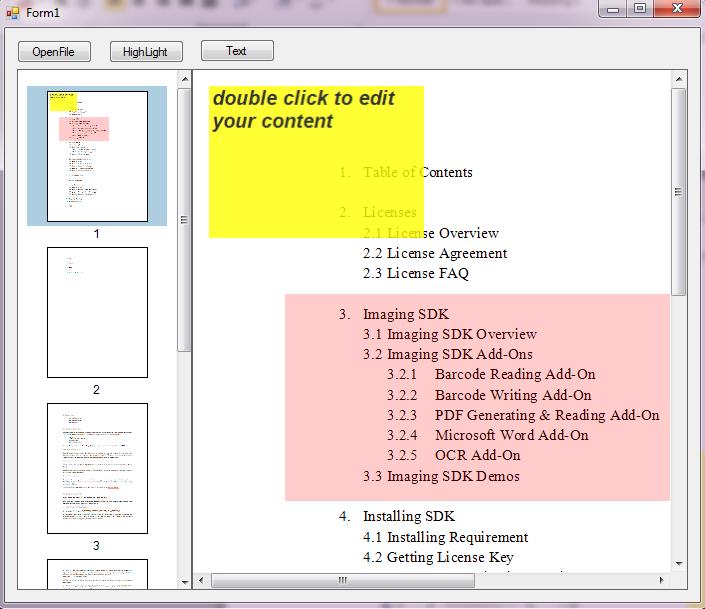
this.winViewer1.TextAnnoStyle.TextFont = new Font("Arial", 15, FontStyle.Bold |

FontStyle.Italic);

this.winViewer1.TextAnnoStyle.Transparency = 0.8f;

this.winViewer1.DrawText();

}



*Note: For each annotation, the attributes that can be modified have been listed in the Form1. If*

*not listed, after the annotation property changes, there is no corresponding effect.*

**Customize the Way Open and Store Files**

The WinViewer control offers developers two APIs to LoadFile. One is **LoadFromFile(string**

**filePath)**, and the other is **LoadFromFile(Stream stream)**. Developer can feel free to invoke

different APIs.

Here, RasterEdge just provides you sample code on how to load file from local and save file

locally. Certainly, you can customize the way to open and store files.

**Customize the Drop-down List**

There are several events in Form1, for example: OnFileAdded, PageIndexChanged and

SelectedIndexChanged, etc. These events are available to be used to customize the drop-down

list. If you don’t use the drop-down list, please omit these events.

**Events**

**Description**

OnFileAdded

Mainly in order to obtain the total number of document pages.

PageIndexChanged

Click on thumbnail to change the drop-down list displayed

value.

SelectedIndexChanged

According to the value of selected drop-down list to switch

pages.

Users can choose to copy different codes according to their needs for project.

**XDoc.Converter**

**Overview**

RasterEdge XDoc.Converter is an advanced .NET SDK for developing high performance

documents and images conversion applications. Using RasterEdge XDoc.Converter, you are

empowered to easily add mature and professional file conversion capabilities to your .NET

applications. This .NET file converter SDK supports various commonly used document and image

file formats, including Microsoft Office (2003 and 2007) Word, Excel, PowerPoint, PDF, Tiff,

Dicom, SVG, Jpeg, Png, Bmp, and Gif.

Product Features

 Built in C# managed code, completely integrated into .NET windows and web projects

 Provide robust and intuitive .NET APIs for easy and fast documents and images

conversion

 Support converting to PDF from MS Word, Excel, PowerPoint, Tiff, Dicom, and raster

images

 Capable of converting to Tiff file from PDF, Word, Excel, PowerPoint, Dicom, and raster

images

 Able to convert and change PDF, Word, Excel, PowerPoint, Tiff and Dicom to raster

images

 Allow .NET programmers to convert PDF document to Microsoft Word document

 Easy to deploy and distribute derived works that are compiled with RasterEdge

XDoc.Converter

**Tutorial**

.NET programming examples for all supported functions are provided online. Please see detailed

tutorial from: http://www.rasteredge.com/how-to-csharp/xdoc/converter/.

**XDoc.PDF**

**Overview**

RasterEdge XDoc.PDF is a professional .NET PDF solution that provides complete and advanced

PDF features. With simple integration, it can be used to enhance your .NET document imaging

application. To be more specific, we develop multiple robust and feasible APIs for implementing

high performance PDF manipulations, like PDF document editing, conversion, creation,

protection, compression, extraction, navigation, annotation, etc.

Product Features

 Quick to open or load PDF document from local file or byte array from database

 Easily create, merge, append, split, and save PDF document file(s)

 Allow to create, edit and manipulate PDF pages, like insert, delete, move, copy, paste,

rotate, etc.

 Rapidly render and convert PDF to/from MS Word, Excel, PowerPoint, Tiff, raster

images, .NET Graphics, REImage, SVG, and XPS

 Protect your PDF document with the help of password and digital signatures

 Support compressing your PDF file without sacrificing render quality

 Capable of editing PDF metadata, like searching, annotating, and saving metadata

 Empower to navigate PDF document content quickly via bookmark, outline, thumbnail,

and text search

 Able to add and edit various types of annotations on PDF document page

**Tutorial**

.NET programming examples for all supported functions are provided online. Please see detailed

tutorial from: http://www.rasteredge.com/how-to/csharp-imaging/pdf-overview/.

**Programming with RasterEdge Document Imaging SDK**

**Overview**

Introduction to RasterEdge Document Imaging

RasterEdge Document Imaging SDK features in reliable and straight forward APIs to load,

convert, annotate and save document file such as Office Word, Excel, PowerPoint, PDF, TIFF,

and DICOM. You can build projects which provide functions of viewing, annotating,

collaborating and managing documents at any time and any place. Developers can develop

either client software or online ASP.NET application with Rasteredge Document Imaging SDK.

You can find an online demo at http://www.rasteredge.com/dotnet-imaging/web-viewer-demo/.

The main features of our product are:

1. Load and create document formats including PDF, TIFF, Dicom, Word, Excel, and

PowerPoint.

2. Convert document pages to different image formats including raster images and vector

image file such as SVG files which can be viewed directly in web browsers. With

customized option such resolution and crop region.

3. Annotate on pages and save the annotation in native file format.

4. Page level processing for multi-page TIFF and PDF document including appending,

inserting, deleting and sorting pages.

5. Combine, split or extract pages from PDF or TIFF document.

6. Burn barcode in document page.

7. Scan barcode from documents.

8. Convert documents to PDF, TIFF and SVG file format.

In RasterEdge Document Imaging toolkit, a document and its page content within is represented

as a document class object in memory. These document classes of specific document types are

all derived from BaseDocument Class which contains common interfaces for document

operations. Similarly, Pages in document are represented as page objects derived from

BasePage Class.

For example, for PDF we use PDFDocument class object to reference a physical PDF document

file or stream. You can construct a PDFDocument using the following constructor.

PDFDocument doc = new PDFDocument(@"D:\sample1.pdf");

Or construct a word document.

DOCXDocument doc = new DOCXDocument(@"D:\BugList\sample1.docx");

You can get page from document object using the following code.

PDFPage page = doc.GetPage(0);

PDFDocument and DOCXDocument Class both implement APIs from BaseDocument, while

PDFPage and DOCXPage adapt APIs from BasePage.

**Register Process**

You need to add the following line of code in the entry point of your project to use functions in

the SDK.

WorkRegistry.Reset();

If you encounter Registry Exception when running the program, please refer to the

solutions here.

**TIFF**

**TIFF File Overview**

The Tagged Image File Format (known as TIFF) is a bitmapped and lossless image format, which

is often used to store large but high-quality images. RasterEdge Image provides the ability to

manipulate with TIFF file from various aspects using TIFFDocument and TIFFPage classes located

in RasterEdge.Imaging.TIFF namespace. And this part introduces functions of the TIFF assembly.

Here are the features supported by the TIFF assembly.

 Load or create TIFF file from tiff file, stream or an image collection

 Provide six types of tiff compression. CCIT 1D, Group4Fax, Group3Fax, LZW, PackBits,

JPEG.

 Retrieve and modify the IPTC, TIFF, EXIF filed and get the information of xml.

 Draw the specific part of page on a given graphics, scale operation is offered.

 Get preview of the first page for quick loading

 Convert TIFF file to bmp, gif, png, and jpeg image types

 Convert TIFF file to PDF

 Convert TIFF file to SVG file which can be displayed in web browsers.

 Add rich text and graphical annotations to TIFF file

 Rich APIs to process TIFF pages, like page extraction and adding image to page

 Add standard 1d & 2d barcodes to specified location of TIFF file

 Read barcodes information from Tiff file

**Requirements**

The APIs related to TIFF are located under:

**Assemblies: RasterEdge.Imaging.Basic.dll**

**RasterEdge.Imaging.TIFF.dll**

**NameSpace: RasterEdge.Imaging.Basic**

**RasterEdge.Imaging.TIFF**

**TIFF Programming Classes**

 **TIFFDocument**: This class is an extension of BaseDocument. It represents a high-level

model of the pages within a TIFF file. When a TIFFDocument object is created from an

existing TIFF file, it contains a collection of TIFFPage objects for each page within the file.

It is easy to use TIFFDocument to retrieving information from the files. You can also

reorder or remove pages from TIFF files.

 **TIFFPage**: This class uses BasePage as prototype. It is a high-level abstraction of a page

within a TIFF file. You obtain images from TIFFPage object individually.

**Construct, Load and Save a TIFFDocument**

**Load or Create a TIFFDocument**

You can load a Tiff file from a file path, stream or byte[].

APIs:

**TIFFDocument doc = new TIFFDocument(string filePath);**

**TIFFDocument doc = new TIFFDocument(Stream s);**

**TIFFDocument doc = new TIFFDocument(Byte[] byteArray);**

C#

using RasterEdge.Imaging.TIFF;

TIFFDocument doc = new TIFFDocument(filePath);

TIFFDocument doc = new TIFFDocument(stream);

TIFFDocument doc = new TIFFDocument(byteArray);

You can also create a TIFFDocument from Image Source. Like this

C#

public static void TestImageToTIFF()

{

//invoke this method only once at the beginning of your code register all assemblies

you referenced to your project

WorkRegistry.Reset();

//Construct image source from image files

REImage img1 = new REImage(@"c:\samplePNG.png", ImageType.PNG);

REImage img2 = new REImage(@"c:\samplePNGAnnotated1.jpg", ImageType.JPEG);

REImage[] imageSource = new REImage[2];

imageSource[0] = img1;

imageSource[1] = img2;

// create a TIFF document using image source, one image per page

TIFFDocument doc = new TIFFDocument(imageSource, TIFFCompression.JPEG);

doc.Save(@"c:\sample.TIFF");

}

See also REImage programming

Once you have a TIFFDocument object, you can get information about pages contained in this

document.

//get total page count in this document

int pageCount = doc.GetPageCount();

// get the first page of this document, note that the page index starts at 0

TIFFPage page = (TIFFPage)doc.GetPage(0);

**Save a TIFF File**

After a TIFF file has been read, created or modified, you can save it by invoking the save

methods in TIFFDocument class. This method takes in either an output Stream or an output

filePath.

Related APIs:

**void Save(string filePath);**

**byte[] SaveToBytes();**

**void SaveToStream(Stream stream);**

**Clear Temp files generated in the process**

Since tiff file might be too large to hold in memory, temp files are cached in the intermediate

process. Use the following API to clear the temp files when the tiff processing task is done. Once

this API is called, all cached files are deleted and you need to create new TIFFDocument object

to do further processing.

//clear temp files for processing this tiff file

// this code should be exectuted after alloperations on tiffsare done.

TIFFDocument.ClearTmpFiles();

**Get preview of Tiff Document**

RasterEdge document imaging sdk provide you with APIs to get a Bitmap of the first page in the

tiff file. You are able to get a preview of this tiff document without load and process the whole

document in memory.

Related APIs:

**static Bitmap GetPreviewImage(string file**， **Size targetSize)**

**static Bitmap GetPreviewImage(byte[] data,Size targetSize)**

**static Bitmap GetPreviewImage(Stream s,Size targetSize)**

**TIFF Rendering and Conversion**

RasterEdge XDoc.Converter supports converting TIFF document to/from various image or

document types with customized options. In details, you can convert TIFF document to PNG,

JPEG, BMP, and GIF image formats, as well as PDF file. In addition, TIFF document can be

converted from documents and images as well, like MS Word, Excel, PowerPoint, PDF, Dicom,

PNG,

JPEG,

BMP,

GIF,

etc.

Please

see

more

how

to

articles

online:

http://www.rasteredge.com/how-to-csharp/xdoc/converter/.

**TIFF Annotation**

You can add annotations like text and highlight on TIFF page. For detailed information about

annotation objects, please refer to Annotation section.

You need the following dll(s) to implement this feature.

**RasterEdge.Imaging.TIFF.dll**

**RasterEdge.Imaging.Annotation.dll**

**RasterEdge.Imaging.Basic.dll**

Sample code:

// generate the annotation object

AnnotationHandler anno = AnnotationGenerator.CreateLineAnnotation(new

RasterEdge.Imaging.Annotation.Basic.LinePoint(0, 0), new

RasterEdge.Imaging.Annotation.Basic.LinePoint(100, 100));

// create a line annotation starting at point (0.0) and end with point(100,100) note

only the relative position of the start and end point is used.

TIFFDocument doc = new TIFFDocument(@"c:\sample.tif");

TIFFPage page = (TIFFPage)doc.GetPage(0);

//add the line at point (100,100) on the tiff page. The position is measured at

default 96 resolution with respect to the tiff image.And therefore subject to the

change of the size of the tiff image

page.AddAnnotation(anno);

doc.Save(@"c:\annotatedSample.tif");

See also Annotations

**Editing TIFF Metadata**

You can get the tag collection of the specific tiff page, which is comprised of EXIF Fields, TIFF

Fields and IPTC Fields͘ Every EXIF and TIFF Field is made up of a “tag”, a “dataType” and a

“values”͘ The IPTC field is made up of a“ID”, a “Section” and a “value”͘

Sample code:

C#

public TagCollection GetTagCollection(int pageIndex, TIFFDocument doc)

{

return tifDoc.GetTagCollection(pageIndex);

}

Get the specific page’s XML information return as a string:

C#

public string GetXmpString(int pageIndex, TIFFDocument doc)

{

Return doc.GetXMPMetadata(0);//pageIndex = 0

}

**Add, Delete, Update and Find a TIFF Field**

Assume you have retrieved the tag collection already, then you can edit the TIFF Field, we

provide 8 constructors to create a TIFF Field, and you can choose the field’s data type easily.

*Note there are some important fields which can’t be removed. Otherwise the tiff file would be*

*corrupted. The TagID must be a hexadecimal.*

C#

TIFF Field Constructor:

public TIFFField(int tagID, byte value);

public TIFFField(int tagID, byte[] values);

public TIFFField(int tagID, string str);

public TIFFField(int tagID, uint value);

public TIFFField(int tagID, uint[] values);

public TIFFField(int tagID, ushort value);

public TIFFField(int tagID, ushort[] values);

public TIFFField(int tagID, uint frac, uint deno);

Add TIFF Field:

Note: the tagID must be hexadecimal and add a tiff field with an existing id is not permitted

void AddTifField(int pageIndex, TIFFDocument tifDoc, TagCollection tagCol)

{

tagCol.AddTifTag(new TIFFField(0x1111, 123));

tagCol.AddTifTag(new TIFFField(0x2222, new byte[3] { 0, 1, 2 }));

tagCol.AddTifTag(new TIFFField(0x3333, new ushort[3] { 0, 1, 2 }));

tagCol.AddTifTag(new TIFFField(0x4444, 3, 5));

tagCol.AddTifTag(new TIFFField((int)TIFFTag.Artist, "www.RasterEdge.com"));

}

Delete TIFF Field:

Remove a TIFF Field form the specific page.**Note: there are some significant fields can’t be removed, if so,**

**the document would be corrupted. The tagID must be a hexadecimal**

void DeleteTifField(int pageIndex, TIFFDocument tifDoc,TagCollection tagCol)

{

tagCol.DeleteTifTag(0x1111);//you can define a tag ID like this

tagCol.DeleteTifTag((int)TIFFTag.Artist);//or use our TIFFTag Enum

}

Update TIFF Field:

Find the field which tagID is 0x1111,and change it’s value to 456, find the field which tagID is

Artist,and change it’s value to”Welcome to RasterEdge”

void UpdateTifField(int pageIndex, TIFFDocument tifDoc,TagCollection tagCo)

{

tagCol.UpdateTifTag(new TIFFField(0x1111, 456));

tagCol.UpdateTifTag(new TIFFField((int)TIFFTag.Artist, "Welcome to RasterEdge"));

}

Find TIFF Field:

If there is a tiff field which tagID is Artist, then it willed be returned

TIFFField FindTifField(int pageIndex, TIFFDocument tifDoc, TagCollection tagCol)

{

return tagCol.FindTifTag((int)TIFFTag.Artist);

}

**Add, Delete, Update and Find a EXIF Field**

The manipulation of EXIF Fields is just as same as TIFF Fields

**Add, Delete, Update and Find a IPTC Field**

C#

IPTC Field Constructor:

public IPTCField(byte section, byte id, byte[] data);

public IPTCField(byte section, byte id, string str);

Add IPTC Field:

void AddIptcField(int pageIndex, TIFFDocument tifDoc, TagCollection tagCol)

{

tagCol.AddIPTCTag(new IPTCField(2, 2, new byte[3] { 0, 1, 2 }));

tagCol.AddIPTCTag(new IPTCField(2, 3, "www.RasterEdge.com"));

}

Delete IPTC Field:

void DeleteIptcField(int pageIndex, TIFFDocument tifDoc, TagCollection tagCol)

{

tagCol.DeleteIPTCTag(new IPTCField(2, 2, new byte[3] { 0, 1, 2 }));

tagCol.DeleteIPTCTag(new IPTCField(2, 3, "www.RasterEdge.com"));

}

Updata IPTC Field:

void UpdateIptcField(int pageIndex, TIFFDocument tifDoc, TagCollection tagCol)

{

//find theIPTC Field, which section is 2 and id is 2, then change it’s value to

byte[]{4,5,6}

//find the IPTC Field, which section is 2 and id is 2, then change it’s value to

“Welcome to RasterEdge”

tagCol.UpdateIPTCTag(new IPTCField(2, 2, new byte[3] { 4, 5, 6 }));

tagCol.UpdateIPTCTag(new IPTCField(2, 3, "Welcome to RasterEdge"));

}

Finde IPTC Field:

List<IPTCField>FindIptcField(int pageIndex, TIFFDocument tifDoc, TagCollection tagCol)

{

//if there is a iptc field, which section is 2 and id is 3, then it will be

returned

//note: a iptc field has the same section and id as another is permmited

return tagCol.FindIPTCTag(2,3);

}

**Get the XML Information**

The xml information is parsed from the tiff tag named XmpData.

C#

String GetXmpString(int pageIndex, TIFFDocument tifDoc)

{

return tifDoc.GetXMPMetadata(0);

}

**Editing TIFF Document**

**Re-Order Pages of TIFF File**

With TIFF processing dll, it is easy for you to re-order pages of a TIFF document as long as you

provide a new order sequence. See the sample code below:

C#

public void SortPages(String filePath, int[] pageOrder)

{

TIFFDocument doc = new TIFFDocument(filePath);

doc.SortPage(pageOrder);

}

**Swap two pages of TIFF File**

You can swap two pages of the tiff file, suppose you have a TIFFDocument object already.

C#

public void SwapPages(int pageIdxF, int pageIdxB, TIFFDocument tifDoc)

{

tifDoc.SwapTwoPages(pageIdxF, pageIdxB);

}

**Insert Pages to TIFF File**

You can also insert pages to TIFF document. Suppose you have a TIFFPage class object named

page, you can do something like this,

C#

public void InsertPageToFile(Stream s, int index, TIFFPage page)

{

TIFFDocument doc = new TIFFDocument(s);

doc.InsertPage(page, index);

}

**Delete Pages from TIFF File**

Now you can delete pages of the TIFF file, Input is page index list that you want to delete.

Suppose you have a TIFFDocument class object named document.

C#

public void DeletePageFromDocument(TIFFDocument document, int pageIdx)

{

document.DeletePage(pageIdx);

}

public void DeletePagesFromDocument(TIFFDocument document, int[] pageIdxs)

{

document.DeletePages(pageIdxs);

}

**Split TIFF File**

To split TIFF document, you can use TIFFDocument class. There is a static method called

SplitDocument, which takes either an input Stream or an input path, an input index and a list of

output paths or output Streams. For example, if the target file has eight pages and you input

two as the page index using SplitDocument, then the first three pages (index starts at 0) will be

included in one TIFF document and the rest of five pages will be included in the other TIFF file.

C#

public void splitTIFFDocument(string sourceFilePath, int pageIdx, string[] destnsPath)

{

TIFFDocument.SplitDocument(sourceFilePath, pageIdx, destnsPath);

}

**Combine TIFF File**

Using TIFFDocument class, you can also combine TIFF documents. There is a static method

called Combine, which takes either an output Stream or an output path and any number of

input paths or input Streams. The input TIFF documents are combined in order to create one

output TIFF document.

C#

public void CombineDocumentAndSaveItToFile(string[] docList, string destnFilePath)

{

TIFFDocument.CombineDocument(docList, destnFilePath);

}

**Generate Barcode on TIFF**

RasterEdge Barcode processing dll offers comprehensive functions for developers to generate

and design both 1d & 2d barcode images on TIFF file.

Sample Code:

//generate an code39 barcode

Linear linearBarcode = new Linear();

linearBarcode.Type = BarcodeType.CODE39;

linearBarcode.Data = "123456789";

linearBarcode.Resolution = 96;

linearBarcode.Rotate = Rotate.Rotate0;

// load tiff document, you can also load document like tiff, word, excel,ppt

TIFFDocument tiff = new TIFFDocument(@"c:\sample.tiff");

// get the first page

BasePage page = tiff.GetPage(0);

// generate reimage of this barcode

REImage barcodeImage = linearBarcode.ToImage();

//add barcode image to the first page

page.AddImage(barcodeImage, new System.Drawing.PointF(100f, 100f));

// save changes to the tiff

tiff.Save(@"c:\sample.tiff");

**Read Barcode Information from TIFF**

You can read barcode information from tiff document.

Sample code:

public static void ReadBarcodeFromTIFF(string filename, int pageIndex)

{

//generate tiff document

TIFFDocument doc = new TIFFDocument(filename);

//get the page you want to read barcode from

BasePage page = doc.GetPage(pageIndex);

//set reader setting

ReaderSettings setting = new ReaderSettings();

setting.AddTypesToRead(BarcodeType.Code39);

// read out barcode information

Barcode[] barcodes = BarcodeReader.ReadBarcodes(setting, page);

//output barcode information

foreach (Barcode barcode in barcodes)

{

Console.WriteLine(barcode.DataString);

}

}

**How to**

**How to View Thumbnail Image of a TIFF Document**

With RasterEdge document Imaging APIs, you can generate image out of document with proper

image size set. To view thumbnail images of a tiff document, you can set a smaller image size.

WorkRegistry.Reset();// invoke this static method only once at the beginning of codes

TIFFDocument doc = new TIFFDocument(@"c:\sample.tif");

string thumbnailDirectory = @"c:\thumbnail";

int pageCount = doc.GetPageCount();

for (int i = 0; i < pageCount; i++)

{

// get tiff page

TIFFPage page = (TIFFPage)doc.GetPage(i);

//generate thumbnail image of 50 pixel wide and 100 pixel high

REImage img = page.ConvertToImage(new Size(50, 100));

// save the thumnail image under the directory

img.Save(ImageType.PNG, thumbnailDirectory + @"\sample" + i + @".png");

}

**How to Generate Magnified Images of TIFFDocument**

You can set the zoom factor of the target image.

WorkRegistry.Reset();// invoke this static method only once at the beginning of codes

//load TIFF document

TIFFDocument doc = new TIFFDocument(@"c:\sample.tif");

// Set directory to store the zoomed image

string Image2X = @"c:\Sample2X";

// convert TIFF to png images with a zoom factor of 2f remember to set the factor to be a float

number

doc.ConvertToImages(ImageType.PNG, 2f, Image2X, "sample");

//or if you know the desired resolution to generate the image you can set the resolution

doc.ConvertToImages(ImageType.PNG, 192, Image2X, "sample");

Because in some cases you only need to show part of the image magnified, we provide you with

API to crop the TIFF image. This API costs fewer resources and less process time.

WorkRegistry.Reset();// invoke this static method only once at the beginning of codes

//load tiff page

TIFFDocument doc = new TIFFDocument(@"c:\sample.tif");

//get the first page of the tiff document

TIFFPage page = (TIFFPage)doc.GetPage(0);

// set the part of the original tiff image that you want to crop.

//The size and postion is measured at 96 resolution, which is the normal resolution for desktop

monitor

Rectangle sourceRect = new Rectangle(0,0,100,100);

// the target size of the REImage generated, note that the width and height is 2 times as large

as the source rectangle

// which means the source region is magnified by a factor of 2

Size targetSize = new Size(200,200);

// crop the desired region from the original TIFF image

REImage image =page.CropImage(sourceRect, targetSize);

image.Save(ImageType.PNG, @"c:\cropped.png");

**How to Add Watermark onto TIFF Document**

You can use annotation of Embedded Image, Text or RubberStamp to add annotation (Water

Mark) onto Tiff image.

Sample code:

// set annotation size

//the position of the annotation relative to its potential container

float x = 120.0f;

float y = 120.0f;

float width = 300.0f;

float height = 100.0f;

string text = "this is RubberStamp Water Mark";

Font font = new Font("Arial", 12.0F, FontStyle.Italic);

AnnotationBrush fontBrush = new AnnotationBrush();

fontBrush.FillType = FillType.Solid;

fontBrush.Solid\_Color = REColor.FromArgb(System.Drawing.Color.Blue.ToArgb());

RubberStampAnnotation anno = AnnotationGenerator.CreateRubberStampAnnotation(x, y, width, height,

text, font, fontBrush);

anno.OutLine = new LinePen();

anno.OutLine.Brush = new AnnotationBrush();

anno.OutLine.Brush.FillType = FillType.Solid;

anno.OutLine.Brush.Solid\_Color = new REColor(255, 10, 100, 100);

anno.OutLine.Width = 2.0f;

anno.Fill = new AnnotationBrush();

anno.Fill.FillType = FillType.Solid;

anno.Fill.Solid\_Color = new REColor(255, 20, 20, 20);

anno.CornerRadius = 1000f;

//set the overall transparency, this value has an overall impact on all

//colors related to this annotation. Note the default value is 1 which

// is complete transparency

anno.SetTransparency(0.4f);

// create a line annotation starting at point (0.0) and end with point(100,100) note only the

relative position of the start and end point is used.

TIFFDocument doc = new TIFFDocument(@"c:\sample.tif");

TIFFPage page = (TIFFPage)doc.GetPage(0);

//add the line at point (100,100) on the tiff page. The position is measured at default 96

resolution with respect to the tiff image.And therefore subject to the change of the size of the

tiff image

page.AddAnnotation( anno);

doc.Save(@"c:\annotatedSample.tif");

**Word**

**Introduction to Word Functions**

With RasterEdge.Imaging.MSWordDocx.dll, you can do the following.

 Load an existing Word(docx, doc) file

 Convert the pages of Word file to image

 Convert word file to PDF, TIFF or SVG file

 Annotate on word file, and save changes in native file format

Prerequisite for using these functions is to reference our Word processing dll

(RasterEdge.Imaging.MSWordDocx.dll) to you project.

**.docx and .doc file formats**

RasterEdge Word SDK support .docx and .doc word file formats.

.doc is the older format of the word files supported by 2003 or earlier version of MS Word

.docx is the spreadsheet supported by 2007 or later version of MS Word

Here is a table showing two file formats and their corresponding RasterEdge Programming Class.

.docx

.doc

DOCXDocument

DOCDocument

DOCXPage

DOCPage

Because two kinds of programming Classes inherit the same Supper Classes (BaseDocument and

BasePage), the APIs are identical. Thus all the demo code snippets in the following sections are

applicable to both docx and doc files. Please choose corresponding programming class with

respect to different file formats.

Note: To use APIs to handle .doc/xls/ppt files, you need to manually include zlibwapi.dll(in the

download package) in your APPDomain Path.

**Load and Save Word Document**

**About Word Programming Classes**

Word document assembly provides two Word document processing classes, which are

DOCXDocument and DOCXPage.

 DOCXDocument: This class refers to Word document file represented in memory and it

contains all document information of a Word file. It is an extension of BaseDocument.

 DOCXPage: This class refers to Word document page contained in DOCXDocument. It

uses BasePage as prototype.

**Word Document Object**

DOCXDocument represents the main structure of a document and the pages it contains. When a

DOCXDocument object is created, you can obtain information about the pages and other

document structures.

**Load Word document**

The example that follows shows how to load a Word file into DOCXDocument.

Generate a DOCXDocumentobject from stream.

DOCXDocument doc = new DOCXDocument(Stream s);

Or for file with .doc suffix

DOCDocument doc = new DOCDocument(Stream s);

Generate a DOCXDocument object from local file path.

DOCXDocumentdoc = new DOCXDocument(@"D: \sample1.docx");

Or for file with .doc suffix

DOCDocumentdoc = new DOCDocument(@"D: \sample1.doc");

Once you have created a document object, you can obtain information about this document,

and get page object out of it by calling **GetPage(int pageIndex).**

C#

public int GetPageCount(Stream s)

{

DOCXDocumentdocument = new DOCXDocument(s);

return document.GetPageCount();

}

**Save Word Document**

You can save DOCXDocument to file after modifications, like adding annotations on Word.

Related APIs:

**void Save(String fileName)**

**void SaveToStream(Stream stream)**

**byte[] SaveToBytes()**

**Get Preview of Word Document**

RasterEdge document imaging sdk provide you with APIs to get a Bitmap of the first page in the

word document file. You are able to get a preview of this word document without load and

process the entire document in memory.

Related APIs:

**static Bitmap GetPreviewImage(string file**， **Size targetSize)**

**static Bitmap GetPreviewImage(byte[] data,Size targetSize)**

**static Bitmap GetPreviewImage(Stream s,Size targetSize)**

**Annotations on Word Document**

You can add different types of annotations onto DOCXPage object and save the changes back

in .docx file format.

You can find an online annotation demo at http://www.rasteredge.com/dotnet-imaging/web-

viewer-demo/.

To use functions above, the required assemblies are:

**RasterEdge.Imaging.MSWordDocx.dll**

**RasterEdge.Imaging.Annotation.dll**

**RasterEdge.Imaging.Basic.dll**

Sample code:

// generate the annotation object

// create a line annotation starting at point (0.0) and end with point(100,100).

AnnotationHandler anno = AnnotationGenerator.CreateLineAnnotation(new

RasterEdge.Imaging.Annotation.Basic.LinePoint(0, 0), new

RasterEdge.Imaging.Annotation.Basic.LinePoint(100, 100));

// load Word document

DOCXDocument doc = new DOCXDocument(@"c:\sampleword.docx");

DOCXPage page = (DOCXPage)doc.GetPage(0);

//add the line at point (100,100) on Word page. The position is measured at //default 96

resolution with respect to word image.And therefore subject to the change of //the size of Word

image

page.AddAnnotation(anno);

// save the modified Word back to file

doc.Save(@"c:\annotatedSample.docx");

See also Annotation

**Word Rendering and Conversion**

RasterEdge XDoc.Converter supports converting Word document to various image and

document types with customized options. Besides, Word conversion from PDF is also supported.

To be specific, you can convert Word document to PNG, JPEG, BMP, and GIF image formats. You

can also convert Word document to other document types including TIFF, PDF and SVG.SVG is a

vector image format that is supported by HTML5 standard and therefore can be viewed at most

of the up to date web browsers. Please see more how to articles online:

http://www.rasteredge.com/how-to-csharp/xdoc/converter/.

**Create Barcode in Word**

RasterEdge Barcode processing dll offers comprehensive functions for developers to generate

and design both 1d & 2d barcode images on word file.

Sample Code:

public static void CreateBarcodeInWord()

{

//generate a code39 barcode

Linear linearBarcode = new Linear();

linearBarcode.Type = RasterEdge.Imaging.Barcode.Creator.BarcodeType.CODE39;

linearBarcode.Data = "123456789";

linearBarcode.Resolution = 96;

linearBarcode.Rotate = Rotate.Rotate0;

// load word document, you can also load document like tiff, pdf, excel, ppt

DOCXDocument doc = new DOCXDocument(@"c:\sample.docx");

// get the first page

BasePage page = doc.GetPage(0);

// generate reimage of this barcode

REImage barcodeImage = linearBarcode.ToImage();

//add barcode image to the first page

page.AddImage(barcodeImage, new System.Drawing.PointF(100f, 100f));

// save changes to the word

doc.Save(@"c:\sample.docx");

}

See Also Barcode Create

**Read Barcode from Word**

You can read barcode information from Word document.

Sample code:

public static void ReadBarcodeFromWord(string filename, int pageIndex)

{

//generate word document

DOCXDocument doc = new DOCXDocument(filename);

//get the page you want to read barcode from

BasePage page = doc.GetPage(pageIndex);

//set reader setting

ReaderSettings setting = new ReaderSettings();

setting.AddTypesToRead(RasterEdge.Imaging.Barcode.Scanner.BarcodeType.Code39);

// read out barcode information

Barcode[] barcodes = BarcodeReader.ReadBarcodes(setting, page);

//output barcode information

foreach (Barcode barcode in barcodes)

{

Console.WriteLine(barcode.DataString);

}

}

See also Barcode Read

**How to’s**

**How to: Create Thumbnail of DOCXDocument**

You can convert DOCXDocument to image files with a zoom factor suitable for thumbnail image.

// load word document

DOCXDocument doc = new DOCXDocument(@"c:\sample.docx");

// compute zoom out factor

// get the first page as sample page

DOCXPage page =(DOCXPage)doc.GetPage(0);

// the original height and width are measured at 96 dpi

float originalWidth = page.GetWidth()\*96;

// assume you want a thumbnail size of 500 pixel in width and compute the zoomfactor

float zoomFactor = 500f/originalWidth;

string directory = @"c:\Thumbnail";

// construct thumbnail image under the directory with a file name thumail01.png, thumnail02.png

doc.ConvertToImages(ImageType.PNG,zoomFactor,directory,"thumbnail");

**Excel**

**Introduction to Excel Functions**

With RasterEdge.Imaging.MSExcel.dll, programmers are provided with APIs to do the following:

 Load an existing Excel (xlsx and xls) file

 Convert the pages of Excel file to image

 Convert Excel file to PDF, TIFF or SVG file

 Annotate on Excel file, and save changes in native file format

Prerequisite for using these functions is to reference our Excel processing dll

(RasterEdge.Imaging.MSExcel.dll) to your project.

**.xlsx and .xls file formats**

RasterEdge Excel SDK support .xlsx and .xls Excel file formats.

.xls is the older format of the spreadsheet supported by 2003 or earlier version of MS Excel

.xlsx is the spreadsheet supported by 2007 or later version of MS Excel

Here is a table showing two file format and their corresponding RasterEdge Programming Class.

.XLSX

.XLS

XLSXDocument

XLSDocument

XLSXPage

XLSPage

Because two kinds of programming Classes inherit the same Supper Classes (BaseDocument and

BasePage), the APIs are identical. Thus all the demo code snippets in the following sections are

applicable to both xlsx and xls files. Please choose corresponding programming class with

respect to different file formats.

Note: To use APIs to handle .doc/xls/ppt files, you need to manually include zlibwapi.dll(in the

download package) in your APPDomain Path.

**About Excel Programming Classes**

Excel document assembly provides two document processing classes, which are

XLSXDocument/XLSDocument and XLSXPage/XLSPage.

 XLSXDocument/XLSDocument: This class refers to Excel document and it contains all

document information of Excel file. It is an extension of BaseDocument.

 XLSXPage: This class refers to Excel sheet contained in XLSXDocument/XLSDocument

and uses BasePage as prototype.

**Excel Document Object**

XLSXDocument/XLSDocument represents the main structure of an xls/xlsx file and the

sheet/pages it contains. When a XLSXDocument/XLSDocument object is created, we can easily

extract information about the pages/sheet and other document structures.

**Load Excel Document**

The example that follows shows how to load an Excel file into XLSXDocument/XLSDocument.

Generate an XLSXDocument/XLSDocument object from stream.

XLSXDocument doc = new XLSXDocument (Stream s);

XLSDocument doc = new XLSDocument (Stream s);

Generate an XLSXDocument/XLSDocument object from file path.

XLSXDocument doc = new XLSXDocument(@"D:\BugList\sample1.xlsx ");

XLSDocument doc = new XLSDocument(@"D:\BugList\sample1.xlsx ");

Once you have created a document object, you can obtain information about this document,

and get page object out of it by calling **GetPage(int pageIndex)**.

C#

public int GetPageCount(Stream s)

{

XLSXDocument document = new XLSXDocument(s);

return document.GetPageCount();

}

**Save Excel Document**

You can save XLSXDocument to file after modifications like adding annotations have been done.

Related APIs:

**void Save(String fileName)**

**void SaveToStream(Stream stream)**

**byte[] SaveToBytes()**

**Get preview of Excel Document**

RasterEdge excel-add on provide you with APIs to get a Bitmap of the first page/sheet in the

excel document file. You are able to get a preview of this excel document without load and

process the entire document in memory. Note that for excel sheet/page in large size, the image

may be cropped for better presentation.

Related APIs:

**static Bitmap GetPreviewImage(string file**， **Size targetSize)**

**static Bitmap GetPreviewImage(byte[] data,Size targetSize)**

**static Bitmap GetPreviewImage(Stream s,Size targetSize)**

**Annotations on Excel Document**

You can add different types of annotations onto XLSXPage object and save the changes back

in .xlsx native file format.

You can find an online annotation demo at http://www.rasteredge.com/dotnet-imaging/web-

viewer-demo/.

To use functions above, the required assemblies are:

**RasterEdge.Imaging.MSExcel.dll**

**RasterEdge.Imaging.Annotation.dll**

**RasterEdge.Imaging.Basic.dll**

Sample code:

// create a line annotation starting at point (0.0) and end with point(100,100).

AnnotationHandler anno = AnnotationGenerator.CreateLineAnnotation(new

RasterEdge.Imaging.Annotation.Basic.LinePoint(0, 0), new

RasterEdge.Imaging.Annotation.Basic.LinePoint(100, 100));

// load excel document

XLSXDocument doc = new XLSXDocument(@"c:\sample.xlsx ");

XLSXPage page = (XLSXPage)doc.GetPage(0);

//add the line at point (100,100) on the excel page. The position is measured at default 96

resolution with respect to the excel page image. And therefore subject to the change of the

size of the excel image

page.AddAnnotation(anno);

// save the modified excel back to file

doc.Save(@"c:\annotatedSample.xlsx");

**Excel Rendering and Conversion**

RasterEdge XDoc.Converter supports converting Excel document to various image and

document types with customized options. You can convert Excel document to PNG, JPEG, BMP,

and GIF image formats. You can also convert Excel document to other document types including

TIFF, PDF and SVG. SVG is a vector image format that is supported by HTML5 standard and

therefore can be viewed at most of the up to date web browsers. Please see more how to

articles online: http://www.rasteredge.com/how-to-csharp/xdoc/converter/.

**Create Barcode in Excel**

RasterEdge Barcode processing dll offers comprehensive functions for developers to generate

and design both 1d & 2d barcode images on Excel file.

Sample Code:

//generate a code39 barcode

Linear linearBarcode = new Linear();

linearBarcode.Type = RasterEdge.Imaging.Barcode.Creator.BarcodeType.CODE39;

linearBarcode.Data = "123456789";

linearBarcode.Resolution = 96;

linearBarcode.Rotate = Rotate.Rotate0;

// load excel document, you can also load document like tiff, word, excel,ppt

XLSXDocument doc = new XLSXDocument(@"c:\sample.xlsx");

// get the first page

BasePage page = doc.GetPage(0);

// generate reimage of this barcode

REImage barcodeImage = linearBarcode.ToImage();

//add barcode image to the first page

page.AddImage(barcodeImage, new System.Drawing.PointF(100f, 100f));

// save changes to the excel

doc.Save(@"c:\sample.xlsx");

See Also Barcode Create

**Read Barcode from Excel**

You can read barcode information from Excel document.

Sample code:

public static void ReadBarcodeFromExcel(string filename, int pageIndex)

{

//generate excel document

XLSXDocument doc = new XLSXDocument(filename);

//get the page you want to read barcode from

BasePage page = doc.GetPage(pageIndex);

//set reader setting

ReaderSettings setting = new ReaderSettings();

setting.AddTypesToRead(RasterEdge.Imaging.Barcode.Scanner.BarcodeType.Code39);

// read out barcode information

Barcode[] barcodes = BarcodeReader.ReadBarcodes(setting, page);

//output barcode information

foreach (Barcode barcode in barcodes)

{

Console.WriteLine(barcode.DataString);

}

}

See also Barcode read

**How to’s**

**How to: Create Thumbnail of Excel**

You can convert Excel document to image files with a zoom factor suitable for thumbnail image.

// load excel document

XLSXDocument doc = new XLSXDocument(@"c:\sample.xlsx");

// create thumbnailImage of target size 100\*130

// the original height and width are measured at 96 dpi

for (int i = 0; i < doc.GetPageCount(); i++)

{

BasePage page = doc.GetPage(i);

// create a thumbnail image that is cropped from the orignial image

REImage img = page.CropImage(new Rectangle(0, 0, 100, 130), new Size(100, 130));

img.Save(ImageType.PNG, @"c:\thumbnail" + i + ".png");

}

**PowerPoint**

**Introduction to PowerPoint Functions**

With RasterEdge.Imaging.MSPPT.dll, programmers are provided with APIs to do the following:

 Load an existing power point file (pptx, ppt)

 Convert the pages of PowerPoint file to images

 Convert PowerPoint file to PDF, TIFF or SVG file

 Annotate on PowerPoint file, and save changes in native file format

Prerequisite for using these functions is to reference our PowerPoint processing dll

(RasterEdge.Imaging. MSPPT.dll) to your project.

**.pptx and.ppt file formats**

RasterEdge Excel SDK support .pptx and .ppt Excel file formats.

.pptx is the older format of the PowerPoint file supported by 2003 or earlier version of

MS PowerPoint

.ppt is the PowerPoint supported by 2007 or later version of MS PowerPoint

Here is a table showing two file format and their corresponding RasterEdge Programming Class

.pptx

.ppt

PPTXDocument

PPTDocument

PPTXPage

PPTPage

Because two kinds of programming Classes inherit the same Supper Classes (BaseDocument and

BasePage) the APIs are identical. Thus all the demo code snippets in the following sections are

applicable to pptx and ppt files. Please choose corresponding programming class with respect to

different file formats.

Note: To use APIs to handle .doc/xls/ppt files, you need to manually include zlibwapi.dll(in the

download package) in your APPDomain Path.

**About PowerPoint Programming Classes**

PowerPoint document assembly provides two document processing classes, which are

PPTXDocument and PPTXPage.

 PPTXDocument: This class refers to PowerPoint document and it contains all document

information of PowerPoint file. It is an extension of BaseDocument.

 PPTXPage: This class refers to PowerPoint document page contained in PPTXDocument

and uses BasePage as prototype.

**PowerPoint Document Object**

PPTXDocument represents the main structure of a document and the pages it contains. When a

PPTXDocument object is created, we can easily extract information about the pages and other

document structures.

**Load PowerPoint Document**

The example that follows shows how to load a pptx file into PPTXDocument.

Generate a PPTXDocument object from stream.

PPTXDocument doc = new PPTXDocument (Stream s);

Generate a PPTXDocument object from file path.

PPTXDocument doc = new PPTXDocument (@"c:\sample.pptx");

or for file with .ppt suffix

PPTDocument doc = new PPTDocument (@"c:\sample.ppt");

Once you have created a document object, you can obtain information about this document,

and get page object out of it by calling **GetPage(int pageIndex)**.

C#

public int GetPageCount()

{

PPTXDocument doc = new PPTXDocument (@"c:\sample.pptx");

doc.GetPageCount();

}

**Save PowerPoint Document**

You can save PPTXDocument to file after modifications for example adding annotations have

been done.

Related APIs:

**void Save(String fileName)**

**void SaveToStream(Stream stream)**

**byte[] SaveToBytes()**

**Get Preview of PowerPoint Document**

RasterEdge powerpoint addon provides you with APIs to get a Bitmap of the first page/slide in

the powerpoint document file. You are able to get a preview of this powerpoint document

without load and process the entire document in memory.

Related APIs:

**static Bitmap GetPreviewImage(string file**， **Size targetSize)**

**static Bitmap GetPreviewImage(byte[] data,Size targetSize)**

**static Bitmap GetPreviewImage(Stream s,Size targetSize)**

**Annotations on PowerPoint Document**

You can add different types of annotations onto PPTXPage object and save the changes back

in .pptx native file format.

You can find an online annotation demo at http://www.rasteredge.com/dotnet-imaging/web-

viewer-demo/.

To usefunctions above, the required assemblies are:

**RasterEdge.Imaging.MSPPT.dll**

**RasterEdge.Imaging.Annotation.dll**

**RasterEdge.Imaging.Basic.dll**

Sample code:

// create a line annotation starting at point (0.0) and end with point(100,100).

AnnotationHandler anno = AnnotationGenerator.CreateLineAnnotation(new

RasterEdge.Imaging.Annotation.Basic.LinePoint(0, 0), new

RasterEdge.Imaging.Annotation.Basic.LinePoint(100, 100));

// load powerpoint document

PPTXDocument doc = new PPTXDocument(@"c:\sample.pptx");

PPTXPage page = (PPTXPage)doc.GetPage(0);

//add the line at point (100,100) on the powerpoint page.

//The position is measured at default 96 resolution with respect to the powerpoint page rendered

as image.

page.AddAnnotation(anno);

// save the modified pptx back to file

doc.Save(@"c:\annotatedSample.pptx");

**PowerPoint Rendering and Conversion**

RasterEdge XDoc.Converter supports converting PowerPoint document to various image and

document types with customized options. You can convert PowerPoint document to PNG, JPEG,

BMP, and GIF image formats. You can also convert PowerPoint document to other document

types including TIFF, PDF and SVG. SVG is a vector image format that is supported by HTML5

standard and therefore can be viewed at most of the up to date web browsers. Please see more

how to articles online: http://www.rasteredge.com/how-to-csharp/xdoc/converter/.

**Create Barcode in PowerPoint**

RasterEdge Barcode processing dll offers comprehensive functions for developers to generate

and design both 1d & 2d barcode images on power point file.

Sample Code:

// generate a code39 barcode

Linear linearBarcode = new Linear();

linearBarcode.Type = RasterEdge.Imaging.Barcode.Creator.BarcodeType.CODE39;

linearBarcode.Data = "123456789";

linearBarcode.Resolution = 96;

linearBarcode.Rotate = Rotate.Rotate0;

// load ppt document, you can also load document like tiff, word, excel,pdf

PPTXDocument doc = new PPTXDocument(@"c:\sample.pptx");

// get the first page

BasePage page = doc.GetPage(0);

// generate reimage of this barcode

REImage barcodeImage = linearBarcode.ToImage();

//add barcode image to the first page

page.AddImage(barcodeImage, new System.Drawing.PointF(100f, 100f));

// save changes to the powerpoint

doc.Save(@"c:\sample.pptx");

See Also Barcode Create

**Read Barcode from PowerPoint**

You can read barcode information from powerpoint document.

Sample code:

public static void ReadBarcodeFromPPT(string filename, int pageIndex)

{

//generate powerpoint document

PPTXDocument doc = new PPTXDocument(filename);

//get the page you want to read barcode from

BasePage page = doc.GetPage(pageIndex);

//set reader setting

ReaderSettings setting = new ReaderSettings();

setting.AddTypesToRead(RasterEdge.Imaging.Barcode.Scanner.BarcodeType.Code39);

// read out barcode information

Barcode[] barcodes = BarcodeReader.ReadBarcodes(setting, page);

//output barcode information

foreach (Barcode barcode in barcodes)

{

Console.WriteLine(barcode.DataString);

}

}

See also Barcode read

**How to’s**

**How to: Create Thumbnail of PowerPoint Document**

You can convert powerpoint document to image files with a zoom factor suitable for thumbnail

image.

// load Power Point document

PPTXDocument doc = new PPTXDocument(@"c:\sample.pptx");

// compute zoom out factor

// get the first page as sample page

PPTXPage page = (PPTXPage)doc.GetPage(0);

// the original height and width are measured at 96 dpi

float originalWidth = page.GetWidth() \* 96;

// assume you want a thumbnail size of 100 pixel in width and compute the zoomfactor

float zoomFactor = 100f / originalWidth;

string directory = @"c:\Thumbnail";

// construct thumbnail image under the directory with files name of the pattern thumail01.png,

thumnail02.png

doc.ConvertToImages(ImageType.PNG, zoomFactor, directory, "thumbnail");

**DICOM**

**DICOM Overview**

DICOM (Digital imaging and Communications in Medicine) is an international standard (ISO

12052) respective to medical imaging and its related workflow & data management. It defines

the file format that can be used for data exchange in clinical circumstances demand. Among

tens of thousands of medical imaging devices, DICOM is one of the most widely used medical

information standards. In 1993, ACR-NEMA Joint Commission released the third version of

DICOM standard (DICOM 3.0) which is regarded as the international standard in the field of

medical imaging informatics. It covers almost all information exchange protocols for medical

digital image, like image acquisition, archiving, communication, display and search. The

implementation of DICOM standard greatly **simplifies the exchange of medical image**

**information and promotes the research and development of** teleradiology system and image

management and communication system (PACS). In addition, openness and connectivity of

DICOM makes it possible to be integrated with other medical application systems (HIS, RIS).

**Programming with DICOM**

There are two important classes in DICOM programming, DCMDocument and DCMPage. Every

Dicom file is represented as a DCMDocument object, and the images and related information is

represented as DCMPage object.

**Load DICOM File**

You can load a file, stream or byte[] containing a Dicomdocument as a DCMDocument object.

DCMDocument doc = new DCMDocument(@"c:\a.dcm");

**DICOM Rendering and Conversion**

RasterEdge XDoc.Converter supports converting Dicom image to various image and document

types with customized options. You can convert Dicom image to PNG, JPEG, BMP, and GIF image

formats. You can also convert Dicom image to other document types including TIFF and PDF.

Please

see

more

how

to

articles

online:

http://www.rasteredge.com/how-to-

csharp/xdoc/converter/.

**JBIG2**

**JBIG2 Codec Overview**

RasterEdge Image JBIG2 codec can be used to decode and encode JBIG2 images using the

Microsoft .NET Framework. JBIG2 compression is an open standard and can compress bitonal

images 2-5 times more than the same image compressed with the industry standard TIFF CCIT

Group4 compression. This codec is available as a plug-in that integrates with RasterEdge Image

seamlessly.

**Feature List**

 Able to decode any page from a 1-bit JBIG2 image

 Encode a single or multi-page document as JBIG2 image(s) or image only PDF document

 Support lossless and lossy compressions

 Support encoding and decoding from any stream

 Read a specified region from an existing JBIG2 image stream

 Generate image only PDF document with embedded JBIG2 image(s)

 Integrated with RasterEdge PDFEncoder to encode PDF images with other compression

formats.

**How to Decode an JBIG2 Image**

Sample Code:

// invoke this method only once in the beginning of your code register all assemblies you

referenced to your project

WorkRegistry.Reset();

//decode image in jbig2 format

REImage img = newREImage(@"c:\sample.jb2", ImageType.JBIG2);

// save the image in png format

img.Save(ImageType.PNG, @"c:\sampleModified.png");

**JPEG 2000**

**JPEG 2000 Codec Overview**

RasterEdge JPEG 2000 codec can be used to decode and encode JPEG 2000 images using the

Microsoft .NET Framework. It uses wavelet compression technology to compress photo graphic

images further than any other available compression schemes. This codec is available as a plug-

in and integrates with RasterEdge seamlessly.

**Feature List**

 Able to decompress JPEG2000 images stored in any compatible jp2, j2k or code stream

 Support decoding JPEG2000 directly to 8-bit grayscale, 24-bit RGB, 16-bit grayscale, and

48-bit RGB

 Lossless compression

**How to Decode a JPEG 2000 Image**

Sample Code:

// invoke this method only once in the beginning of your code register all assemblies you

referenced to your project

WorkRegistry.Reset();

//decode image in jpeg2000 format

REImage img = new REImage(@"c:\sample.jpx", ImageType.JPEG2000);

// save the image in png format

img.Save(ImageType.PNG, @"c:\sampleModified.png");

**RasterEdge OCR**

RasterEdge OCR SDK provides you with the functions to recognize characters out of images and

documents types that are supported by RasterEdge Document Imaging SDK. Please see detailed

tutorial online: http://www.rasteredge.com/how-to/csharp-imaging/ocr-sdk/.

**RasterEdge RasterImage**

**RasterImage Overview**

RasterEdge Imaging SDK provides you functions to operate on over 100 raster image formats.

You can load, convert, process and save different image formats.

**Requirements**

The APIs related to RasterImage are located under:

**Assemblies: RasterEdge.Imaging.Raster.dll**

**NameSpace: RasterEdge.Imaging.Raster**

**About RasterImage Programming Classes**

 RasterImage Object: This class will provide more Image Processing methods.

 ConvertHandler: Provide several static methods for image conversion. (Support over

100 image formats).

 ImageProcess: Provide static methods for image processing, such as rotate, crop, resize,

trim, append, monochrome, quantizeColor, transformColorSpace, transformCompressin

and so on.

 LoadOption: To provide specific instructions when loading an image from file, byte array,

bitmap or from stream. If no load options are specified, Raster will use default values

when loading the image. More functions are under development and will be released

soon.

 SaveOption: Prior to saving an image, specify the image's save options, by setting the

save option’s properties. More functions are under development and will be released

soon.

**Convert Image**

RasterEdge XDoc.Converter provides reliable encoding and decoding functions for raster images

of different formats. With APIs provided, you can convert between different image formats with

ease. C# Sample Code:

WorkRegistry.Reset();

String inputFilePath = @"\*\*\*.\*\*\*";

String outputFilePath = @"\*\*\*";

// convert to a BMP

ImageConverter.ToImage(inputFilePath, outputFilePath + ".bmp", FileType.IMG\_BMP);

// convert to a GIF

ImageConverter.ToImage(inputFilePath, outputFilePath + ".gif", FileType.IMG\_GIF);

// convert to a JPEG

ImageConverter.ToImage(inputFilePath, outputFilePath + ".jpg", FileType.IMG\_JPEG);

// convert to a PNG

ImageConverter.ToImage(inputFilePath, outputFilePath + ".png", FileType.IMG\_PNG);

// convert to a SVG

ImageConverter.ToImage(inputFilePath, outputFilePath + ".svg", FileType.IMG\_SVG);

Related APIs:

**public static FileType GetImageType(String filePath)**

**public static FileType GetImageType(Stream fileStream)**

**public static ConvertResult ToImage(String srcFilePath, String desFilePath, FileType fileType)**

**public static ConvertResult ToImage(String srcFilePath, String desFilePath, ImageSaveOption**

**option)**

**public static ConvertResult ToImage(Stream srcStream, Stream desStream, FileType fileType)**

**public static ConvertResult ToImage(Stream srcStream, Stream desStream, ImageSaveOption**

**option)**

**public static ConvertResult ToDocument(String srcFilePath, String desFilePath, FileType**

**fileType)**

**public static ConvertResult ToDocument (String srcFilePath, String desFilePath,**

**DocumentSaveOption option)**

**public static ConvertResult ToDocument (Stream srcStream, Stream desStream, FileType**

**fileType)**

**public static ConvertResult ToDocument (Stream srcStream, Stream desStream,**

**DocumentSaveOption option)**

**Load Image**

Load image from byte array, stream and file path. Load option is provided to provide specific

instructions and it’s under development.

C#

**//Construct a RasterImage**

RasterImage raster = new RasterImage(@”C:\input.tif”);

**//Construct a RasterImage, Use LoadOption, Load the image as thumbnail**

LoadOption loadOption = new LoadOption();

**// not preserve the aspect ratio**

loadOption.MaintainAspectRatio = false;

loadOption.ThumbnailSize = ThumbnailSize.OneQuarter;

RasterImage raster = new RasterImage(@"F:\input.png", loadOption);

**//Construct a RasterImage, Use LoadOption**

LoadOption loadOption = new LoadOption();

**// not preserve the aspect ratio**

loadOption.MaintainAspectRatio = false;

**//resize image without antialias,it’s much faster than with antialias**

loadOption.LoadResizeAntiAlias = false;

**//When more than one of these operations is requested, the order is:**

**//Crop, Resize, Rotate**

loadOption.CropRectangle = new Rectangle(10, 10, 50, 50);

**//target image size**

loadOption.Resize = new Size(500, 500);

**//rotate target image**

loadOption.RotateAngle = RotateAngle.Rotate90;

RasterImage raster = new RasterImage(@"F:\input.png", loadOption);

Related APIs:

**public RasterImage(string imagePath);**

**public RasterImage(String imagePath, LoadOption loadOption)**

**public RasterImage(byte[] imageDataBytes);**

**public RasterImage(byte[] imageDataBytes, LoadOption loadOption)**

**public RasterImage(Stream imageStream);**

**public RasterImage(Stream imageStream,LoadOption loadOption)**

**public RasterImage(Bitmap bmp);**

**public RasterImage(Bitmap bmp,LoadOption loadOption)**

**Save Image**

Save image to byte array, stream and filePath. SaveOption is under development. Currently, only

TIFF and BMP is supported.

C# Sample Code:

**//Use saveOption to save file**

RasterImage raster = new RasterImage(@”C͗\input͘gif”);

SaveOption saveOption = new SaveOption();

**//Set the output file’s image format to tif**

saveOption.ImageFormat = REImageFormat.TIFF;

**//Set output.tif file’s colorSpace**

saveOption.Tiff.ColorSpace = REColorSpaceType.GRAYColorspace;

**//Set output.tif file’s compression**

saveOption.Tiff.Compression = RECompression.Group4Compression;

**//If multiPage = true, only save the first page to tif. Otherwise, save the whole file**

saveOption.Tiff.MultiPage = true;

**//Save the output.tif file**

raster.Save(@”C͗\output͘tif”,saveOption);

SampleCode2:

RasterImage raster = new RasterImage(@"F:\Anti.png");

SaveOption saveOption = new SaveOption();

saveOption .ImageFormat = REImageFormat.BMP;

**//RLECompression only support the bmp which colors less than 256**

saveOption .Bmp.Compression = RECompression.RLECompression;

raster.Save(@"F:\output.bmp",saveOption );

Related APIs:

**public byte[] SaveToBlob(REImageFormat format);**

**public byte[] SaveToBlob(SaveOption saveOption);**

**public void SaveToStream(Stream stream, REImageFormat format);**

**public void SaveToStream(Stream stream, SaveOption saveOption);**

**public void Save(string filePath);**

**public void Save(string filePath, SaveOption saveOption);**

**Image Process**

Using the static method, you can resize, flip, mirror, rotate, distort, shear and transform images,

adjust image colors, and apply various special effects.

C# Sample Code:

**//Reduce image’s color to specific num**

RasterImage raster = newRasterImage(@”C͗\input͘png”);

**//Use FloydSteinbergDitherMethod to reduce the color,and set the output file’s**

**colorSpace to Grayscale**

ImageProcess.QuantizeColor(raster, 256,

REDitherMethod.FloydSteinbergDitherMethod, REColorSpaceType.GRAYColorspace);

raster.Save(@”F͗\output͘png”)͖

**//Transform image’s color space**

RasterImage raster = new RasterImage(@"F:\input.tif");

ImageProcess.TransformColorspace(raster, REColorSpaceType.CMYKColorspace);

raster.Save(@"F:\output.tif");

**//Transform image’s compression**

RasterImage raster = new RasterImage(@"F:\input.tif");

ImageProcess.TransformCompression(raster, RECompression.JPEGCompression);

raster.Save(@"F:\output.tif");

**//BlurImage**

RasterImage raster = newRasterImage(@"F:\input.png");

**//Setting radius to 0, you will get an appropriative Gaussian radius;set the sigma to**

**0.7. The larger this value is, the more blurry the image would be.**

ImageProcess.BlurImages(raster,0,0.7);

raster.Save(@"F:\output.png");

**//Gammacorrect image**

RasterImage raster = new RasterImage(@"F:\24RGB\_0.png");

ImageProcess.GammaImages(raster,0.7);

raster.Save(@"F:\Test.tif");

**//Append image: direct = 0, append direction is Horizontal; direct = 1, append**

**direction is Vertical**

RasterImage raster = new RasterImage(@”C:\input.png”);

ImageProcess.AppendImages(raster, 1);

raster.Save(@”C:\output.png”);

Related APIs:

**public static void AddNoiseImages(RasterImage image, NoiseType noise);**

**public static void AdaptiveResizeImages(RasterImage imge, int width, int height);**

**public static void AppendImages(RasterImage image, int direction);**

**public static void BlurImages(RasterImage image, double radius, double sigma);**

**public static void CropImage(RasterImage image, Rectangle rect, int pageindex);**

**public static void FlipImages(RasterImage image);**

**public static void GammaImages(RasterImage image, double gamma);**

**public static void MonochromeImages(RasterImage image);**

**public static void QuantizeColor(RasterImage image, int num, REDitherMethod ditherMehod,**

**REColorSpaceType colorspace);**

**public static void RotateImages(RasterImage image, double degree);**

**public static void ReSizeImages(RasterImage imge, int width, int height, REInterpolateMethod**

**polateMethod);**

**public static void ShadowImages(RasterImage image, double percentOpacity, double sigma,**

**int x, int y);**

**public static void SharpenImages(RasterImage image, double radius, double sigma);**

**public static void SolarizeImages(RasterImage image, double factor);**

**public static void TrimImages(RasterImage image);**

**public static void TransformColorspace(RasterImage image, REColorSpaceType colorSpace);**

**public static void TransformCompression(RasterImage image, RECompression compression);**

**.....**

**Programming with Images**

**Overview**

RasetEdge Image toolkit is a powerful imaging solution for your desktop or web server side

application. With a collection of controls for ASP.NET and Windows, you can integrate a light

and powerful solution into your document imaging and image processing project. Licensing is

straightforward and runtime royalty free on desktop. All RasterEdge imaging related assemblies

are available as managed components and are natively built as .NET2.0 assemblies. Therefore, it

is compatible for .NET 2.0 and higher platforms. And there are several add-on modules to meet

your specific requirements.

RasterEdge Imaging basic

Including Codecs for basics image formats (Bitmap, GIF, PNG

assembly(Core SDK)

and JPEG) and image processing.

PDF SDK

Fast and powerful Codec, imaging, annotating and editing

for document of PDF format.

TIFF SDK

Fast and powerful Codec, imaging, annotating and editing

for document of TIFF format.

Barcode Reader SDK

Read barcode from an image (or document page) obtained.

Barcode Generator SDK

Write barcode to specific image (or document page).

DICOM Codec SDK

Codecs for DICOM image format.

JBIG2 Codec SDK

Codecs for JBIG2 image format.

JPEG2000 Codec SDK

Codecs for JPEG2000 image format.

**Image Concept**

**REImage, the Core Programming Class for Images**

In RasterEdge Imaging SDK we represent a raster image which is described in sample or pixel as

REImage. It is similar to the concept of Bitmap in .Net programming. Before discussing REImage

Class in details, we would like to provide you with some optional knowledge that may help you

understand the theory behind raster images. You can skip this section and learn REImage in

programming.

**Image Data**

Image data contains color information for every pixel in image. REImage provides a property of

ImageData from which you can use unsafe code to change/set image data directly.

**Image Compressions**

When you add images to certain file format, you may select different image compression

schemes to reduce file size.

**Why Compression**

In a raw state, images can occupy a rather large amount of memory both in RAM and in storage.

Image compression reduces the storage space required by an image and the bandwidth needed

when streaming that image across a network.

**Types of Compression**

Compression Types

There are two types of compression algorithms, namely lossless and lossy. Lossless compression

grants the integrity of data which means the decompressed image is the same as the original,

with no data lose. For lossy compression, some data in image will be lost during compression, so

the resulting image is not identical to original one.

CompressionMethods

The following compression methods are available in RasterEdge Image.JPEG Compression.

 Deflate/PNG Compression

 LZW Compression

 CCIT Group 4 / Group 3 Compression

 RLE(Run Length Encoding) Compression

 JPEG2000 Compression

 JBIG2 Compression

RasterEdge Imaging toolkit provides a number of codecs to encode and decode image in various

compression modes. Different document types may support different compression schemes.

**Image Codecs**

An Image codec is a program that can encode and decode an image in specific format.

REImage can read and write most common image formats. Images are read and written with

BaseDecoders and BaseEncoders for specific type, for example PNGEncoder and PNGDecoder

for encoding and decoding PNG images. Plug-ins for JPEG2000 and some other codecs are

available separately.

**Supported Formats**

REImage natively supports image (document) formats listed in the table below. This table also

shows the location of these codecs in particular assembly or plug-in.

**ImageType**

**ImageDecoder**

**ImageEncoder**

**Assembly**

Jpeg

JPEGDecoder

JPEGEncoder

RasterEdge.Imaging.Basic.codec

Png

PNGDecoder

PNGEncoder

RasterEdge.Imaging.Basic.codec

Bmp

BMPDecoder

BMPEncoder

RasterEdge.Imaging.Basic.codec

Gif

GifDecoder

GifEncoder

RasterEdge.Imaging.Basic.codec

JBIG2

JBIG2Decoder

JBIG2Encoder

RasterEdge.Imaging.Basic.codec

Jpeg2000

Jpeg2000Decoder JpegEncoder

RasterEdge.Imaging.JPEG2000

Dicom

DicomDecoder

DicomEncoder

RasterEdge.Imaging.DICOM

**REImage the Core Image Class in RasterEdge Imaging SDK**

**Introduction**

In RasterEdge Imaging SDK we represent raster image which is described in sample or pixel as

REImage. It is similar to the concept of Bitmap in.Net programming and you can get bitmap

object from a REImage object.

Rasteredge Imaging sdk provides straightforward APIs to load, modify, convert, and save

REImage object which encapsulate the concept of encoder or decoder. Below are example

codes for loading and saving REImage from/to various image formats. Please note that you can

make full use of REImage class when combined with functionalities from Document Imaging SDK.

One typical example is to convert document page to REImage. Try our online demo at

http://www.rasteredge.com/demo/online-document-viewer/ for more detailed information.

*Note: RasterEdge Document Imaging SDK represents PDF, TIFF, Dicom, Office Word, Excel, and*

*PowerPoint as document object. You can get REImage from BaseDocument and BasePage. See*

*Document Imaging for more Information.*

**Requirements**

The APIs related to REImage are located under:

**Assemblies: RasterEdge.Imaging.Basic.dll**

**NameSpace: RasterEdge.Imaging.Basic;**

**How to Create REImage**

**Create REImage from Image File, Stream & Byte Array**

Create REImage from image file, stream & byte array that contain the image. Sample APIs in

REImage Class are listed below.

**public REImage(String fileName, ImageType sourceType)**

**public REImage(Stream stream, ImageType sourceType)**

**public REImage(byte[] fileData,ImageType sourceType)**

The parameter type ImageType can be chosen for any supported image format mentioned in

previous section. You can also let your program choose the ImageType automatically using

similar API as follows:

**public REImage(String filename)**

Demo code to construct an image source:

List<REImage> tmpImgList = new List<REImage>();

DirectoryInfo d = new DirectoryInfo(filepath);

foreach (FileInfo file in d.GetFiles())

{

// construct image source from jpeg and png images

if (file.Extension.ToLower().Contains("jpeg") ||

file.Extension.ToLower().Contains("png"))

{

REImage img = new REImage(file.FullName);

tmpImgList.Add(img);

}

}

REImage[] ImageSource = tmpImgList.ToArray();

**Create Image from Bitmap**

**publicREImage (Bitmap image)**

**Annotate on REImage**

You can draw different kinds of predefined annotations onto REImages, and save them to file or

stream. Below is the sample code:

publicstaticvoid TestREImageAnnotation()

{

// invoke this method only once at the start of your project to register all

assemblies you referenced to your project

WorkRegistry.Reset();

//Load an PNG file to REImage object

REImage img = new REImage(@"c:\samplePNG.png", ImageType.PNG);

// create a line annotation starting at point (0.0) and end with point(100,100), note

that only the relative position of the start and end point is used.

LineAnnotation anno = AnnotationGenerator.CreateLineAnnotation(new LinePoint(0, 0),

new LinePoint(100, 100));

// set line stroke color and style

anno.LinePen = PenGenerator.CreateLinePen(new REColor(255, 0, 0, 255), 5.0F);

anno.LinePen.StartCap = new LineCap();

anno.LinePen.StartCap.Cap\_Width = 2.0f;

anno.LinePen.StartCap.CapStyle = LineCapStyle.Diamond;

anno.SetTransparency(0.4f);

// draw line annotation on the REImage, The position is measured at default 96

resolution with respect to the image.And therefore it is subject to the change of the

size of the tiff image

img.DrawAnnotation(new Point(100, 100), anno);

img.Save(ImageType.PNG, @"c:\samplePNGAnnotated.png");

}

See also Annotation

**Save REImage**

You can save REImage to file, stream and byte array with specified image type. Useful APIs in

REImage Class:

**void Save(ImageType targetType, String fileName)**

**void Save(ImageType targetType, Stream stream**）

**byte[] Save(ImageType targetType)**

**ImageProcessing**

RasterEdge Imaging provides you with the ability to perform operations on existing REImage

object to create a new REImage object. These functions are located under

RasterEdge.Imaging.Processing Assmebly. Generally speaking, there are three kinds of image

processing operations available for you.

Category

Illustration

Channels Processing

Operate on images with multiple components, like color images

Effects Processing

Perform visual effects on images like mosaic or beveling

Filters

Perform mathematical filtering like high or low pass filtering

Transforms

Perform coordinate transforms or depth transforms like rotate or

ripple

APIs to perform these operations are in the ImageProcessing Class located in

RasterEdge.Imaging.Processing assembly.

**Annotations**

**Introduction to Annotations**

RasterEdge Annotation is a managed .NET Assembly that can perform annotation capabilities to

mark, draw, and visualize objects on an image or document. Annotation objects include

primitive shapes (lines, rectangle, polygon and ellipse), text, callout, rubber stamp, hotspot,

freehand, signature, arrow, embedded images and hot spots. You can set properties of these

annotations, and add these annotations onto REImage and Document.

With Windows Forms viewer control or ASP.NET AJAX driven Webviewer, these annotations can

be independently resized, moved, rotated, and placed on different layers. Annotation can be

imported

or

exported

from/to

an

xml

file.

Try

our

online

Demo

at

http://www.rasteredge.com/demo/online-document-viewer/ for more detailed information.

Features of Annotation Assembly

 Draw an arbitrary number of annotation objects to an image or document

 Object Oriented Design for every annotation object

 Annotation objects can be moved, resized, and rotated independently from image or

document

 GDI+ graphics allows any object to be rendered at variable transparency

 Save or load annotations as a separate XML file

 Annotations can be rotated along with image in 90 degree increments

 Individual points from annotations supporting points (Freehand, Polygon, etc.) can be

repositioned

 Able to change shape of annotation object

 Annotations can be burned onto image with a single method

 Render annotation object to REImage for further operation

 Various properties can be set when creating an annotation object

Following is a list of supported annotations by RasterEdge Annotation assembly.

 Rectangle

 Ellipse

 Line

 Freehand

 Freehand Lines

 Text

 Rectangular HotSpot

 Freehand HotSpot

 Embedded Image

 Referenced Image

 Polygon

 Lines

 RubberStamp

 CalloutAnnotation

 ArrowAnnotation

 SignatureAnnotation

**Requirements**

Assemblies you need:

**RasterEdge.Imaging.Annotation.dll**

**RasterEdge.Imaging.Basic.dll**

Namespace:

using RasterEdge.Imaging.Annotation;

using RasterEdge.Imaging.Basic;

**Generate an AnnotationObject**

You can generate an Annotation object programmatically without using a graphics interface.

When generating an annotation, a number of properties can be set. Following is an example of

generating a hotspot annotation object.

*Note: In RasterEdge Imaging, Annotation is represented as an Annotation object derived from*

*the supper class AnnotaionBasic. Since some APIs are inherited from this super class, a type*

*conversion may be needed.*

First, add reference to RasterEdge.Imaging.Annotation.

Example Code:

using RasterEdge.Imaging.Annotation

**How to Generate an Annotation Object Programmatically**

C#

// set annotation size

// set xand y coordinate of the right top vertex of this annotation rectangle

// the actual shape ofrubberStamp is contained in this rectangle

float x = 120.0f;

float y = 120.0f;

float width = 300.0f;

float height = 100.0f;

string text = "this is RubberStamp annotation";

Font font = new Font("Arial", 12.0F, FontStyle.Italic);

AnnotationBrush fontBrush = new AnnotationBrush();

fontBrush.FillType = FillType.Solid;

fontBrush.Solid\_Color = REColor.FromArgb(System.Drawing.Color.Blue.ToArgb());

RubberStampAnnotation anno = AnnotationGenerator.CreateRubberStampAnnotation(x, y, width, height,

text, font, fontBrush);

anno.OutLine = new LinePen();

anno.OutLine.Brush = new AnnotationBrush();

anno.OutLine.Brush.FillType = FillType.Solid;

anno.OutLine.Brush.Solid\_Color = new REColor(255, 10, 100, 100);

anno.OutLine.Width = 2.0f;

anno.Fill = new AnnotationBrush();

anno.Fill.FillType = FillType.Solid;

anno.Fill.Solid\_Color = new REColor(255, 20, 20, 20);

anno.CornerRadius = 1000f;

//set the overall transparency. This value has an overall impact on all

//colors related to this annotation. Note, the default value is 1 which

// is complete transparency

anno.SetTransparency(0.4f);

**Burn Annotation to Document or Image**

Once you generate an Annotation object, you can add and burn it onto documents like PDF, TIFF

and Word, and then save it in native file format. You can also burn annotation on to documents

using REImage objects.

**How to Burn Annotation Object to Document (PDF, TIFF, WORD, EXECEL, PPT)**

Example Code:

public static void TestPDFAddAnnotation()

{

//use code in previous section to generate a rubber starmp annotaion

RubberStampAnnotation anno = TestAnnotation.addRubberStampAnnotation();

//load an PDF Document

PDFDocument doc = new PDFDocument(@"c:\REImage.pdf");

// add this annotation in the first page

doc.GetPage(0).AddAnnotation(new PointF(100f, 100f), anno);

// save this annotated PDF to file

doc.Save(@"c:\annotated.pdf");

}

public static void TestWordAnnotation()

{

//use code in previous section to generate a rubber starmp annotaion

RubberStampAnnotation anno = TestAnnotation.addRubberStampAnnotation();

//load an Word Document

DOCXDocument doc = new DOCXDocument(@"c:\TestWord.docx");

Console.WriteLine(doc.GetPageCount());

// add this annotation in the first page

doc.GetPage(0).AddAnnotation(new PointF(100f, 100f), anno);

// save this annotated Word to file

doc.Save(@"c:\annotated.docx");

}

public static void TestTIFFAnnotation()

{

//use code in previous section to generate a rubber starmp annotaion

RubberStampAnnotation anno = TestAnnotation.addRubberStampAnnotation();

//load a Tiff Document

TIFFDocument doc = new TIFFDocument(@"c:\sampleTiff.tiff");

Console.WriteLine(doc.GetPageCount());

// add this annotation in the first page of the Tiff document

doc.GetPage(0).AddAnnotation(new PointF(100f, 100f), anno);

// save this annotated Tiff to file

doc.Save(@"c:\annotated.Tiff");

}

See Also PDF Annotation

See Also TIFF Annotation

See Also WORD Annotation

**How to Burn Annotations on Images**

Once you get a REImage object and an annotation object, you can draw annotations onto it and

save it to different image types.

Example Code:

public static void TestREImageAnnotation()

{

// invoke this method only once at the beginning of your code to register all assemblies

you referenced to your project

WorkRegistry.Reset();

// load an PNG file to REImage object

REImage img = new REImage(@"c:\samplePNG.png", ImageType.PNG);

// create a line annotation starting at point (0.0) and ending with point(100,100).

Note that, only the relative position of the start and end point is used

LineAnnotation anno = AnnotationGenerator.CreateLineAnnotation(new LinePoint(0, 0),

new LinePoint(100, 100));

// set line stroke color and style

anno.LinePen = PenGenerator.CreateLinePen(new REColor(255, 0, 0, 255), 5.0F);

anno.LinePen.StartCap = new LineCap();

anno.LinePen.StartCap.Cap\_Width = 2.0f;

anno.LinePen.StartCap.CapStyle = LineCapStyle.Diamond;

anno.SetTransparency(0.4f);

// draw line annotation on the REImage. The position is measured at default 96

resolution with respect to the image. Therefore, it is subject to the change of the size

of the Tiff image

img.DrawAnnotation(new Point(100, 100), anno);

img.Save(ImageType.PNG, @"c:\samplePNGAnnotated.png");

}

**Annotations on ASP.NET DocumentViewer or Windows Form DocumentViewer**

Using our ASP.NET or Windows Form Document Viewer, you can annotate on a document with

graphics interface. RasterEdge.Imaging.WebViewer assembly contains web controls that you can

embed in your ASP.NET project by copying a couple of lines of JavaScript. Try our online demo at

http://www.rasteredge.com/demo/online-document-viewer/ for more detailed information. For

WinForm project, there are controls located in RasterEdge.Imaging.WinControl assembly and

you can add to your own project.

**Annotation Assemblies**

To implement full functions of RasterEdge Imaging Annotation, three assemblies are needed.

Assembly

Description

RasterEdge.Imaging.Annotation.dll

Annotation Classes, all you need to generate an

Annotation object programmatically.

RasterEdge.Imaging.WebViewer.dll

Include Web controls that you can integrate into

your own ASP.NET project. You can draw, add, and

burn annotations to document or image using

graphics interface powered by JavaScript.

RasterEdge.Imaging.WinControl.dll

Include Windows Form viewer controls that you can

integrate into your own WinForm project. You can

draw, edit, and burn annotations to document or

image using graphics interface provided in these

viewer controls.

See Also Web Viewer

**Metadata**

BaseMetadata Class

This class can be used as a collection of Metadata. For detailed usage, please see corresponding

Metadata SampleCode.

BaseMetadataItem class

As an abstract class, it is used to record the implementation of each element of Metadata. And

each Metadata type will be illustrated in detail.

There’s a limitation to the supportive Metadata types. You may use BaseMetadataType to check

whether your desired Metadata is supported or not.

**Introduction to Metadata**

Metadata is data that describes other data. RasterEdge Imaging allow viewing and manipulation

of metadata stored in an image.

The RasterEdge.Imaging.Codec namespace contains classes that handle image metadata.

Metadata is a convenient way to store textual information in an image. RasterEdge Image allows

this information to be accessed and manipulated. For example, it is possible to store the

metadata information in a database, build a metadata viewer application, and to add your own

metadata in the form of EXIF, IPTC, XMP, or COM markers. See the Metadata Demo installed

with RasterEdge Image for an example of metadata use.

**Supported Metadata Types**

RasterEdge Image supports the following metadata types.

 EXIF tags

 IIM(IPTC)

 XMP data

 TIFF Tags

In JPEG images, metadata is stored in "APPn markers". EXIF information is stored in an"APP1

marker", and IPTC and Photoshop Resource information is stored in an "APP13"marker. These

markers are created automatically when a JPEG image is encoded. Alternatively, you can use a

method to copy metadata without re-compressing JPEG images.

**Image Formats Supporting Metadata**

The following Image Formats support Metadata.

**Metadata**

**Operations**

**Image Format**

**types**

EXIF

JPEG/Exif/TIFF

XMP

GIF89a/JPEG/JPEG2000/PNG/TIFF/ProstScript/SVG/PDF/

Html/DNG/Adobe Illustrator

IIM(IPTC)

JPEG/Exif/TIFF/Jpeg2000/PNG

**EXIF Metadata**

**Parse Exif Metadata from TIFF File**

Exchangeable image file format (officially Exif, not EXIF according to JEIDA/JEITA/CIPA

specifications) is a standard that specifies the formats for images, sound, and ancillary tags used

by digital cameras (including smartphones), scanners and other systems handling image and

sound files recorded by digital cameras.

**Embed Exif to TIFF**

So far, embedding Exif information into TIFF file is supported. If you have a TIFFDocument object

and want to embed Exif information, you may do as follows.

**Parse & Update Exif from File**

Support parsing Exif Tag from TIFF file and modifying Exif value. We represent Exif Tag as

EXIFDefine. For more Tag information, please refer to the International Standard Exif 2.0 or

visithttp://www.awaresystems.be/imaging/tiff/tifftags/privateifd/exif.html.

You can get the accurate Exif information from a TIFDocument with the following method.

EXIFMetadata exif = (EXIFMetadata)doc.GetEXIFMetadata(1);//Get EXIF information of the

first TIF page

Console.WriteLine(metadata[0].ExifItemValue.Values[0]);//Have access to specific Tag

value

Below is the method to add an EXIF.

TIFFTag tag = (TIFFTag)0x8888;

TIFFField field = new TIFFField(tag, TIFFDataType.ASCII);

Field.Value.Add(new TIFFAscii(“RasterEdge”);

EXIFMetadataItem item = new EXIFMetadataItem(field);

Metadata.Add(item);

Doc.AddEXIFMetadataItem(item, 1);

Then, EXIF information is successfully embedded into TIFF file (more image formats are also

supported). You can freely customize your own data as well. Certainly, the deletion of EXIF is

easy to achieve.

**XMP**

Extract XMP from supportive file format as Stream or byte[] array.

We currently support getting XMP data from TIFF, TIFFEP, and DNG files, and you can use the

following code.

TIFFDoucment doc = (TIFFDocument)REFile.OpenDocuemntFile(@“1.tif”, new TIFDecoder());

String xmp = doc.GetXMPMetadata(0);//0 represents XMPData of the first page

Another method:

Byte[] xmp = doc. GetXMPMetadataAsArray(1);

**Barcode Read**

Developed as powerful linear & 2d barcodes recognition SDK, this Barcode Reader Add-on can

be easily combined with .NET Imaging SDK and available in any .NET applications. It supports

detecting and reading barcodes from .NET project images or documents. More than 20 linear

and 2d barcodes are supported.

**Linear Barcodes**

**2DBarcodes**

Australia Post

EAN-13

ISSN

POSTNET

Data Matrix

Codabar

Identcode

ITF-14

RM4SCC

PDF-417

Code 39

Intelligent Mail

Leitcode

UPC-A

QR Code

Code 128

Interleaved 2 of 5

Patch Code

UPC-E

EAN-8

ISBN

PLANET

If you want to test our Barcode Reader Add-on in .NET, C#, VB.NET applications, please firstly do

as follows:

 Integrate package dlls into Visual Studio (2005 or later) project by adding reference;

In all, you need these dlls: RasterEdge.Imaging.Basic.dll,

RasterEdge.Imaging.Basic.Codec.dll, and RasterEdge.Imaging.Barcode.Scanner.dll.

 Now, you may easily read barcodes in C# and VB.NET Classes.

**How to’s**

**How to: Read Barcode from Image**

To read barcode from image file, you may directly refer to the following demo codes in C# and

VB.NET. In details, codes below allow you to load an image file containing barcode into a

REImage object and read barcode information out of that REImage object. Also, you can set and

scan a specific region of REImage. If you want to read a barcode image file, our barcode reader

add-on can also help with that.

In C# Class

Demo Code:

public static void ReadBarcodeFromImage(string filename)

{

// invoke this static method only once at the beginning of codes

WorkRegistry.Reset();

//load an image containing the barcode image on it.

REImage reImage = new REImage(filename);

// set reader setting

ReaderSettings setting = new ReaderSettings();

// read qr code

setting.AddTypesToRead(BarcodeType.QRCode);

// read also code128

setting.AddTypesToRead(BarcodeType.Code128);

// read image

Barcode[] barcodes = BarcodeReader.ReadBarcodes(setting, reImage);

foreach (Barcode barcode in barcodes)

{

//output barcode data onto the screen

Console.WriteLine(barcode.DataString);

}

}

In VB.NET Class

Demo Code:

Public Shared Sub ReadBarcodeFromImage(filename As String)

WorkRegistry.Reset()

' invoke this static method only once at the beginning of codes

'load an image containing the barcode image on it.

Dim reImage As New REImage(filename)

' set reader setting

Dim setting As New ReaderSettings()

' read qr code

setting.AddTypesToRead(BarcodeType.QRCode)

' read also code128

setting.AddTypesToRead(BarcodeType.Code128)

' read image

Dim barcodes As Barcode() = BarcodeReader.ReadBarcodes(setting, reImage)

For Each barcode As Barcode In barcodes

' output barcode data onto the screen

Console.WriteLine(barcode.DataString)

Next

End Sub

**How to: Read Barcode from Document**

Besides images, RasterEdge .NET barcode reader add-on also empowers you to read barcodes

from several document files, like TIFF, PDF, Word and Excel. In the following parts, you will get

the demo codes for barcode reading from these documents. Please note that, for each

document barcode scanning, you also need to add corresponding dll into your project, including

RasterEdge.Imaging.TIFF.dll, RasterEdge.Imaging.PDF.dll, RasterEdge.Imaging.MSWordDocx.dll,

and RasterEdge.Imaging.MSExcel.dll.

In C# Class

Demo Code: For Barcode Reading from PDF

public static void ReadBarcodeFromPdf(string filename, int pageIndex)

{

// invoke this static method only once at the beginning of codes

WorkRegistry.Reset();

//generate pdf document

BaseDocument doc = new PDFDocument(filename);

//get the page you want to read barcode from

BasePage page = doc.GetPage(pageIndex);

//set reader setting

ReaderSettings setting = new ReaderSettings();

setting.AddTypesToRead(BarcodeType.Code39);

// read out barcode information

Barcode[] barcodes = BarcodeReader.ReadBarcodes(setting, page);

//output barcode information

foreach (Barcode barcode in barcodes)

{

Console.WriteLine(barcode.DataString);

}

}

In VB.NET Class

Demo Code: For Barcode Reading from PDF

Public Shared Sub ReadBarcodeFromPdf(filename As String, pageIndex As Integer)

WorkRegistry.Reset()

' invoke this static method only once at the beginning of codes

'generate pdf document

Dim doc As BaseDocument = New PDFDocument(filename)

'get the page you want to read barcode from

Dim page As BasePage = doc.GetPage(pageIndex)

'set reader setting

Dim setting As New ReaderSettings()

setting.AddTypesToRead(BarcodeType.Code39)

' read out barcode information

Dim barcodes As Barcode() = BarcodeReader.ReadBarcodes(setting, page)

'output barcode information

For Each barcode As Barcode In barcodes

Console.WriteLine(barcode.DataString)

Next

End Sub

**Advanced ReaderSettings**

Below is a table list of settings that you can control in .NET barcode reading applications.

**Parameters in**

**Description**

**ReaderSettings**

Direction

Used to control the direction of whole barcode reading

process. Valid values: BottomToTop, LeftToRight, RightToLeft,

TopToBottom, Undefined.

NumberOfBarcodeToRead

Used to define the numbers of barcodes that you want to

read from image or document. This can be used to save

barcode scanning time. For instance, there’s one barcode on

your image, you may set this to be 1. Then our reader will

scan it only and save time from scanning the whole image.

ReadingQuality

Used to determine which is the priority, speed when set to

false or quality of reading when set to true.

RegionOfInterest

Used to define a rectangle for specific area barcode reading.

To achieve this, you can set the region of interest as (start

point horizontal, start point vertical, stop point horizontal,

stop point vertical). Please note that, the points values should

be in percentage. For example, (0, 0, 50, 50) means to scan

left top quarter of the image.

ScanInterval

Used to control intervals between two scans.

SkipValidation

Used to validate accuracy of barcode scanning.

Type

Used to define a barcode type for scanning from your image.

This is suggested to be set if you need to scan a specific

barcode type. If it is “All”, the scanning process will be slowed

down.

**Barcode Create**

Barcode Creator Add-on is an advanced control, seamlessly integrating with RasterEdge

DocImageSDK for .NET. This Barcode Creator Add-on can be easily used in .NET Framework 2.0

and above for high-quality linear and 2d barcodes creation and customization. More than 20

barcodes can be created. See as below.

**Linear Barcodes**

**2DBarcodes**

Codabar

Code 128

Identcode

ITF-14

RM4SCC

Data Matrix

Code 11

EAN-8

Intelligent Mail

Leitcode

UPC-A

PDF-417

Code 2 of 5EAN-13

Interleaved 2 of 5 MSI Plessey

UPC-E

Micro PDF-417

Code 39

EAN-128

ISBN

PLANET

QR Code

Code 93

GS1 DataBar ISSN

POSTNET

Micro QR Code

To test our Barcode Creator Add-on in .NET, C#, VB.NET applications, you need to:

 Integrate package dlls into Visual Studio (2005 or later) project by adding reference;

In all, you need these dlls: RasterEdge.Imaging.Basic.dll,

RasterEdge.Imaging.Basic.Codec.dll, and RasterEdge.Imaging.Barcode.Creator.dll.

 Ready to create barcodes in C# and VB.NET Classes.

**How to’s**

**How to: Draw Barcode on Image**

The demos below can help you draw barcodes on images and the supported image formats

include png, gif, jpeg, bmp and tiff. Here, we take QR Code drawing in C# and VB.NET Classes as

examples.

In C# Class

Demo Code:

WorkRegistry.Reset();// invoke this static method only once at the beginning of codes

// create an REImage you want to draw on

REImage reImage = new REImage(@"c:\samplePNG.png", ImageType.PNG);

//Create a QR Code barcode

QRCode barcode = new QRCode();

barcode.Data = "test123456789"; //Input encodable data

barcode.X = 20.0F; //Set module size

//set the qr code margin which for qr code should be 4 times the width of X module

barcode.LeftMargin = barcode.RightMargin = barcode.TopMargin = barcode.BottomMargin =

20.0f \* 4;

barcode.Resolution = 96; //Set QR Code barcode printing resolution

barcode.Rotate = Rotate.Rotate0; //Set rotate

barcode.DrawBarcode(reImage, 0, 0); //Draw barcode on REImage with location x and y

reImage.Save(ImageType.PNG, @"c:\qr.png");

You can not only draw barcode on image but also on document page.

Demo Code:

WorkRegistry.Reset();// invoke this static method only once at the beginning of codes

Linear linearBarcode = new Linear();

linearBarcode.Type = BarcodeType.CODE39;

linearBarcode.Data = "123456789";

linearBarcode.Resolution = 96;

linearBarcode.Rotate = Rotate.Rotate0;

// load pdf document, you can also load document like tiff, word, excel, ppt

PDFDocument pdf = new PDFDocument(@"c:\REImage.pdf");

// get the first page

BasePage page = pdf.GetPage(0);

// create barcode image from barcode Object

REImage barcodeImage = linearBarcode.ToImage();

// add barcode image to the desired location

page.AddImage(barcodeImage, new System.Drawing.PointF(100f, 100f));

pdf.Save(@"c:\REImageBarcoded.pdf");

In VB.NET Class

Demo Code:

WorkRegistry.Reset()

' invoke this static method only once at the beginning of codes

' create an REImage you want to draw on

Dim reImage As New REImage("c:\samplePNG.png", ImageType.PNG)

'Create a QR Code barcode

Dim barcode As New QRCode()

barcode.Data = "test123456789"

'Input encodable data

barcode.X = 20F

'set the qr code margin which for qr code should be 4 times the width of X module

barcode.LeftMargin = barcode.RightMargin = barcode.TopMargin = barcode.BottomMargin =

20.0f \* 4

'Set module size

barcode.Resolution = 96

'Set QR Code barcode printing resolution

barcode.Rotate = Rotate.Rotate0

'Set rotate

barcode.DrawBarcode(reImage, 0, 0)

'Draw barcode on REImage with location x and y

reImage.Save(ImageType.PNG, "c:\qr.png")

Besides drawing barcode on image, you can also draw it on document page.

Demo Code:

WorkRegistry.Reset()

' invoke this static method only once at the beginning of codes

Dim linearBarcode As New Linear()

linearBarcode.Type = BarcodeType.CODE39

linearBarcode.Data = "123456789"

linearBarcode.Resolution = 96

linearBarcode.Rotate = Rotate.Rotate0

' load pdf document, you can also load document like tiff, word, excel,ppt

Dim pdf As New PDFDocument("c:\REImage.pdf")

' get the first page

Dim page As BasePage = pdf.GetPage(0)

' create barcode image from barcode Object

Dim barcodeImage As REImage = linearBarcode.ToImage()

' add barcode image to the desired location

page.AddImage(barcodeImage, New System.Drawing.PointF(100F, 100F))

pdf.Save("c:\REImageBarcoded.pdf")

**How to: Create Barcode and Save as Image**

If you want to directly create a barcode and save it as image in C# or VB.NET Class, the following

demo codes may help you.

In C# Class

Demo Code:

Linear barcode = new Linear();//Create a barcode

barcode.Type = BarcodeType.CODE128;//Select barcode type

barcode.Data = "123456789";//Set barcode data

barcode.X = 1.0F;//Set x

barcode.Y = 60.0F;//Set y

barcode.Resolution = 96;//Set resolution

barcode.Rotate = Rotate.Rotate0;//Set rotate

REImage reImage = barcode.ToImage();

reImage.Save(ImageType.PNG, @"c:\qr.png");

In VB.NET Class

Demo Code:

Dim barcode As New Linear()

'Create a barcode

barcode.Type = BarcodeType.CODE128

'Select barcode type

barcode.Data = "123456789"

'Set barcode data

barcode.X = 1F

'Set x

barcode.Y = 60F

'Set y

barcode.Resolution = 96

'Set resolution

barcode.Rotate = Rotate.Rotate0

'Set rotate

Dim reImage As REImage = barcode.ToImage()

reImage.Save(ImageType.PNG, "c:\qr.png")

**Advanced Settings**

Below is a table of all supportive barcodes properties. You may free to customize all the

properties in your .NET project.

**ClassAttribute**

**Value**

**Description**

**Barcode Type**

All linear

Type: bool; Default:

Enabled to add barcode

barcodes. For

AddCheckSum

false

checksum at the end.

Intelligent Mail,

this property is

not applied.

AutoResize

Type: bool; Default:

Enabled to automatically resize

All barcodes

false

the created barcode image.

BackColor

Type: Color;

Used to set the background

All barcodes

Default:white

color of barcode image.

Used to set the horizontal

Type: int

BarAlignment

All barcodes

alignment of barcode inside the

Default: Center

image. Valid values: Left,

Center, Right.

Used to set the height of

Type: float

barcode image. If setting is

BarcodeHeight

All barcodes

Default: 0

smaller than required minimum

height, the barcode image will

be automatically reset.

Used to set the width of

Type: float

barcode image. If setting is

BarcodeWidth

All barcodes

Default: 0

smaller than required minimum

width, the barcode image will

be automatically reset.

Used to set the value of top and

Type: float

BearerBarHori

ITF14

bottom bars (horizontal bearer

Default: 1

bars). Valid values: 0 to 10,

which should be a multiple of X.

Used to set the value of left and

Type: float

BearerBarVert

ITF14

right bars (vertical bearer bars).

Default: 1

Valid values: 0 to 10, which

should be a multiple of X.

Used to set the size of bottom

margin. For linear barcodes,

Type: float

10X is recommended. For Data

BottomMargin

All barcodes

Default: 0

Matrix, PDF417, Micro PDF417,

QR Code, Micro QR Code,

X/2X/4X, 2X, X, 4X, 2X are

recommended respectively.

Type:

Used to set the start character

CodabarStartCh

CODABAR

CodabarStartStopCh

of Codabar. Valid values:

ar

ar

CodabarStartStopChar.A, B, C,

Default: A

D.

Type:

Used to set the stop character

CodabarStopCh

CODABAR

CodabarStartStopCh

of Codabar. Valid values:

ar

ar

CodabarStartStopChar.A, B, C,

Default: A

D.

Used to set the number of

Type: int

ColumnCount

PDF417

columns and this should be

Default: 5

increased according to your

data size. Valid values: 1 to 30.

Used to input the data value

Type: string

Data

All barcodes

that will be encoded in barcode.

Default: ""

Valid values vary from different

barcode types.

Used to set the data encoding

mode of Data Matrix. Valid

Type:

values:

DataMatrixDataMod

DataMatrixDataMode.Auto,

e

ataMatrixDataMode.ASCII,

DataMatrix

Default:DataMatrixD

DataMatrixDataMode.C40,

ataMode.ASCII

DataMatrixDataMode.Text,

DataMatrixDataMode.X12,

DataMatrixDataMode.Edifact,

DataMatrixDataMode.Base256.

Used to set the data encoding

Type:PDF417DataM

mode of (Micro) PDF417. Valid

DataMode

ode

PDF417,

values: PDF417DataMode.Auto,

Default:

MicroPDF417

PDF417DataMode.Text,

PDF417DataMode.T

PDF417DataMode.Byte,

ext

PDF417DataMode.Numeric,

PDF417DataMode.Customer.

Used to set the data encoding

Type:

mode of (Micro) QR Code. Valid

QRCodeDataMode

values:

QRCode,

Default:

QRCodeDataMode.Auto,

MicroQRCode

QRCodeDataMode.A

QRCodeDataMode.AlphaNumer

uto

ic, QRCodeDataMode.Byte,

QRCodeDataMode.Numeric,

QRCodeDataMode.Kanji.

Used to enable the output data

Type: int

ECI

QRCode

stream to have different

Default: 3

interpretations that differ from

the default character set.

Type: PDF417ECL

Used to set the error correction

PDF417

Default:

level of PDF417. Valid values:

PDF417ECL.Level\_2

PDF417ECL.Level\_0 to

ECL

PDF417ECL.Level\_8.

Type: QRCodeECL

Used to set the error correction

QRCode

Default:

level of QR Code. Valid values:

QRCodeECL.L

QRCodeECL.L, QRCodeECL.M,

QRCodeECL.Q, QRCodeECL.H.

FileId

Type: int

Use to set FileId property to be

DataMatrix

Default: 0

identified to the same file.

Type: FNC1

Used to encode GS1 compatible

EAN128,

FNC1

Default: FNC1.

barcode and FNC1 value should

DataMatrix,

FNC1\_NONE

be set to FNC1.FNC1\_1ST\_POS.

QRCode

ForeColor

Type: Color

Used to set the foreground

All barcodes

Default: black

color of barcode image.

Type:

Used to define the format of

DataMatrixFormatM

Data Matrix to use on that

ode

symbology. Valid values: please

FormatMode

DataMatrix

Default:

directly see enum

DataMatrixFormatM

DataMatrixFormatMode.Forma

ode.

t\_\*X\*.

Format\_10X10

Type: float

Used to set the space between

I

CODE39

Default: 1.0f

two characters of barcode.

Valid value: a multiple of X.

Used to set the size of left

margin. For linear barcodes,

Type: float

10X is recommended. For Data

LeftMargin

All barcodes

Default: 0

Matrix, PDF417, Micro PDF417,

QR Code, Micro QR Code,

X/2X/4X, 2X, X, 4X, 2X are

recommended respectively.

Macro

Type: bool

Enabled to apply Macro PDF417

PDF417,

Default: false

function.

MicroPDF417

MacroSegmentI

Type: int

Used to set the position of

PDF417,

ndex

Default: 0

current symbol in the sequence,

MicroPDF417

which begins with 0.

MacroSegment

Type: int

Used to set the number of total

PDF417,

Count

Default: 0

symbols which consist of the

MicroPDF417

sequence.

MacroFileIndex Type: int

Be identified to the same file.

PDF417,

Default: 0

MicroPDF417

CODABAR,

Used to set the ratio of wide

Type: float

CODE2OF5,

N

bar to narrow bar. Valid values:

Default: 2.0f

CODE39,

2.0 to 3.0 inclusive.

INTERLEAVED25,

ITF14

Parity

Type: int

Used to set Parity property of

QRCode

Default: 0

QR Code.

CODE39,

Type: bool

CODE128,

Default: false(for

Enabled to use the tilde

EAN128,

ProcessTilde

linear

character "~" to specify special

DataMatrix,

barcodes)/true (for

characters in input data.

QRCode,

2d barcodes)

MicroQRCode,

PDF417,

MicroPDF417

Type: int

Used to set the barcode image

Resolution

All barcodes

Default: 72

resolution (in DPI, Dots per

inch).

Used to set the size of right

margin. For linear barcodes,

Type: float

10X is recommended. For Data

RightMargin

All barcodes

Default: 0

Matrix, PDF417, Micro PDF417,

QR Code, Micro QR Code,

X/2X/4X, 2X, X, 4X, 2X are

recommended respectively.

Used to rotate barcode image

to a desired position. Valid

Type: Rotate

Rotate

values: Rotate.Rotate0,

All barcodes

Default: Rotate0

Rotate.Rotate90,

Rotate.Rotate180,

Rotate.Rotate270.

RowCount

Type: int

Used to set the number of

PDF417

Default: 3

rows. Valid values: 3 to 90.

ShortTallRatio

Type: float

Used to set the ratio of short

PLANET, POSTNET

Default: 0.4f

bar to tall bar (Y).

All linear

Enabled to show the check

ShowCheckSum

Type: bool

barcodes. For

digit(s) at the end of barcode

Char

Default: true

Intelligent Mail,

text.

this property is

not applied.

ShowStartStopI

Type: bool

Enabled to show a \* at the

CODE39

nText

Default: true

beginning and end of barcode

text.

ShowText

Type: bool

Enabled to show barcode text

All linear barcodes

Default: true

under the barcode bars.

StructuredAppe

Type: bool

Enabled to apply Structured

DataMatrix,

nd

Default: false

Append function to barcode.

QRCode

Type: int

Used to set the number of total

DataMatrix,

SymbolCount

Default: 0

symbols which consist of the

QRCode

sequence.

Type: int

Used to set the position of

DataMatrix,

SymbolIndex

Default: 0

current symbol in the sequence,

QRCode

which begins with 0.

EAN8\_2, EAN8\_5,

Used to input the supplement

EAN13\_2,

Type: string

data to encode in add-on

EAN13\_5, ISBN\_2,

SupData

Default: ""

barcode. Valid values: 2 or 5

ISBN\_5, ISSN\_2,

digits.

ISSN\_5, UPCA\_2,

UPCA\_5, UPCE\_2,

UPCE\_5

EAN8\_2, EAN8\_5,

EAN13\_2,

Used to set the height of add-

Type: float

EAN13\_5, ISBN\_2,

SupHeight

on barcode bar. Valid value: a

Default: 0.8f

ISBN\_5, ISSN\_2,

multiplicator of Y.

ISSN\_5, UPCA\_2,

UPCA\_5, UPCE\_2,

UPCE\_5

EAN8\_2, EAN8\_5,

EAN13\_2,

Used to set the space between

Type: float

EAN13\_5, ISBN\_2,

SupSpace

the main barcode and its add-

Default: 15

ISBN\_5, ISSN\_2,

on barcode.

ISSN\_5, UPCA\_2,

UPCA\_5, UPCE\_2,

UPCE\_5

TextColor

Type: Color

Used to set the color of barcode

All linear barcodes

Default: black

text.

Type: Font

Used to set the font style of

TextFont

All linear barcodes

Default:

barcode text.

new Font("Arial", 9f,

FontStyle.Regular)

TextMargin

Type: float

Used to set the space between

All linear barcodes

Default: 6

barcode bar and data text.

Used to set the size of top

margin. For linear barcodes,

Type: float

10X is recommended. For Data

TopMargin

All barcodes

Default: 0

Matrix, PDF417, Micro PDF417,

QR Code, Micro QR Code,

X/2X/4X, 2X, X, 4X, 2X are

recommended respectively.

TopTextColor

Type: Color

Used to set the color of text

ISBN, ISSN

Default: black

above barcode.

Type: Font

Used to set the font style of text

Default:

TopTextFont

ISBN, ISSN

above barcode.

new Font("Arial", 9f,

FontStyle.Regular)

Enabled to apply Truncated

Type: bool

PDF417 function, which may be

Truncated

PDF417

Default: false

used if space considerations are

the primary concern and

symbol damage is unlikely.

Type

Type: BarcodeType

Used to set the linear barcode

All linear barcodes

Default: CODE128

type.

Use to set the unit of measure

Type:

for all size related settings.

UOM

UnitOfMeasure

All barcodes

Valid values:

Default: PIXEL

UnitOfMeasure.PIXEL,

UnitOfMeasure.CM,

UnitOfMeasure.INCH.

UPCENumber

Type: int

Used to set the number system

UPCE

Default: 0

of UPC-E. Valid values: 0, 1.

Type:

Used to set the version of Micro

MicroPDF417Versio

PDF417. Valid values: please

MicroPDF417

n

directly see enum

Default:

MicroPDF417Version

Version

Version\_3X08.

Version\_\*X\*.

Type:

Used to set the version of QR

QRCode

QRCodeVersion

Code. Valid values: V1 to V40.

Default:

QRCodeVersion.V1

Type:

Used to set the version of Micro

MicroQRCodeVersio

QR Code. Valid values: M1 to

MicroQRCode

n

M4\_Q.

Default:

Version.M4\_L.

Type: float

Used to set the width of

X

All barcodes

Default: 1

barcode bar module (narrow

bar).

XtoYRatio

Type: float

Used to set the ratio of bar

PDF417, Micro

Default: 0.3333333f

width to bar height.

PDF417

Y

Type: float

Used to set the height of

All linear barcodes

Default: 60

barcode bar module.

**TWAIN Scanning**

**TWAIN Scanning Overview**

With RasterEdge.Imaging.Twain assembly you can acquire raw images from input devices such

as scanner and camera. You can query the capacities supported by the device. You may set or

alter scanning properties, if supported, for the acquisition process.

Since RasterEdge.Imaging.Twain is a dll that is built on official Twain driver. Please install the

official TWAINDSM.dll (provided in our dlls collection) manually before using

RasterEdge.Imaging.Twain. Simply put TWAINDSM.DLL in x86 folder under the Windows System

directory (normally C:\Windows\System32). On a 64bit system, make sure TWAINDSM.DLL in

x64 folder ends up in the WOW64 System directory (normally C:\Windows\SysWOW64)。

Following are the basic classes you need to know about to gain image from scanning process.

**Acquisition**

The Acquisition object is the primary class in RETwain. You can drop this component onto a

Form after adding it to the toolbox, or you can instantiate it directly. This is the only class that

you need to add standard image acquisition capabilities to an application. For greater control

over the acquire process, this class contains a collection of Device objects that control numerous

properties used for the image acquisition.

**TWAINDevice**

The TWAINDevice object provides full access to a TWAIN compatible source on the system. Use

it to open a connection to the device, to get and set properties, and then to acquire one or more

images. Because this class represents a system device resource, you cannot create an instance

of it. You can obtain an instance to a Device object by calling GetAvailbleDevices, or from the

Devices collection in the Acquisition object.

**Getting Started with RETwain**

If you do not want to set and query properties of the Device or customize your acquisition, an

instance of the Acquisition class is enough for basic scanning process. Call the acquire method

for the scanning process. Default property and device are used.

*Note: You need to add reference to RasterEdge.Imaging.Twain.dll if you want to use RETwain.*

Example code

public void Scanning()

{

Acquisitionacq = new Acquisition();

acq.ImageAcquired += new ImageAcquiredEventHandler(acq\_ImageAcquired);

acq.Aquire();

}

**Setting Up Events**

You need to use events when acquiring images. When an image is acquired, the ImageAcquired

event fires, providing an ImageAcquiredEventArgs object that contains the image. At the very

least, theImageAcquired event must be handled, and it is recommended that the

AcquireCanceled and AcquireCompleted events also be handled. The following code shows how

the image is handled.

this.acquisition.ImageAcquired += new ImageAcquiredEventHandler (OnImageAcquired);

private void OnImageAcquired(object sender, ImageAcquiredEventArgsargs)

{

// If the image exists, save it to local file or do some operation

if(args.OutputImage!= null)

{

args.OutputImage.Save(RasterEdge.Imaging.Basic.ImageType.PNG, @"C:\test.png");

}

}

**Getting and Setting Properties**

To get or set a device property, you must open a connection to the device using the **Open()**

method. Whenever the **Open()** method is invoked, the **Close()** method must be invoked to close

the connection. Closing a connection resets all of the device properties to their default values

and therefore a device should be closed after the image or all desired properties have been

acquired.

Note: To get a device object you must get it through an **Acquisition** Instance.

The code below opens a connection to the device in order to retrieve the default

Resolutionvalues of the device, and then closes the connection. This technique can be useful if

you are looking for a device of choices with specific default properties or capabilities.

C#

device.Open();

int res = device.ScanSetting.Resolution;

int bitDepth = device.ScanSetting.BitDepth;

device.Close();

**How to’s**

**How to Do Console Based Scanning**

Scanning from the Console is done similarly to scanning in a WinForms application

public class AcquisitionClass

{

static void Main(string[] args)

{

Acquisition acquisition = new Acquisition();

AddEvents(acquisition);

count = 0;

List<string> names = acquisition.GetAvailbleDevicesName();

string deviceName = names[0];

TWAINDevice device = acquisition.GetDevice(deviceName);

Console.Out.WriteLine("---Beginning Scan---");

device.Open();

//scan all availble pages

device.ScanSetting.ShouldTransferAllPages = true;

device.Acquire();

Console.Out.WriteLine("---Ending Scan---\n Press Enter To Quit");

Console.In.Peek();

}

private static void AddEvents(Acquisition acquisition)

{

acquisition.ImageAcquired += new

ImageAcquiredEventHandler(acquisition\_ImageAcquired);

acquisition.AcquireCompleted += new

AcquireCompletedEventHandler(acquisition\_AcquireCompleted);

acquisition.AcquireCanceled += new

AcquireCanceledEventHandler(acquisition\_AcquireCanceled);

}

static void acquisition\_AcquireCanceled(object sender, EventArgs args)

{

Console.Out.WriteLine("Acqusition Canceled");

}

static void acquisition\_AcquireCompleted(object sender, AcquireCompletedEventArgs args)

{

Console.Out.WriteLine("Acquisition Finished");

}

static int count;

static void acquisition\_ImageAcquired(object sender, ImageAcquiredEventArgs args)

{

string filename = "out" + count++ + ".tif";

//save scanned images as png

args.OutputImage.Save(ImageType.PNG, @"c:\"+count+".png");

Console.Out.WriteLine("Frame " + count + " Acquired. Saved At: " + filename);

}

}

**How to Scan Many Pages into a PDF or TIFF file**

A common task in the document imaging world is to scan many pages into a single file. RETwain,

along with REImage can easily be used to accomplish this task. The two most popular multipage

image formats are TIFF and PDF, and this example will concentrate on PDF. Since in RasterEdge

Imaging, the TIFFDocument and PDFDocument have common prototype, BaseDocument Class,

so the procedure for TIFF is very similar.

The key is to set up the event to assist with the process of scanning from device and convert the

image obtained to form a PDFDocument.

 **AcquireCanceled**: Raised if the user cancels the acquisition process.

 **AcquireCompleted**: Raised when the scanner has finished scanning the last page.

 **ImageAcquired**: Raised each time the scanner finishes scanning a page.

Note: You need to reference RasterEdge.Imaging.PDF Assembly and RasterEdge.Imaging.Twain

Assembly to your project to complete the following function.

private bool \_acquireCanceled;

private List<REImage> \_imageList;

public void ScanImages()

{

\_acquireCanceled = false;

Acquisition myAcquisition = new Acquisition();

myAcquisition.AcquireCanceled += new AcquireCanceledEventHandler(OnAcquireCanceled);

myAcquisition.AcquireCompleted += new

AcquireCompletedEventHandler(OnAcquireCompleted);

myAcquisition.ImageAcquired += new ImageAcquiredEventHandler(OnImageAcquired);

TWAINDevice device = myAcquisition.DefaultDevice;

device.Open();

device.ScanSetting.ShouldTransferAllPages = true;

\_imageList = new List<REImage>();

device.Acquire();

}

private void OnImageAcquired(object sender, ImageAcquiredEventArgs args)

{

if (args.OutputImage != null)

{

//Add Scaned images to image collection

\_imageList.Add(args.OutputImage);

}

}

private void OnAcquireCanceled(object sender, EventArgs args)

{

\_acquireCanceled = true;

}

private void OnAcquireCompleted(object sender, AcquireCompletedEventArgs args)

{

if (\_acquireCanceled)

return;

//form a pdf document using image collection scanned

PDFDocument doc = new PDFDocument(\_imageList.ToArray());

//save pdf document

doc.Save(@"c:\ScannedPDF.pdf");

doc.Dispose();

// do some extra work to finish compose the pdf document.

////Get a tiff document using similar operation

//TIFFDocument doc = new TIFFDocument(\_imageList.ToArray());

////save tiff document

//doc.Save(@"c:\scannedTiff.tif");

//doc.Dispose();

}

}

**Licensing RasterEdge Imaging**

**Purchasing License**

Before you can use RasterEdge Toolkit to develop your commercial software, you need to obtain

required SDK license. The license can be purchased directly from the RasterEdge website

(http://www.rasteredge.com/purchase/).

If you have any questions, please contact us at support@rasteredge.com.