RasterEdge Developer's Guide

Introduction

About RasterEdge Image

- RasterEdge Image is a suit of .NET assemblies to use in project that can acquire, create, load, write, display, annotate or process images. In simple words, it deals with images of various formats. RasterEdge Image assemblies can be used in C#, ASP.NET and other .NET-compatible projects.
- In addition to core processing dlls, RasterEdge Image includes .NET controls that can be added to WinForm and WebForm projects. These controls are easy to be incorporated into your image processing projects.
- Image source can be obtained from files or databases, and from scanners using our Twain application.
- Image source imported can be in different formats including various raster image formats such as BMP, TIFF, GIF, PNG, JPEG, JBIG2, and JPEG2000. And image can be converted from a number of document formats, for example, PDF, Microsoft Word, Excel, and PowerPoint. This is realized by our mighty codec assemblies.
- Conversion between image formats can be extremely easy using codecs of RasterEdge.
- Over 20 kinds of image processing commands are available. These commands are categorized into 4 types including image transforms, filters, channels and effects.

About RasterEdge Document Imaging

RasterEdge Document Imaging is a powerful document imaging assembly. It can perform high efficiency document image scanning, viewing, annotating and editing. The Document Imaging toolkit is easy to be integrated into your project. It comprises of an imaging class library for different document formats such as TIFF, PDF, Dicom, Word, Excel and PowerPoint. The toolkit also includes a Windows Forms control library, an ASP.NET AJAX-Enabled Server-Side Image Viewer, and a Twain application for Twain Scanning. Annotation is supported for both Windows Form and ASP.NET Web Form controls.

The product feature includes:

- Encoding and decoding for multi-page TIFF.
- Multipage PDF encoding and decoding.
- Imaging viewing functions forMicrosoft Word, Excel, and PowerPoint.

- Create PDF or TIFF using image source obtained from file, data base or scanning process.
- Render page in document of above document formats into different image formats to view.
- Specify resolution or size during document pagerendering when try to get thumbnails of pages.
- Twain scanning application enable you to acquire images from scanners or cameras.
- Advanced Windows Form controls to load, display, annotate and save above document formats.
- WebForms controls toload, display, annotate and save above document formats with featured AJAX technology.
- Add image (logo or watermark) to specified page in document.
- Insert, delete, and reorder TIFF pages in multipage TIFF document.
- Insert, delete, and reorder PDF pages in multipage PDF document.
- 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 yourproject using Visual Studio, first you should invoke the Add Reference dialog box, choose browse tab and locate the DLLs you download from our server. The relative path of the DLLs in the download package is RasterEdge.DocImageSDK\RasterEdge.DocImageSDK6.0\bin. Remember to choose appropriate system type. One exception is that you if you use any document imaging assembly like PDF or Word add-on, you should manually add either RasterEdge.Imaging.Drawing.Font32.dll or RasterEdge.Imaging.Drawing.Font64.dll(located under RasterEdge.DocImageSDK\RasterEdge.DocImageSDK 6.0 bin \Font assemblies) to the release or debug folder of your .NET project. This is due to discrepancy of the C++ to .NETproject.

Licensing RasterEdge Imaging

Free Trial

You can gain a30 days free trialand an additional of 30 days extension if you need to continue evaluating this product. The Activation tool for free trial is under the download package with the relative path RasterEdge.DocImageSDK\RasterEdge.DocImageSDK6.0\License manager\ RasterEdge.Imaging License Manager.exe. Double click the .exe file and select Activate trial license. Then the trial license is registered.

Purchasing License

Before you can use RasterEdge Toolkit to develop your commercial software you need to obtain the SDK developer license. The license can be purchased directly from the RasterEdge website.

Activate SDK Developer Licenses

After you purchase the Developer License we will send you an activation code to activate your purchased DLLs, and the Activation tool is under the download package with the relative path RasterEdge.DocImageSDK\RasterEdge.DocImageSDK6.0\License manager\ RasterEdge.Imaging License.In the developing process, you just need to double click the .exe file and choose Activate purchased license to input the activation code that we send to you for registration.

Each RasterEdge SDK developer must have his or her own developer build license. A single developer build license canbe used to develop an unlimited number of single-user client applications deployed to an unlimited number ofclients desktop PC's as long as the licenseis still under maintenance.

Application Licenses

After you finish developing process and want to distribute your application (for example, .exe) to your own customer, you must use the License generating tool under the path RasterEdge.DocImageSDK\RasterEdge.DocImageSDK6.0\License manager\DistributionLicFile.exe to generate a license file, thenput in placewith DLLsand your application.

Do the following steps:

- Add the following declaration**outside** the class of the entry point of your program.

```
[LicenseProvider(typeof(WinFormLicenseProvider))]
```

- Then add the following code into the entry point of your program

```
Licenselicense = System.ComponentModel.LicenseManager.Validate(typeof(Form1), this);
```

- Add a new empty class to your project which inherits `RasterEdgeLicenseProvider`class defined in the RasterEdge.Imaging.Basic.dll

For example, if you construct a WinForm application, the entry point of your program should be something look like this:

```
[LicenseProvider(typeof(WinFormLicenseProvider))]
```

```
publicpartialclassForm1 : Form
    {


public Form1()
        {
            InitializeComponent();
Licenselicense = System.ComponentModel.LicenseManager.Validate(typeof(Form1), this);

        }
```

And a new empty class should be added to your project something like this:

```
    publicclassWinFormLicenseProvider : RasterEdgeLicenseProvider
    {

    }
```

Programming with RasterEdge Image Assembly

Getting Started

RasetEdge Image toolkitis a powerful imaging solution to your desktop or thinclient application. With a collection of controls for ASP.NET and Windows, you can integratea light and powerful solution into your document imaging and image processing project. Licensingis straightforward and runtime royalty free on the 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 assembly(Core SDK)	Including Codecs for basics image formats (Bitmap, GIF&PNG) and image processing.
PDF Add-On	Fast and powerful Codec, imaging, annotating and editing for document of PDF format
TIFF Add-On	Fast and powerful Codec, imaging, annotating and editing for document of TIFF format
Barcode Reader Add-On	Read barcode from an image(or document page)obtained
Barcode Generator Add-On	Write barcode to specific image(or document page)
DICOM Codec Add-On	Codecs for DICOM image format
JBIG2 Codec Add-On	Codecs for JBIG2 image format
JPEG2000 Codec Add-On	Codecs for JPEG2000 image format

Image Concepts

Image Data

Image data contains color information for every pixel in the image. REImage provide a property of ImageData from which you can use unsafe code to change/set image data directly.

Image Compressions

Why Compression

In a raw state, images can occupy a rather large amount of memory both in RAM and instorage. Image compression reduces the storage space required by an image and the bandwidthneeded when streaming that image across a network.

Types of Compression

CompressionTypes

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 the image will be lost during compression, so the resulting image is not identical to the original one.

CompressionMethods

The following compression methods are available in RasterEdge Image.
This topic describes compression and the different methods of compression available in REImage.
In a raw state, images can occupy a rather large amount of memory both in RAM and storage. Image compression reduces the storage space required by an image and the bandwidth needed when streaming that image across a network.
We contains compressions below: (This content is in RasterEdge.Imaging.Basic)

Use method void EncodeImage(BaseImage image, FileIOMgr fileIOMgr);

- 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 the image in various compression modes. For more information, pleasesee REImage.

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 the 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 the image (document) formats listed in the table below. This table also shows the location of these codecs in particular assembly or plug-in.

Format	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
Tiff	TIFFDecoder	TIFFEncoder	RasterEdge.Imaging.TIFF
Pdf	PDFDecoder	PDFEncoder	RasterEdge.Imaging.PDF
JBIG2	JBIG2Decoder	JBIG2Encoder	RasterEdge.Imaging.Basic.codec
Jpeg2000	Jpeg2000Decoder	JpegEncoder	RasterEdge.Imaging.JPEG2000
Dicom	DicomDecoder	DicomEncoder	RasterEdge.Imaging.DICOM

How to Create a Decoder or Encoder

The BaseDecoder/BaseEncoder class has the common interfaces for decoding/ encoding images data and converts it into REImage object. All subclasses for specific image(document) format inherit these APIs.

Note: RasteEdge Imaging represents PDF, TIFF, Dicom, Office Word, Excel, and PowerPoint as REDocument object.You can still get REImage from the REDocument in a slightly different way. See Document Imaging for more Information.

You can create a specific encoder or decoder object using a couple lines of code. Consider the encoder and decoder objects are cheap to make, and you can create such an object every time you need to do encoding or decoding.

Example code:

```
REImage image  = REFile.OpenImageFile(path, new PNGEncoder());
```

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 illustratedin detail.

There's a limitation tothe 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 storethe metadata information in a database, build a metadata viewer application, and to addyour 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 types	Operations	Image Format
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 successfullyembedded 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);
```

REImage

REImage is the core class in RasterEdge Image assembly for representing images obtained from file, stream or scanning process. Every image is represented as a REImage object in memory. With different codecs you can create REImage instances from decoding image data from different image formats stored in file, stream or byte array. You can also obtain REImage instance from scratches or other REImage. After you get a REImage instance you can convert it to Bitmap to view the image or encode REImage into various other image data formats. REImage object serves as a transit during conversionfrom one image format to another. With Document Imaging assembly and proper add-on, you can render a document page into REImage instance of specified size and resolution and then convert to Bitmap for viewing. In addition, you can choose to decide to view the image in our WinForm& WebForm Viewers or anything alike. You can also convert image into a specified format to store in local file system or upload it to a database.

How to's

How to Create REImage Object

A REImage instance is an in-memory representation of a raster (or pixel-based) image. A REImage is defined by several characteristics as listed in the below.

Width	How wide the image is in pixels
Height	How tall the image is in pixels
Resolution	How many pixels span a unit of measure, e.g. pixels per inch Pix-
ImageMode	How the pixels represent color?

REImage can be created by the following methods:

a) From scratch
b) From other Bitmap
c) From File
d) From Byte Array
e) From Stream
f) From REPage object

ReImage can represent black and white, gray, and color images in a number of different formats. They can be converted to and from any format.

The easiest way to create a new image from scratch is with the following code:

```
ReImage image = new ReImage(width, height,ImageMode.Pixel24bppBgr);
```

This creates a new blank image of the given width and height in 24 bit per pixel color using the RGB color model. The image data is set to zero, which in this case creates a black image.

Please refer to table below for more details of ImageMode.

Mode	Story
UNDEF	Equals REImage(width, height);
Indexed1bpp	Indexed, 1 bit per pixel, Binary Image, such as Monochrome (Black/White)
Indexed4bpp	Indexed, 4 bit per pixel, 16 colors in RGB, EGA
Indexed8bpp	Indexed, 8 bit per pixel, 256 colors in RGB, VGA at low resolution
GrayScale16bpp	Gray Scale, 16 bit per pixel, 65536 shades of gray
RGB555	RGB, 16 bit per pixel, 5/5/5 for R/G/B, High Color
RGB565	RGB, 16 bit per pixel, 5/6/5 for R/G/B, High Color
ARGB1555	ARGB, 16 bit per pixel, 1/5/5/5 for A/R/G/B, High Color
RGB888	RGB, 24 bit per pixel, 8/8/8 for R/G/B, True Color
RGB0888	RGB, 32 bit per pixel, 8/8/8 for R/G/B, True Color
ARGB8888	ARGB, 32 bit per pixel, 8/8/8/8 for A/R/G/B, True Color
PARGB8888	Premultiplied ARGB, 32 bit per pixel, 8/8/8/8 for A/R/G/B, True Color
RGB48bpp	RGB, 48 bit per pixel, 16/16/16 for R/G/B, Deep Color
ARGB64bpp	ARGB, 64 bit per pixel, 16/16/16/16 for A/R/G/B, Deep Color
MAX	

You can create a new REImage object from a Bitmap object. You can do something like below.

```
Bitmap bm = new Bitmap (width, height);

ReImage image = new ReImage(bm);
```

How to Open an Image

Toopen(load) images in REImagewith a given path, you can use the following code:

```
ReImage image = REFile.openImageFile(filePath, newPNGDecoder());
```

Note: For different image formats you should use different decoder. For example for a png image, you should use PNGDecoder object as the parameter. See also how to create a decoder or encoder.

From Stream

C# example

```
REImage img = new REImage(stream, new PNGDecoder())
```

Or from REPage that is obtained from document format like PDF, TIFF, Office Word, Excel, and PowerPoint.

C# example

```
using(PDFDocument doc = new PDFDocument(stream))
{
REPage page= doc.GetPage(0);
REImage image = (REImage )page.ToImage();
}
```

See also Document Imaging

How to Save REImage in Native Image Format

You can use encoder to save REImage to a specific image format in file or stream. The following is the one possible APIto use.

```
SaveImageFile(REImage image, String filePath, BaseEncoder encoder)
```

You can convert REImage to byte array by specifying the compression (image format) by the convenient method ToByteArray ().

How to Convert Raster Image to a Bitmap Object (int Format32bppArgb)

Since many .NET APIs use Bitmap Class instance as input or output. REImage Class provides ToBitmap() method to generate a Bitmap object out of it. See examples below.

```
        REImage image;
// specify the bitmap dimension
        Bitmap bm = image.ToBitmap(image.width, image.height);
```

You can also use the default dimension:

```
ReImage image;
Bitmap bm = image.ToBitmap();
```

See Also Image Processing

See Also Accessing Images

See Also Image Convert

See Also Document Imaging

See Also View Images

Process Images

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 to 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.

Accessing Images

Images can be stored in a database with REImage by using a subclass of Stream, or the convenientToByteArray() and FromByteArray() methods of the REImage object.
When using a SQL database, the image should be stored in a binary image field. In MS Access this would be an OLE field. The following code samples show how to read and writeimages with either SQL or Access databases using ADO.NET.

- Write image to database

Writing an REImage into a SQL Database

ExampleCode:

C#

```
SqlConnection myConnection = null;
try
{
//save image to byte array and allocate enough memory for the image
byte[] imagedata = image.ToByteArray(new RasterEdge.Imaging.Basic.Codec.JPGEncoder());
//create the SQL statement to add the image data
myConnection = new SqlConnection(CONNECTION_STRING);
SqlCommand myCommand = new SqlCommand("INSERT INTO RasterEdge_Image_
Database
(Caption, ImageData) VALUES ('" + txtCaption.Text + "',
@Image)", myConnection);
SqlParameter myParameter = new SqlParameter("@Image",
SqlDbType.Image, imagedata.Length);
myParameter.Value = imagedata;
myCommand.Parameters.Add(myParameter);
//open the connection and execture the statement
myConnection.Open();
myCommand.ExecuteNonQuery();
}f
inally
{
myConnection.Close();
}
```

Visual Basic

```
Dim myConnection As SqlConnection = Nothing Try 'establish connection
and SELECT statement
myConnection = New SqlConnection(CONNECTION_STRING)
Dim myCommand As SqlCommand =
New SqlCommand("SELECT ImageData FROM RasterEgde_Image_Database
" + _
"WHERE Caption = '" + txtCaption.Text + "'", myConnection)
myConnection.Open()
'get the image from the database
Dim imagedata() As Byte = CType(myCommand.ExecuteScalar(), Byte())
If (Not imagedata Is Nothing) Then Return
REImage.FromByteArray(imagedata)
Else
MessageBox.Show("Image does not exist in database.")
Return Nothing End If Finally
myConnection.Close()
End Try
```

View Images or Documents

RasterEdge Imaging toolkit includes ASP.NET AJAX enabled viewer control and Windows Form controland you can integrate these controls into your .NET or ASP.NETproject to view document image.

WebViewer(ASP.NET WebForm Control)

Windows Viewer (Windows Form Control)

Image Convert

RasterEdge Imaging toolkit implements reliable encoding and decodingfunctions for raster images of different formats. You can convert between different image formats with only a couple lines of codes.In addition, RasterEdge Imaging toolkit enables featured document imaging processing,soyou can create documents like TIFF and PDF using an image source. Rendering documents to images is also easy to use. So you can convert between raster image formats and PDF, as well as TIFF.

How to's

How to ConvertBetween Raster Images Using Codecs

To convert between raster images, all you need is the codec functions provided by RasterEdge Imaging toolkit. For example, you can convert a Bimap image to PNG format using the following code.

```
REImage img  = REFile.OpenImageFile(@“c:\test.bmp”, new BMPDecoder());

REFile.SaveImageFile(@“c:\test.png”, new PNGEncoder());
```

How to ConvertRaster Images to PDF or TIFF

You can convert image to PDF pages and then form a PDF Document using PDFPage(REImage) API or simply use PDFDocument(List<REImage>) to form a multipage PDF document.

See Also Document Imaging.

Document Imaging

Introduction to RasterEdge Document Imaging

Document Imaging is the most important feature in RasterEdge Imaging toolkit, so we would like to explain it in details. Basically you can present documents in various kinds of formats like PDF, TIFF, Dicom, Office Word, Excel(coming soon)and PowerPoint(coming soon) in images. You are able to not only read them from images created out of these documents, but also annotate on them, and burn the changes into the native file. Above features will enable you to communicate

with documents at any time any place, no matter whether it is through the client software or some online ASP.NET application you develop.

The main features of our product are:

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

2. Render document pages to images of various images formats to view.

3. Annotate on the document and burn the changes to it in native file format as embedded image.

4. Page level for multi-page TIFF and PDF document. Processing category includes appending, inserting, deleting and sorting pages. The source of the pages could be other documents or images or simply a blank page you create.

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

6. Burn barcode in document page.

7. Convertamong TIFF, PDF document and image of various formats.

Basic

In RasterEdge Document Imaging toolkit, every document and their page content is represented as a document class object in memory. These document classes despite of specific document format are all derived from REDocument Class which is in turn derived from BaseDocument Class. Pages in document use REPage and BasePage as prototypes. These classes are merely interfaces for document and their page content. 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 = newPDFDocument(@"D:\sample1.pdf");
```

Or construct a word document

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

//note we only support word document that is created by word 2007 or newer version.Where the word files end with a.docx suffix.
```

In BaseDocument and REDocument assembly, you can see the common APIs for all documents classes supported by RasterEdge Document Imaging.

You can generate generate images from documents such as PDFDocument or DOCXDument

PDF

Introduction to PDF Functions

RasterEdge Image provides customers reliable PDF technology with the following features.

- Open or load a PDF document from local file or byte array from database.
- Create and save PDF document file.
- Rasterize PDF document to raster images.
- Convert PDF document into other image format.
- Rich text and graphical annotations can be added to PDF file.
- Edit PDF document,like splitting document or combiningseveral documents into one PDF document.
- Practical PDF page processing functions, like adding or deleting PDF document pages and adding image to PDF page.
- Own 1d & 2d barcodes generating and reading functions.

Before you use these functions above, you should reference our PDF processing dll (RasterEdge.Imaging.PDF.dll) and other necessary assemblies(like RasterEdge.Imaging.Basic) to you project.

About PDF Programming Classes

PDF document assembly provides two PDF document processing classes, which are PDFDocument and PDFPage.

- PDFDocument: This class refers to PDF document and it contains all document information of PDF file. It is an extension of REDocument which is in turn an extension of BaseDocument.
- PDFPage: This class refers to PDF document page contains in PDFDocument and use REPage and BasePage as prototypes.

PDFDocument Objects

At the heart of the PDF manipulation API is a class called PDFDocument. This class represents the main structure of a document and the pages it contains. When a PDFDocument object is created, we can easily extract information about the pages and other document structures.

The example that follows shows how to use different methods to create a PDFDocument object:

C#

Input Stream that contains a PDF:

```
PDFDocument doc = newPDFDocument(Stream s);
```

Input Path of PDF:

```
PDFDocument doc = newPDFDocument(@"D:\BugList\sample1.pdf");
```

Once you have created a document object, the sample code below tells you how to get the number of pages of this object.

C#

```
public int GetPageCount(Stream s)

{

        PDFDocument document = new PDFDocument(s);

        return document.GetPageCount();

}
```

Viewing images from PDFDocument

The following example shows how to get PDFPage class object from the PDFDocument object by providing an index. If you input zero as the page index, then you will get the first page information of the file.

C#

```
public void GetPDFPageFromPDFDocument(String filePath, int index)

{

        PDFDocument document = new PDFDocument(filePath);

        PDFPage page = (PDFPage)document.GetPage(index);

}
```

Once you have some PDFPage object, you can convert these objects to images and save them as the image format that you want. Supported formats include png, bmp, and jpeg. The code below shows you how to render a pdf page and save it as an png image file

C#

```
using RasterEdge.Imaging.PDF.dll;

using RasterEdge.Imaging.Basic.dll;

public void PDFPageToImage(String sourceFilePath, int index, String destnFilePath)
        {
```

```
    PDFDocument doc = new PDFDocument(sourceFilePath);
    PDFPage page = (PDFPage)doc.GetPage(index);
    REImage image = (REImage)page.toImage();
    // you can specify the image dimension by calling
    //REImage image = (REImage)page.toImage(1000,2000);
        REFile.SaveImageFile(image, destnFilePath, new
        RasterEdge.Imaging.Basic.Codec.PNGEncoder());
}
```

REFile is an utility class in RasterEdge.Imaging.Basic.dll.

A more advanced approach to get Bitmap image out of a PDFPage(or DOCXPage) is to render the page onto a REDrawingContext instance. And get bitmap from the context object. It allows you to control the resolution and offset of the images. Another point is that you can reuse the context object to generating images as many times as you like improve the rendering efficiency at the cost of memory. The following code is an example

```
    // first construct a DrawingREContext object
    DrawingREContext context = new DrawingREContext();


    page.RenderToContext(context);
    // set the resolution of the context a bigger resolution generate a larger images
    context.SetResolution(96, 96);
        /*  you can also set the region of the img you want to show. for example the
following code specify a region which starts  at the point (0,0) with width and height
both to be 100 pixels.*/
    context.SetValidRegion(0, 0, 100, 100, Units.PX);

    Bitmap img = context.GetBitmap();
```

Introduction to Editing PDF Documents

RasterEdge Image provides tools for high level editing and composition of PDF documents.

With these tools you can:

- Rearrange, add, or remove pages from existing PDF documents.
- Split existing PDF documents into separate documents.
- Combine any number of existing PDF documents into a single document.
- Update some pages from existing PDF documents.

All these functionalities are tied into simple object models that don't require the programmer to memorize the PDF specification. Much of these functionalities have been extended to cover

existing PDF generation tools, including the document imaging tool for generating image only PDF documents.

With PDF processing dll,there is no obstacle to rearrange pages of a PDF document as long as you provide a new sequence of the PDF document. If the new sequence doesn't contains all page indexes, then the rest page will auto to add to the file tail. For example, an PDF document has five pages, and the new sequence just have two numbers which are 5 and 3.What the API SortPages do is set the new sequence as 5,3,1,2,4. See the sample code below:

C#

```
using RasterEdge.Imaging.PDF.dll;

public void SortPages(String filePath, List<int> pageOrder)

{

        PDFDocument doc = new PDFDocument(filePath);

        doc.SortPages(pageOrder);

}
```

As long as you use PDFDocument class, you will find many useful APIs will simplify your work. It's possible for you to replace some of the pages of the document with some of the pages of the other document. What you should pay an attention is the parameter pageIdx which begins with zero. That is if you input two as the page index, the third page actually updated.

C#

```
PDFDocument document1 = new PDFDocument(sourceFile);

PDFDocument document2 = new PDFDocument(destnFile);

BasePage page = document1.getPage(index1);

document2.updatePage(page, pageIdx);
```

PDF processing dll also gives you the right to insert pages to PDF document. The range of the parameter index is from zero to pageCount. If you input zero as the index, then the new page will insert to the PDF document as the first page. Other case, the new page will be inserted to the PDF document as the (index+1) page. Suppose you have a PDFPage class object named page, and then what you should do is imitate the code below:

C#

```
using RasterEdge.Imaging.PDF.dll;
```

```
public void InsertPageToFile(Stream s, int index ,PDFPage page)

{

        PDFDocument doc = new PDFDocument(s);

        doc.InsertPage(page, index);

}
```

Now you can add pages to PDF file, as well as delete pages of the PDF file. No matter a single page or multi-pages, no matter continuous or discontinuous pages, what you should do is provide the page index that you want to delete. If you want to delete the third page, the pageIdx you are supposed to input is two. Suppose you have a PDFDocument class object named document.

C#

```
document.DeletePage(int pageIdx);

document.DeletePages(List<int> pageIdx);
```

Introduction to PDF Annotations

RasterEdge Image enables developers to add reliable annotations to PDF document page. These objects include text, free hand, line, lines, ellipse, rectangle, rubber stamp, hotspot and embedded image. With the PDF processing dll and the Annotation processing dll provided by RasterEdge Image, developers can:

- Add rich annotations to PDF document
- Added annotations can be flexibly and easily edited as independent objects
- Free to resize, move, rotate and reshape on PDF page
- Adjust the font style of created text annotation on PDF document
- Able to burn generated image annotations onto the native file format as embedded images

To use functions above,the required assemblies are：

- RasterEdge.Imaging.PDF.dll
- RasterEdge.Imaging.Annotation.dll
- RasterEdge.Imaging.Basic.dll

About PDF Barcode

RasterEdge Barcode processing dll offers mature APIs for developers to generate, write and create 1d and 2d barcodes on PDF document page. You can easily adjust and customize most attribute properties of generated barcodes on PDF document file. Three steps to complete adding barcode to PDF page.

step1: Get the page from a PDF document

```
BasePage PDFDocument.GetPage(int pageIndex);
```

step2: Convert barcode to image

```
BaseImage BaseBarcode.ToImage();
```

step3: Add the barcode image to Page

```
void PDFPage.AddImage(REImage img, float positionX, float positionY);
```

Some APIs for scanning barcode from PDF document arestill provided. You may take three steps to complete you work.

step1: Get a PDFPage from a PDFDocument

```
BasePage PDFDocument.GetPage(int index);
```

step2: Convert PDFPage to REImage

```
BaseImage PDFPage.ToImage(int width, int height);
```

step3: Specify barcode type, offset and number to read

```
void BarcodeScanner.Scan(REImage img , BarcodeType type);
```

About PDF File Format

Adobe created the PDF file format to enable the encapsulation of any document that could be printed digitally so that it retains its content as text, images or graphics as high quality aspossible in typography as intended and accurate color representation. The file format is anobject-oriented format that describes a document as a series of pages, each of which is representedby a list of high-level drawing and compositing operations that are modeled afterthe PostScript imaging model. In addition to page content, a document could also includeinteractive features, navigation tools, dynamic forms, and multimedia. PDF is meant to bea publication format, rather than an editable format. Generation of PDF is considered to bea final step in the creation of a document.

RasterEdge Image provides the means to break into the PDF model. The details of PDF structure are hidden from client code. Instead, client code works with higher level objects, like PDFDocument and PDFPage.

How to's

How to: Create a PDF Document

PDF Assembly provides programmers two ways to create new PDF document. One is creating new PDF document with blank pages, and the other is creating an image only PDF. To create PDF

document, you can use the PDFDocument class. There are two constructors: one takes an input pageCount, and the other takes a list of REImage as input.

The example that follows demonstrates how to generate a new PDF document with blank pages.

C#

```
usingRasterEdge.Imaging.PDF;

public PDFDocument CreatePDFDocument(int pageCount)

{

        return new PDFDocument(pageCount);

}
```

The example that follows shows how to generate a PDF document only with image source.

C#

```
usingRasterEdge.Imaging.Basic;

using RasterEdge.Imaging.PDF;

public PDFDocument CreateImageOnlyPDFDocument(List<REImage> images)

{

        returnnew PDFDocument(images);

}
```

How to: Open a PDF Document

You can load a PDFDocument object from file or Stream using the APIs PDFDocument(String path) or PDFDocument(Stream s). The following examples show you how to do this.

C#

```
using RasterEdge.Imaging.PDF;

public void OpenFromFilePath(String filePath)

{

        PDFDocument document = new PDFDocument(filePath);

}
```

C#

```
using RasterEdge.Imaging.PDF;

public void OpenFromStream(Stream s)

{

        PDFDocument document = new PDFDocument(s);

}
```

How to: Save a PDF Document

After a PDF document has been read, created or modified, you can save it by invoking the save method in PDFDocument class. This method takes either an output Stream or an output filePath.

C#

```
using RasterEdge.Imaging.PDF.dll;

public void SavePDFToFile(PDFDocument doc, String filePath)

{

        doc.Save(filePath);

}

public void SavePDFToFile(PDFDocument doc, Stream s)

{

        doc.Save(s);

}
```

How to: Convert PDF Document to Image

To convert PDF document to image, first you should know how to get a PDFPage class object. You can use the API GetPage(int index) in PDFDocument. And the sample code below shows you how to achieve this.

C#

```
using RasterEdge.Imaging.PDF.dll;

public PDFPage getPageOfDocument(String filePath, int index)

{

        PDFDocument document = new PDFDocument(filePath);

        BasePage page = document.GetPage(index);
```

```
        return (PDFPage)page;
}
```

Once you have some PDFPage class objects, you can use the API ToImage(int width, int height) in PDFPage class to convert these objects to images.

C#

```
using RasterEdge.Imaging.PDF.dll;
public List<REImage>  ConvertToREImage(String filePath,)
{
        List<REImage> imageList = new List<REImage>();
        PDFDocument document = new PDFDocument(filePath);
        for(int i=0;i<document.GetPageCount();i++)
        {
                PDFPage page = (PDFPage)document.GetPage(i);
                REImage image = (REImage)page.ToImage();
                imageList.Add(image);
        }
        return imageList;
}
```

How to: Combine PDF Documents

To combine PDF documents, you can use the PDFDocument class. 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 PDF documents are combined in order to create one output PDF document. This is essentially a one line task which can be demonstrated in the following code.

C#

```
using RasterEdge.Imaging.PDF;
public void CombineDocument (List<PDFDocument>docList, String destnFilePath)
{
```

```
        PDFDocument.Combine(docList, destnFilePath);
}
```

How to: Split PDF Documents

It's easy for you to divide source PDF file into two smaller PDF documents use PDFDocument 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 the SplitDocument, then, the first three pages will be included in one PDF document and the rest five pages will be included in the other PDF file.

C#

```
using RasterEdge.Imaging.PDF;

public void splitPDFDocument(String sourceFilePath, int pageIdx, List<String> destnsPath)

{

        PDFDocument.SplitDocument(sourceFilePath, pageIdx, destnsPath);

}
```

How to: Extract Pages from PDF Document

Professional PDF page processing allows programmers to extract PDF pages that can be either continuous or discontinuous. You can extract multiple PDF pages use the static method called ExtractPagesFromDocument which takes three arguments, input sourceFilePath, a list of pageIdx and output destnFilePath or output destnStream. You can also extract a single PDF page use the static method called GetOnePageDocument which takes three arguments, input sourceFilePath, input index and output destnStream or output destnFilePath. Both methods presents in PDFDocument class object.

Using the comprehensive sample code below, you can extract multiple PDF pages from two different PDF files and combine these PDF pages into a new PDF file.

C#

```
using RasterEdge.Imaging.PDF.dll;

public void ExtractPagesToFormNewDocument(String filePath1, String filePath2, List<int> pageList1, List<int> pageList2, String destnPath)

{

   // two temporary strings to store the extracted documents file paths
```

```
   String tmpFilePath1 = @"c:\tmpFile1";

   String tmpFilePath2 = @"c:\tmpFile2";

PDFDocument.ExtractPagesFromDocument(filePath1, pageList1, tmpFilePath1);

PDFDocument.ExtractPagesFromDocument(filePath2, pageList2, tmpFilePath2);

   List<String> docPathList = new List<string>();

docPathList.Add(tmpFilePath1);

docPathList.Add(tmpFilePath2);

PDFDocument.Combine(docPathList, destnPath);

}
```

How to: Add Annotations to PDF Page

To add annotation on PDF Page, you should first create a graphical annotation. Different graphical annotations need different class objects to generate. These class objects present in the Annotation assembly. After creating a graphical annotation, you can use PDFPage class. There are two methods: one method is called AddFloatingItem which takes an input REItem, and the other method is called MergeItemsToPage which takes 0 arguments. Combining the two methods, you can achieve the result you want. REItem is a class object that presents in RasterEdge.Imaging.Basic.dll, which is an extension of BaseItem.

In the following part, we use the rectangle annotation as an example to demonstrate how to add annotation to PDF document.

C#

```
using RasterEdge.Imaging.PDF.dll;

using RasterEdge.Imaging.Annotation.dll;

public void DrawRectAnnotationOnPDFDocument(PDFDocument doc, int pageIndx)

{

        //generate a rectangle annotation and set its properties

         RectangleAnnotation rectAnn = new RectangleAnnotation();

         //unit is pixel

         rectAnn.X = 100;

         rectAnn.Y = 100;
```

```
        rectAnn.Width = 100.0F;

        rectAnn.Height = 100.0F;

        rectAnn.OutLine = new RasterEdge.Imaging.Annotation.Basic.LinePen();

        //fill the rectanglar with white color

        rectAnn.Fill = BrushGenerator.CreateBrush();

        PDFPage page = (PDFPage)doc.GetPage(pageIndx);

        REItem item = rectAnn.CreateAnnotationItem(page);

        page.AddFloatingItem(item);

        page.MergeItemsToPage();

}
```

How to: Add Image to PDF Page

In addition to adding annotations, you can also embed images. You can use the PDFPage class. There is a method called AddImage which takes an input REImage, an input positionX and an input positionY. REImage represents the image you want to embed.positionX and positionY represent the position on the page.

C#

```
using RasterEdge.Imaging.PDF.dll;

public void AddImage(String filePath, int pageIndex, String imagePath, float positionX, float positionY)

{

        PDFDocument document = new PDFDocument(filePath);

        Bitmap _bitmap = new Bitmap(imagePath);

        REImage image = new REImage(_bitmap);

        PDFPage page = (PDFPage)document.GetPage(index);

        page.AddImage(image, positionX, positionY);

}
```

Word

Introduction to Word Functions

With RasterEdge.Imaging.MSWordDocx.dll, RasterEdge Image gives programmers the right to do something included in the following items:

- Load an existing Word file
- Convert the pages of Word file to image
- Save the Word file or some pages of the file to all kinds of image formats.

Prerequisite for using these functions is to reference our Word processing dll (RasterEdge.Imaging.MSWordDocx.dll) to you project.

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 and it contains all document information of Word file. It is an extension of REDocument which is in turn an extension of BaseDocument.
- DOCXPage: This class refers to Word document page contained in DOCXDocument and uses REPage and BasePage as prototypes.

Word Document Project

At the heart of the Word processing dll is a class called DOCXDocument. This class represents the main structure of a document and the pages it contains. When a DOCXDocument object is created, we can easily extract information about the pages and other document structures.

How to's

How to: Load a Word Document

You can load an existing Word document by invoking the constructor of DOCXDocument .The constructor take an input String. And the parameter String presents the path of the Word file you want to open.

C#

```
using RasterEdge.Imaging.MSWordDocx.dll;

public void loadWordFile (String filePath)

{

        DOCXDocument document = new DOCXDocument(filePath);

}
```

How to: Convert Word Document to Raster Images

To convert Word document to raster images, first you should get the DOCXPage class object. To do this, you can use the API GetPage(int index) in DOCXDocument class. You should pay an attention if you input two as the index.In fact, you will get the third page of the document. The sample code below shows you how to do this.

C#

```
using RasterEdge.Imaging.MSWordDocx.dll;

public DOCXPage getPageOfFile(String filePath, int index)

{

        DOCXDocument document = new DOCXDocument(filePath);

        BasePage page = document.GetPage(index);

        return (DOCXPage)page;

}
```

Once you have some DOCXPage class objects, you can convert these objects to images use the API ToImage(int width, int height) in DOCXPage class. You can adjust the parameter width and height to control the size of these images. Suppose there is a DOCXPage class object called page, then the following example demonstrates how to convert this object to an image.

C#

```
using RasterEdge.Imaging.MSWordDocx.dll;

public void ConvertToImage(DOCXPage page, int width , int height)

{

        BaseImage image = page.ToImage(width, height);

}
```

A more advanced approach to get Bitmap image out of a PDFPage(or DOCXPage) is to render the page onto a REDrawingContext instance. And get bitmap from the context object. It allows you to control the resolution and offset of the images. Another point is that you can reuse the context object to generating images as many times as you like improve the rendering efficiency at the cost of memory. The following code is an example

```
// first construct a DrawingREContext object
DrawingREContext context = new DrawingREContext();
```

```
            page.RenderToContext(context);
            // set the resolution of the context a bigger resolution generate a larger images
            context.SetResolution(96, 96);
                /*  you can also set the region of the img you want to show. for example the
        following code specify a region which starts  at the point (0,0) with width and height
        both to be 100 pixels.*/
            context.SetValidRegion(0, 0, 100, 100, Units.PX);

            Bitmap img = context.GetBitmap();
```

How to: Save Word Document to TIFF

Once you know how to convert DOCXPage to image, it is no obstacle for you to save the image to all kinds of formats. In the following part, we take tiff as an example to show you how to perform this action.

C#

```
using RasterEdge.Imaging.MSWordDocx.dll;

using RasterEdge.Imaging.TIFF.dll;

public void SaveToTIFF(String filePath, int index, int width, int height, String destnFilePath)

{

        DOCXDocument document = new DOCXDocument(filePath);

        DOCXPage page = (DOCXPage)document.GetPage(index);

        REImage image = (REImage)page.ToImage(width, height);

        REFile.SaveImageFile(image, destnFilePath, TIFFEncoder());

}
```

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 and deal 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, we list all the features supported by the TIFF assembly.

- Load, create and save a TIFF file

- Provide three compressing interfaces to compress TIFF file
- Convert TIFF file to bmp, gif, png, jpeg, svg and PDF
- Add rich text and graphical annotations to TIFF file
- Rich APIs to process TIFF pages, like page extraction and adding image to page
- Free to add standard 1d & 2d barcodes to specified area of TIFF file

TIFF Programming Classes

- TIFFDocument: This class is an extension of REDocument which is in turn 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 in retrieving information from the files. Because of this, it is possible to reorder and remove pages from TIFF files. In addition, you can extract pages from other TIFFDocument objects to replace the pages of TIFFDocument that you are processing. Finally, TIFFDocument contains a set of static methods that can be used for splitting existing files and combining existing files to one file.
- TIFFPage: This class uses REPage and BasePage as prototypes. It is a high-level model of a page within a TIFF file.

TIFF File Basics

The TIFFDocument class can be used to manipulate images in a multipage TIFF file. And the partbelow provides code examples to demonstrate how to perform these actions.

Load a TIFF File

If you want to load TIFF file from file path, you can use the constructor which takes an input String.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public TIFFDocument LoadTIFFDocumentFromFilePath(String filePath)

{

        return new TIFFDocument(filePath);

}
```

If you want to load TIFF file from stream, then you can use the constructor which takes an input Stream.

C#

```
using RasterEdge.Imaging.TIFF.dll;
```

```
public TIFFDocument LoadTIFFDocumentFromStream(Stream s)

{

        return new TIFFDocument(s);

}
```

Create a TIFF File

TIFFDocument class gives you the right to create not only a TIFF file with blank pages but also a TIFF file from an image source.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public TIFFDocument CreateEmptyTIFFDocument(int pageCount)

{

        return new TIFFDocument(pageCount);

}

public TIFFDocument CreateImageOnlyTIFFDocument(List<REImage> images)

{

        return new TIFFDocument(images);

}
```

Save a TIFF File

After a TIFF File has been read, created or modified, you can save it by invoking the save method in TIFFDocument class. This method takes either an output Stream or an output filePath.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public void SaveTIFFDocumentToFile(TIFFDocument doc, String filePath)

{

       doc.Save(filePath);

}

public void SaveTIFFDocumentToFile(TIFFDocument doc, Streams)
```

```
{
        doc.Save(s);
}
```

Get the Number of TIFF Page

Once you have a TIFFDocument object, it is with no difficulty to get the number of pages of this object.

C#

```
using RasterEdge.Imaging.TIFF.dll;
public int GetPageCount(String filePath)
{
        TIFFDocument doc = new TIFFDocument(filePath);
        return doc.GetPageCount();
}
```

Get Page Object from a TIFF File

TIFFDocument is composed by a series of TIFFPage and many operations are performed on TIFFPage object. So the following example will show you how to get specified TIFFPage object from TIFFDocument by giving a page index.

C#

```
using RasterEdge.Imaging.TIFF.dll;
public void GetPageObject(Stream s, int index)
{
        TIFFDocument doc = new TIFFDocument(s);
        TIFFPage page = (TIFFPage)doc.GetPage(index);
}
```

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 sequence of the TIFF document. See the sample code below:

C#

```
using RasterEdge.Imaging.TIFF.dll;

public void SortPages(String filePath, List<int> pageOrder)

{

        TIFFDocument doc = new TIFFDocument(filePath);

        doc.SortPages(pageOrder);

}
```

Insert Pages to TIFF File

TIFF assembly also gives you the right to insert pages to TIFF document. Suppose you have a TIFFPage class object named page, and then what you should do is imitate the code below:

C#

```
using RasterEdge.Imaging.TIFF.dll;

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,no matter a single page or multi-pages, continuous or discontinuous. And what you should do is provide the page index that you want to delete. Suppose you have a TIFFDocument class object named document.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public void DeletePageFromDocument(TIFFDocument document, int pageIdx)

{

        document.DeletePage(pageIdx);

}

public void DeletePagesFromDocument(TIFFDocument document, List<int> pageIdxs)

{
```

```
        document.DeletePages(pageIdxs);
}
```

With DOCXDocument processing dll, splitting existing TIFF document into separate documents or combine any number of existing TIFF documents into a single document also becomes an easy task.

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

C#

```
using RasterEdge.Imaging.TIFF.dll;

public void splitTIFFDocument(String sourceFilePath, int pageIdx, List<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. This is essentially a one line task which can be demonstrated in the following code.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public void CombineDocumentAndSaveItToFile(List<TIFFDocument> docList, String destnFilePath)

{

        TIFFDocument.Combine(docList, destnFilePath);

}
```

TIFF Annotation

To help programmers add some comments and provide some additional information on TIFF page, RasterEdge Image expressly designs RasterEdge.Imaging.Annotation.dll. Here, we briefly list the main supported TIFF annotation types, which are text, free hand, line, lines, ellipse, rectangle, rubber stamp, hotspot and embedded image. With the TIFF processing dll and the Annotation processing dll provided by RasterEdge Image, developers can:

- Support adding over 10 annotation types to TIFF document
- Annotation created on TIFF image file can be processed as an independent image object
- Adjust the font style of created text annotation on TIFF document
- Able to burn generated image annotations on TIFF document page at the desired location

To use functions above,the required assemblies are：

- RasterEdge.Imaging.TIFF.dll
- RasterEdge.Imaging.Annotation.dll
- RasterEdge.Imaging.Basic.dll

TIFF Barcode

RasterEdge Barcode processing dll offers comprehensive functions for developers to generate and design both 1d & 2d barcode images on TIFF file. You can easily adjust and customize most attribute properties of the generated barcodes on TIFF file. Three steps to complete adding barcode to TIFF page.

step1: Get the page from a TIFF document

```
BasePage TIFFDocument.GetPage(int pageIndex);
```

step2: Convert barcode to image

```
BaseImage BaseBarcode.ToImage();
```

step3: Add the barcode image to Page

```
void TIFFPage.AddImage(REImage img, float positionX, float positionY);
```

Some APIs for scanning barcode from TIFF document arealso provided. You may take three steps to complete you work.

step1: Get a TIFFPage from a TIFFDocument

```
BasePage TIFFDocument.GetPage(int index);
```

step2: Convert TIFFPage to REImage

```
BaseImage TIFFPage.ToImage(int width, int height);
```

step3: Specify barcode type, offset and number to read

```
void BarcodeScanner.Scan(REImage img ,BarcodeType type);
```

How to's

How to: Load a TIFF File

If you want to load TIFF file from file path, you can use the constructor which takes an input String.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public TIFFDocument LoadTIFFDocumentFromFilePath(String filePath)

{

        return new TIFFDocument(filePath);

}
```

If you want to load TIFF file from stream, then you can use the constructor which takes an input Stream.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public TIFFDocument LoadTIFFDocumentFromStream(Stream s)

{

        return new TIFFDocument(s);

}
```

How to: Save a TIFF File

After a TIFF File has been read, created or modified, you can save it by invoking the save method in TIFFDocument class. This method takes either an output Stream or an output filePath.

C#

```
using RasterEdge.Imaging.TIFF.dll;

public void SaveTIFFDocumentToFile(TIFFDocument doc, String filePath)

{

       doc.Save(filePath);
```

```
}

public void SaveTIFFDocumentToFile(TIFFDocument doc, Streams)

{

        doc.Save(s);

}
```

How to: Update TIFF Page

To update TIFF page, you can use TIFFDocument class. There is a method called updatePage which is used to replace some of the document pages with some specified pages.

C#

```
using RasterEdge.Imaging.TIFF.dll;

publicvoid UpagePage(String sourceFile, int pageIdx, String destnFile, int index)

{

        TIFFDocument document1 = new TIFFDocument(sourceFile);

        TIFFDocument document2 = new TIFFDocument(destnFile);

        BasePage page = document1.getPage(index1);

        document2.updatePage(page, pageIdx);

}
```

How to: Add Image on TIFF Page

TIFF processing dll owns strong imaging functions, which can help users add a picture, image or logo to the arbitrary position of existing TIFF file. Developers are also entitled with ability to burn the added image to original TIFF file page for better displaying. In accordance with TIFF compression standards and formats options, programmers can add image or graphics of the formats such as png, bmp, gif and jpeg to a TIFF page. As TIFF file implements several compression mechanisms for embedded image, you can define any supporting type that you like. To add image on TIFF page, you can use the method called AddImage which presents in TIFFPage class. The method takes four arguments, namely REImage, positionX, positionY and TIFFCompression.

The following example demonstrates how to achieve it.

C#

```
using RasterEdge.Imaging.TIFF.dll;
```

```
public void AddImageToPage(TIFFPage page, REImage img, float X, float Y)
{
        page.AddImage(img, x, y, null);
}
```

How to: Convert TIFF Document to Raster Images

To convert TIFF document to all kinds of raster images, the first thing you need to do is to convert source TIFF file pages into a collection of image objects (REImage). After the REImage collection is created, you can output these created image objects to your desired image formats.

The example below shows how to convert TIFF document to JPEG.

C#

```
using RasterEdge.Imaging.TIFF.dll;
using RasterEdge.Imaging.Basic.dll;
public List<Stream> RenderTIFFDocumentToJPEG(String FilePath)
{
   // A list of stream which contains the image data of the JPEG image.
   List<Stream> buffer = new List<Stream>();
   TIFFDocument doc = new TIFFDocument(FilePath);
for (int i = 0; i < doc.GetPageCount(); i++)
   {
        TIFFPage page = (TIFFPage)doc.GetPage(i);
        REImage temp = (REImage)page.ToImage();
        MemoryStream imageStream = new MemoryStream();
        if (temp != null)
        {
                temp.Convert(imageStream, ImageFormat.Jpeg);
        buffer.Add(imageStream);
      }
```

```
    }
return buffer;
}
```

How to: Convert TIFF Document to PDF

To convert TIFF document to PDF, you are supposed to reference some assemblies to your project. They are:

RasterEdge.Imaging.Basic.dll;

RasterEdge.Imaging.TIFF.dll;

RasterEdge.Imaging.PDF.dll.

The steps in converting a TIFF file to PDF file are as follows:

1. Convert source TIFF file pages into a collection of image objects (REImage);
2. Invoke the PDFDocument(List<REImage>) constructor to create a PDFDocument .

C#

```
using RasterEdge.Imaging.TIFF.dll;
using RasterEdge.Imaging.Basic.dll;
using RasterEdge.Imaging.PDF.dll;
publicPDFDocument ConvertTIFFDocumentToPDF(String FilePath)
{
   // A list of stream which contains the image data of the TIFF image.
   List<REImage>imageBuffer = new List<REImage>();
   TIFFDocument doc = new TIFFDocument(FilePath);
for (int i = 0; i < doc.GetPageCount(); i++)
   {
      TIFFPage page = (TIFFPage)doc.GetPage(i);
      REImage temp = (REImage)page.ToImage();
imageBuffer.Add(temp);
    }
```

```
    return PDFDocument(imageBuffer);
}
```

How to: Add Annotations to TIFF Page

To burn annotation on TIFF page, you should first create an annotation. Different annotations need different class objects to generate. These class objects present in the Annotation assembly. After creating an annotation, you can use TIFFPage class. And there are two methods: one method is called AddFloatingItem which takes an input REItem, and the other method is called MergeItemsToPage which takes 0 arguments. Combining these two methods, you can achieve the result you want. REItem is a class object that presents in RasterEdge.Imaging.Basic.dll, which is an extension of BaseItem.

C#

```
using RasterEdge.Imaging.TIFF.dll;

using RasterEdge.Imaging.Annotation.dll;

public void TIFFAnnotation(TIFFDocument doc, int pageIndx)

{

        //generate a rectangle annotation and set its properties

         RectangleAnnotation rectAnn = new RectangleAnnotation();

         //set annotation size

         rectAnn.X = 100;

         rectAnn.Y = 100;

         rectAnn.Width = 100.0F;

         rectAnn.Height = 100.0F;

         //set annotation edge property

         rectAnn.OutLine = new RasterEdge.Imaging.Annotation.Basic.LinePen();

         //set annotation edge width

         obj.OutLine.Width = 5.0F;

         obj.OutLine.Brush = new AnnotationBrush();

         //set annotation edge style

         obj.OutLine.Brush.FillType = RasterEdge.Imaging.Annotation.FillType.Solid;
```

```
            //set the property of filled shape
            rectAnn.Fill = new AnnotationBrush();
            obj.SetTransparency(0.5f);
            BasePage page = doc.GetPage(pageIndx);
            REItem item = obj.CreateAnnotationItem(page);
            page.AddFloatingItem(item);
            page.MergeItemsToPage();
}
```

DICOM

DICOM Overview

Digital Imaging and Communications in Medicine (DICOM) is the standard format used to manage medical imaging information and its related workflow. Developed in 1993, the DICOM standard consists of a file format definition and a network communications protocol. DICOM is rapidly becoming the standard for all electronic health record systems that include imaging information as part of patient records. Adherence to the DICOM standard allows DICOM compliant devices from multiple manufacturers to work together in a seamless fashion as every DICOM complaint device must specify the DICOM classes it supports. Software developers who are DICOM conformant ensure that every medical imaging facility can use their software and that their tools can integrate with any electronic health records system. DICOM was developed by the DICOM Standards Committee and is managed by the Associationof Electical and Medical Imaging Manufacturers. For more information go tohttp:/DICOM.nema.org/(external link).

The DicomDecoder is an ImageDecoder that decodes DICOM images into a REImage. The basic information you need to create a RasterEdge Image Decoder for DICOM images is provided in the table below.

Assembly RasterEdge.Imaging.DICOM.dll

NameSpace RasterEdge.Imaging.DICOM

Use the TagDictionary to read tag from a DICOM image. Give Patients messages or this DICOM file messages. And there are new classes that allow manipulation of DICOM datasets and images closer to the raw formats provided by the file format.

Dicom Tags Dictionary

In this dictionary, you can get more basic information about patients and modify the date as well as other Tags related to image (See REImage or RasterEdge Image). Meanwhile, we provide you with the method to get value of a Tag.

Feature List

Dicom image is embedded in .dcm file. You can use our product to display the image in your viewer, save it to local disk or transmit it on the web. Below is a feature list of DICOM.

Compression Methods	Support Level
None	Support color; monochrome; Single frame; Multiple frame
RLE	Support color; monochrome; Single frame; Multiple frame
JPEG	Except for JPEG-Lossless
JPEG2000	Full support

How to…

You may do as follows to decode a Dicom image.

REImage img = REFile.OpenImageFile("Image.dcm", new DCMDecoder());

It will be returned as REImage if this DICOM image is single frame, or return as first frame for Multiple-FramesDicom image.

We also provide corresponding API to help you acquire multiframe-image and you can view more from following demo code.

Step1: You need to get FileIOMgr of one DICOM image file.

```
FileStream fs = new FileStram(@"D:\head.dcm", FileMode.Open);

FileIOMgr fileMgr = new FileIOMgr(fs);
```

Step2: Call API to decode DICOM image file.

```
List<BaseImage> dcmImageList = DCMDecoder.DecodeImageList(fileMgr);
```

Step3: You are offered several options in this step. The easiest way is to save the file to the local disk, or you can display it at web server.

```
int count = 0;

foreach(BaseImage tmp in dcmImageList)

{

        Bitmap bm = tmp.ToBitmap();
```

```
            bm.Save(@"D:\output\Frame"+count.toString() + ".png");
        }
```

You can acquire more information about this DICOM image file using following method.

```
Dictionary<String, String> dcmTagsValues = DCMDecoder.GetTagDictionary(fileMgr);
```

In DICOM terminology, there is a word called windows, which allows developers to adjust windows from different. We have provided similar API for you to have a free test and you are welcomed to E-mail your questions and suggestions to us.

To obtain a tag Dictionary, you may call TagDictionary() method.

Annotations

Introduction to Annotations

RasterEdge Annotation is a managed .NET Assembly that canperform annotationcapabilities to markup, draw, and visualize objects on an image or document. And these objectsinclude primitive shapes, text, freehand, sticky notes, redactions, images and hot spots.You can set properties of these annotations, produce a REImage, and perform further operations.

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.

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 the 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 the image in 90 degree increments.
- Individual points from annotations supporting points (Freehand, Polygon, etc.) can be repositioned.
- Able to change the shape of the object.
- Annotations can be burned onto the image with a single method.
- Annotation object can produce a 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

Generating an AnnotationObject

You can generate an Annotation object programmatically without a graphic interface.

When generating an annotation,a number of properties can be set. Following is an example to generate 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, sometimes a type conversion is needed.

First, add reference to `RasterEdge.Imaging.Annotation`

Example code:

using`RasterEdge.Imaging.Annotation`

How to Generate an Annotation Object Programmatically

C#

```
HotSpotAnnotation obj = newHotSpotAnnotation();
// set annotation size
                obj.X = 45.0F;
                obj.Y = 57.0F;
                obj.Width = 40.0F;
                obj.Height = 60.0F;

//set filled shape property
                obj.Fill = newAnnotationBrush();
                obj.Fill.FillType  =  RasterEdge.Imaging.Annotation.FillType.Solid;//set  filled
shape style
                obj.Fill.Solid_Color = Color.Gray;//set filled shape color

//set annotation edge property
                obj.OutLine = newAnnotationPen();
                obj.OutLine.Width = 2.0F;//set annotation edge width
                obj.OutLine.Brush = newAnnotationBrush();
                obj.OutLine.Brush.FillType  =  RasterEdge.Imaging.Annotation.FillType.Solid;//set
annotation edge style
                obj.OutLine.Brush.Solid_Color = Color.Blue;//set annotation edge color

//set filled shape property when annotation is active
                obj.ActiveFill = newAnnotationBrush();
                obj.ActiveFill.FillType   =   RasterEdge.Imaging.Annotation.FillType.Solid;//set
filled shape style
                obj.ActiveFill.Solid_Color = Color.Gray;//set filled shape color

//set edge property when annotation is active
                obj.ActiveOutLine = newAnnotationPen();
                obj.ActiveOutLine.Width = 2.0F;//set edge width
                obj.ActiveOutLine.Brush = newAnnotationBrush();
                obj.ActiveOutLine.Brush.FillType                                                =
RasterEdge.Imaging.Annotation.FillType.Solid;//set edge style
                obj.ActiveOutLine.Brush.Solid_Color = Color.Blue;//set edge color
```

Burn Annotation to Document or Image

Once you have an annotation object you can create a REItem object out of it, add it to a page in the document of your choice, and burn the annotation on to the document through this REItem object.

You can add and burn different annotations to documents like PDF, TIFF and Word. You can also burn annotation on to image of various formats using REImage objects.

How to Burn Annotation Object to PDF

Example Code:

Using RasterEdge.Imaging.Annotation;

Using RasterEdge.Imaging.PDF;

```
publicstaticvoidBurnAnnoationToPDF(String filePath, RectangleAnnotation annotation ,int pageIdx,
String annotatedFilePath)
```

```
        {

PDFDocument doc = newPDFDocument (filePath);

            PDFPage newPage = (PDFPage)doc.GetPage(pageIdx);



        // create an floating item from annoation and add to the PDFPage

            RasterEdge.Imaging.Drawing.REItemEx item = annotation.CreateAnnotationItem(newPage);
newPage.AddFloatingItem(item);
        // burn the annotion to the PDFPage
newPage.MergeItemsToPage();
        // save the revised document
doc.Save(annotatedFilePath);
        }
```

Import and Export Annotation

You can import and export annotation from/to an XML file.

Note: Since All Annotation classes are derived from a super Class called AnnoationBasic, the following APIs use this base class as the return type. In practice, you may need a specific sub class inherited from AnnoationBasic, and type conversion may be needed.

APIs:

```
public static AnnoationBasic AnnoationBasic.ReadProperty(XmlDocument xmlDoc);
public void AnnoationBasic.SaveProperty(XmlDocument xmlDoc, string rootName);
```

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.NETproject by copying a couple of lines of JavaScript. 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 may be 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 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 WinFormproject. You can draw, edit, and burn annotations to document or image using graphics interface provided in these viewer controls

See Also View Images or Documents.

TWAIN Scanning (Image Capture)

- TWAIN scanning overview
- Getting started with RETwain
- How to's

TWAIN ScanningOverview

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.

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 aForm after adding it to the toolbox, or you can instantiate it directly. This is the only class 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 objectsthat control numerous properties used for the image acquisition.

Device

The Device object provides full access to a TWAIN compatible source on the system. Use it toopen a connection to the device, to get and set properties, and then to acquire one ormore images. Because this class represents a system device resource, you cannot create aninstance of

it. You can obtain an instance to a Device object by calling GetAvailbleDevices, orfrom 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

```
publicvoid Scanning()
        {
Acquisition acq = newAcquisition();
            acq.ImageAcquired += newEventHandler<ImageAcquiredEventArgs>(acq_ImageAcquired1);
acq.Aquire();


        }
```

Setting UpEvents

You need to use events when acquiring images. When an image is acquired, the ImageAcquiredevent fires, providing an AcquireEventArgsobject that contains the image. At the very least,theImageAcquired event must be handled, and it is recommended that the AcquireCanceledand AcquireFinished events also be handled. The following code shows how the image ishandled.

```
this.acquisition.ImageAcquired += new EventHandler(OnImageAcquired);
private void OnImageAcquired(object sender, AcquireEventArgs e)
{
// If the image exists,save it to local file or do some operation
if(e.Image != null)
{
        REFile.SaveImageFile(e.Image, @"C:\test.png");
        e.Image = null;;// to release the image

}
}
```

Showing the "Select Source" Dialog

Your application should allow users to select which TWAIN device they want to use. This is accomplished bydisplaying the "Select Source" dialog using the `ShowSelectSource ()` method. The code below assumes the `Acquisition` component named`acquisition`.

C#

```
Device device = this.acquisition.ShowSelectSource();
```

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 aconnection resets all of the device properties to their default values and therefore a device should be closed afterthe 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 `Resolution`values of the device, and then closes the connection. This technique can be useful if you arelooking for a device of choiceswith specific default properties or capabilities.

C#

```
device.Open();
ResolutionData res = device.Resolution;
intbitDepth = device.BitDepth;
device.Close();
```

How to's

How to Do Console Based Scanning

How to Scan Many Pages to Form a PDFDocument

How to Do Console Based Scanning

Scanning from the Console is done similarly to scanning in a WinForms applicationwith a few exceptions:

1.) There is no SelectSourceDialog, and another method for selecting a scanner will need to be implemented.

2.) The program has to be prevented from continuing after acquisition. The demo below accomplishes this with the ModalAcquire property being set to true.

```
staticvoid Main(string[] args)
```

```
        {
Acquisition acquisition = newAcquisition();
AddEvents(acquisition);
count = 0;
DeviceCollection devices = acquisition.Devices;
Device selected = SelectDevice(devices);
            selected.HideInterface = true;
            selected.ModalAcquire = true;
Console.Out.WriteLine("---Beginning Scan---");
selected.Acquire();
Console.Out.WriteLine("---Ending Scan---\n Press Enter To Quit");
Console.In.Peek();
        }

privatestaticvoid AddEvents(Acquisition acquisition)
        {
            acquisition.ImageAcquired += newEventHandler (acquisition_ImageAcquired);
            acquisition.AcquireFinished += newEventHandler(acquisition_AcquireFinished1);
            acquisition.AcquireCanceled += newEventHandler(acquisition_AcquireCanceled1);
        }

staticvoid acquisition_AcquireCanceled1(object sender, EventArgs e)
        {
Console.Out.WriteLine("Acqusition Canceled");
        }

staticvoid acquisition_AcquireFinished1(object sender, EventArgs e)
        {
Console.Out.WriteLine("Acquisition Finished");
```

```
        }


staticint count;

staticvoid acquisition_ImageAcquired(object sender, ImageAcquiredEventArgs e)
        {

string filename = "out" + count++ + ".tif";

e.Image.Save(filename, ImageFormat.Tiff);

Console.Out.WriteLine("Frame " + count + " Acquired. Saved At: " + filename);

        }



privatestaticDevice SelectDevice(DeviceCollection devices)
        {

//TODO: Implement a Methodology for selecting a device

            }
```

How to Scan Many Pages to Form a PDFDocument

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: Raisedif the user cancels the acquisition process.
AcquireFinished: 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.

```
publicclassAcquisitionClass
        {
privatebool _acquireCanceled;
privateAcquisition myAcquisition = newAcquisition();
```

```
publicvoid ScanImages()
            {
                _acquireCanceled = false;
                myAcquisition.AcquireCanceled += newEventHandler(OnAcquireCanceled);
                myAcquisition.AcquireFinished += newEventHandler(OnAcquireFinished);
                myAcquisition.ImageAcquired += newEventHandler(OnImageAcquired);

Device activeDevice = myAcquisition.ShowSelectSource();

activeDevice.Acquire();
            }

privatevoid OnImageAcquired(object sender, ImageAcquiredEventArgs e)
            {
if (e.Image != null)
                {


PDFDocument doc = newPDFDocument("outputPDF.pdf");
PDFPage page = newPDFPage(e.Img);
doc.AddPage(page);

doc.Save("outputPDF.pdf");
doc.Dispose();

                }
            }

privatevoid OnAcquireCanceled(object sender, EventArgs e)
            {
                _acquireCanceled = true;
            }

privatevoid OnAcquireFinished(object sender, EventArgs e)
            {
if (_acquireCanceled)
return;

PDFDocument doc = newPDFDocument("outputPDF.pdf");
// do some extra work to finish compose the pdf document.


            }

        }
```