FAQ: How Do I Convert Between PDF and TIFF

Note: The PdfDecoder class is part of our PdfReader (Formerly called PdfRasterizer) module, which is an add-on to DotImage. To use this class, you must add Atalasoft.dotImage.PdfReader.dll as a reference to your project, and you will need a license file for this module. You can request an evaluation license file for this module using the Request Evaluation page of our web site. Simply deselect all options and select only "DotImage PDF Reader"

You will also want to add a PdfDecoder to your RegisteredDecoders collection in a static constructor for your class:

C#

```
tatic MyClass() { RegisteredDecoders.Decoders.Add(new PdfDecoder() { Resolution = 200 }); }
```

VB.NET

```
hared Sub New RegisteredDecoders.Decoders.Add(New PdfDecoder() With {.Resolution = 200}) End
Sub
```

Also, we have a complete, working PDF to TIFF demo and a complete, working TIFF to PDF Demo available which implement these techniques.

A common task that comes across in the document management world is converting from one file format to another. The two most commonly used formats for digital document images are PDF and TIFF. This task may confuse DotImage developer's because the PDF and TIFF codecs function slightly different from each other. Here I will explain the different approaches to this problem. First, I'll start by explaining the simple case, PDF to TIFF.

There are three possible to save a TIFF file in DotImage; The most memory efficient is demonstrated in the PDF to TIFF demo in our demo gallery. It uses a class called FileSystemImageSource which is passed directly to the PdfEncoder.Save method (along with a stream to save to)

C#

```
iffEncoder enc = new TiffEncoder(); using (FileStream fs = new FileStream("pathToSaveTo",
FileMode.OpenOrCreate)) { using (FileSystemImageSource fsis = new
```

```
FileSystemImageSource("PathToSourceTiff", true)) { enc.Save(fs, fsis, null); } }
```

VB.NET

```
im enc As New TiffEncoder() Using fs As New FileStream("pathToSaveTo", FileMode.OpenOrCreate)
Using fsis As New FileSystemImageSource("PathToSourceTiff", True) enc.Save(fs, fsis, Nothing)
End Using End Using
```

The other two approaches are still possible, but strongly discouraged in favor of using our ImageSource as outlined above. The reference here is kept for archival purposes.

The first is to have all of the images (pages of the TIFF file) loaded into memory at once, and pass them all to the TiffEncoder to save the file. As you can tell, this is not very memory efficient for large documents. The second way to save a TIFF file is to append the pages, one by one to an existing TIFF file. This way, we only need to keep a single image in memory at any given time.

As with all mutlipage formats, DotImage lets us read a single page from a file. So we can load each page from the PDF file as needed. As you can see from the following example, the first way is much easier to implement, but the second way will conserve a lot of memory.

C#

```
/ First way TiffEncoder enc = new TiffEncoder(); enc.Save(outStream,myImageCollection,null);
// Second way TiffEncoder noAppend = new TiffEncoder(TiffCompression.Default, true);
PdfDecoder pdf = new PdfDecoder(); for(int i=0; i< numPages; i++) { AtalaImage img =
pdfDecoder.Read(inStream, i, null); noAppend.Save(outStream, img, null); img.Dispose();
outStream.Seek(0, SeekOrigin.Begin); }
```

VB.NET

```
First way Private enc As TiffEncoder = New TiffEncoder()
enc.Save(outStream,myImageCollection,Nothing) ' Second way Dim noAppend As TiffEncoder = New
TiffEncoder(TiffCompression.Default, True) Dim pdf As PdfDecoder = New PdfDecoder() Dim i As
Integer=0 Do While i< numPages Dim img As AtalaImage = pdfDecoder.Read(inStream,i,Nothing)
noAppend.Save(outStream, img, Nothing) img.Dispose() outStream.Seek(0, SeekOrigin.Begin) i +=
1 Loop
```

That’s it! We have just converted a PDF file to a TIFF file. Now to save the TIFF as PDF. The

FAQ: How Do I Convert Between PDF and TIFF

PdfEncoder in DotImage does not allow us to save a single page to an existing PDF file, so we must have all the images ready when we save the file. But the good part is that these images don't necessarily need to be held in memory. The PdfEncoder has the standard overload to save an AtalaImage or ImageCollection, but it also introduces a new class called PdfImageCollection, which is made up of PdfImage's. This class gives us the flexibility to point to a file instead of an image in memory. To do this, simply create a PdfImageCollection that contains a PdfImage that points to the TIFF file that we want to convert.

C#

```
dfImageCollection col = new PdfImageCollection(); Col.Add(new PdfImage("TheDoc.tif", -1,
PdfCompressionType.Auto)); PdfEncoder enc = new PdfEncoder(); enc.Save(outStream,col,null);
```

VB.NET

```
rivate col As PdfImageCollection = New PdfImageCollection() Col.Add(New
PdfImage("TheDoc.tif", -1, PdfCompressionType.Auto)) Dim enc As PdfEncoder = New PdfEncoder()
enc.Save(outStream,col,Nothing)
```

Giving -1 as the frame index will force the entire TIFF image to be loaded. Converting to PDF isn't really any harder than converting the other way, it just requires a little more knowledge of the PDF namespace. Using these techniques, the ability to convert from TIFF to PDF, and vice versa, can be easily integrated into any document imaging application using DotImage.

Namespaces used in these examples:

```
talasoft.Imaging Atalasoft.Imaging.Codec Atalasoft.Imaging.Codec.Pdf
```

PDF to TIFF sample app

Original Article:

Q10125 - FAQ: How Do I Convert Between PDF and TIFF

Atalasoft Knowledge Base

FAQ: How Do I Convert Between PDF and TIFF

https://www.atalasoft.com/KB2/KB/50313/FAQ-How-Do-I-Convert-Between-PDF-and...