Background

DotImage's ImageSource classes are incredibly useful tools for dealing with multipage files in a memory-conscious way.

One of the most important uses of ImageSource classes is to completely replace/eliminate any instances of ImageCollection objects in your solution.

However, as one digs in to our offerings a bit, one may realize that our PdfImageSource takes a stream, but FileSystemImageSource needs physical files on disk... There are times when a FileSystemImageSource would be the correct choice if only it could take a Stream instead of needing files on disk.

There is no general use ImageSource that takes a stream or byte[] in DotImage.

Fear not: we can easily create a StreamImageSource by deriving from the RandomAccessImageSource abstract class

In order to use the example presented at the end of this article, you'll need to grab the StreamImageSource.cs file attached to this case and add it to your project using Add Existing...

Example of Simple StreamImageSource

```
rivate void ConvertToPdfUsingStreamImageSource(string outFile, string inFile) { /* for this example, we're starting from a stream, * we don't care if it's a MemoryStream or a FileStream... * the only restriction is that it must be a Seekable stream object */ FileStream inStream = new FileStream(inFile, FileMode.Open, FileAccess.Read, FileShare.Read); PdfEncoder pdfEnc = new PdfEncoder(); pdfEnc.SizeMode = PdfPageSizeMode.FitToPage; using (FileStream outStream = new FileStream(outFile, FileMode.OpenOrCreate)) { // Once you've added the StreamimageSource class to your project, // you just use it like any of our ImgeSource classes using (Atalasoft.Examples.StreamImageSource sis = new Atalasoft.Examples.StreamImageSource(inStream)) { pdfEnc.Save(outStream, sis, null); } } }
```

Handling byte[] With StreamImageSource

The implementation of StreamImageSource makes it incredibly easy to use it with a byte[] instead

Just pass it a byte[] of a supported image file type instead of a Stream - that's it ... nothing extra for you to do and now you can easily work with byte[] or Stream as you see fit.

Handling Multiple Streams

You may have noticed that the StreamImageSource does not have an option to take multiple streams or byte[]'s.

Do not be tempted to try modifying the StreamImageSource to accomplish this. Although this is possible, it makes the StreamImageSource somewhat more fragile/complex than it needs to be.

The solution: use a MergedImageSource to combine/handle multiple StreamImageSource objects - it just takes a tiny bit of extra work:

```
rivate static void ConvertToPdfUsingMultipleStreamImageSource(string outFile, string[]
inFiles) { // for this example we will make a Generic List of StreamImageSource objects //
which we will then use to construct a MergedImageSource passing the list in as an Array
List<Atalasoft.Examples.StreamImageSource> sources = new
List<Atalasoft.Examples.StreamImageSource>(); foreach (string file in inFiles) { Stream s =
new FileStream(file, FileMode.Open, FileAccess.Read, FileShare.Read); sources.Add(new
Atalasoft.Examples.StreamImageSource(s)); } PdfEncoder pdfEnc = new PdfEncoder();
pdfEnc.SizeMode = PdfPageSizeMode.FitToPage; using (FileStream outStream = new
FileStream(outFile, FileMode.OpenOrCreate)) { // Once you've added the MergedImageSource
class to your project, // you just use it like any of our ImageSource classes using
(MergedImageSource mis = new MergedImageSource(sources.ToArray())) { pdfEnc.Save(outStream,
mis, null); } } }
```

Handling Multiple byte[]s

The same technique for multiple streams could be converted quite simply for multiple byte[] objects. However, do keep in mind that when dealing with a large number of images / streams etc... that FileStream objects are significantly more memory-friendly than MemoryStream objects or byte[] arrays - so please use with care. Running into a System.OutOfMemoryException because you're unexpectedly holding multiple GiB worth of files as byte[]s would be a clear indication that a FileStream or file based approach is a better option.

Just use the Multiple Streams example above, and change this:

```
oreach (string file in inFiles) { Stream s = new FileStream(file, FileMode.Open,
FileAccess.Read, FileShare.Read); sources.Add(new Atalasoft.Examples.StreamImageSource(s)); }
```

to this:

```
oreach (string file in inFiles) { byte[] bytes = File.ReadAllBytes(file); sources.Add(new
Atalasoft.Examples.StreamImageSource(bytes)); }
```

Original Article

Q10424 - HOWTO: StreamImageSource - The Missing ImageSource option

Last Update

2026-11-04 - TD - (Fixed code fences and headings)

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50076/HOWTO-StreamImageSource-The-Missing-...