HOWTO: Separate a Multi-Page PDF With Automatic Repair of Damaged PDFs

Combining vs Separating

This article covers separating a single multi-page PDF into multiple single page PDFs. For help with combining multiple PDFs into a single multi-Page PDF (with Automatic Repair) please see the companion article:

HOWTO:Combine Multiple PDFs with Automatic Repair of Damaged PDFs

Background

DotImage 10.4 introduced a new feature to PdfDocument: the ability to repair many broken/corrupt PDFs

The Repair classes only take action when exceptions are encountered, so the ability to simply add the repair ability directly to any use of PdfDocument is as simple as choosing the constructor that takes RepairOptions. For PdfDocument, there are two such constructors:

When starting from a full path/filename:

```
public PdfDocument(
 string userPassword,
 string ownerPassword,
 string path,
 RepairOptions repairOptions
)
```

When starting from any seekable stream:

```
 PdfDocumentLoadedProgress pageLoaded,

 RepairOptions repairOptions

)
```

NOTE: in 11.2 we deprecated the older constructors that take non secure standard string in favor of SecureString. For newer DotImage, refer to the API reference for the overloads that take SecureString. Please see the attached PdfDocSeparateWithRepair_11.2.cs.zip for an updated 11.2+ class.

However, the PdfDocument.Separate() static convenience method does not have any option to add RepairOptions, thus it will simply throw an exception when encountering corrupt PDF files.

We've attached a static class called PdfDocSeparateWithRepair to this KB article. You can include it in your application and then use it like this:

Using File Name

Using File Name for Input and Directory Name with File Name Pattern for Output

```
tring userPassword = ""; // use empty string if not using passwords (SecureString required
for 11.2+) string ownerPassword = ""; // use empty string if not using passwords
(SecureString required for 11.2+) string source = ... full path and filename of PDF to split
... string destFolder = .. fully qualified directory path to write the output files to ...
string fileNameFormat = "out_{0}.pdf"; // string to format the output file names where {0}
will be replaced by frame index bool overwrite = true; // set to true to allow overwrite of
exsisting output files, false to not allow overwriting RepairOptions repairOpts = new
RepairOptions(); // you can specify various properties if you like
AtalasoftSupportUtils.PdfDocSeparateWithRepair.Separate(userPassword, ownerPassword, source,
destFolder, fileNameFormat, overwrite, repairOpts);
```

Using Stream

Using Stream for Input and PdfStreamCreator Delegate for Output

```
tring userPassword = ""; // use empty string if not using passwords string ownerPassword =
""; // use empty string if not using passwords Stream source = ... stream with source pdf
that will be split... creator ... see creator delegate definition below RepairOptions
repairOpts = new RepairOptions(); // you can specify various properties if you like
```

```
AtalasoftSupportUtils.PdfDocSeparateWithRepair.Separate(userPassword, ownerPassword, source, creator, repairOpts); public static Stream creator(Stream st, int page) { FileStream fs = File.Create(@"C:\pathto\output\fromStream" + page.ToString() + ".pdf"); return fs; }
```

Using a List<MemoryStream> for Output

Using Stream for Input and List<MemoryStream> for Output fnord

```
tring userPassword = ""; // use empty string if not using passwords string ownerPassword = ""; // use empty string if not using passwords Stream source = ... stream with source pdf that will be split... RepairOptions repairOpts = new RepairOptions(); // you can specify various properties if you like List splitStreams = AtalasoftSupportUtils.PdfDocSeparateWithRepair.Separate(userPassword, ownerPassword, source, repairOpts);
```

Utility Class and Sample App

There are two attachments to this KB article.

PdfDocSeparateWithRepair_class.zip contains just the CS source for the PdfDocSeparateWithRepair static class.

PdfDocSeparateWithRepair_class_11.2.zip contains just the CS source for the PdfDocSeparateWithRepair static class, but updated to provide new SecureString methods.

PDfDocumentSeparateWithRepair_SampleApp.zip contains that same class, but is actually used within a small console application as a small "proof of concept" for how to use the PDfDocSeparateWithRepair class. (This uses the old non secure string PdfDocSeparateWithRepair static class)

Original Article

Q10430 - HOWTO: Separate a Multi-Page PDF With Automatic Repair of Damaged PDFs

Last Update

2026-10-12 - TD - (Formatting and code fencing fixes)

HOWTO: Separate a Multi-Page PDF With Automatic Repair of Damaged PDFs

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50070/HOWTO-Separate-a-MultiPage-PDF-With-...