XDoc.PDF Developer Guide – Document Processing Module

Table of Contents

XDoc.PDF Developer Guide – Document Processing Module 1

Combine two or more PDF files to a new PDF file 3

Split a PDF file to two files 3

Split a PDF file to files 4

Split a PDF document by file size 4

Extract pages from an exist PDF file 5

Append all pages in a PDF file to the PDFDocument object 5

Add a page (from a PDF file) to a PDFDocument object 6

Add pages (from a PDF file) to a PDFDocument object 7

Insert a page (from a PDF file) to a PDFDocument object at specified position 7

Insert pages (from a PDF file) to a PDFDocument object at specified position 8

Insert an empty page to a PDFDocument object at specified position 8

Replace a page (in a PDFDocument object) by a page object 9

Delete a page in a PDFDocument object at specified position 10

Delete consecutive pages in a PDFDocument object 10

Delete pages in a PDFDocument object 11

Reorder all pages in a PDFDocument object 11

Swap two pages in a PDFDocument object 12

Extract pages from a PDFDocument object 12

Duplicate a page from a PDFDocument object 12

Extract pages (in a PDFDocument object) to build a new PDFDocument object 13
Merge two PDFDocument objects 13
Add an empty page to a PDF file 14
Add empty pages to a PDF file with the specified start page index and count 15
Delete pages in a PDF file 15
Delete consecutive pages in a PDF file with the specified start page index and count 16
Extract pages from a file to generate a new file 16
Rotate all pages in a PDF file 17
Rotate a specified page in a PDF file 17
Create an empty PDF file 18
Create an empty PDF document object 18
Duplicate pages from a PDFDocument object 19
Move a page to specified position 19
Split file by number of pages 20
Split file by output file size 20
Split file by bookmark 21
Create a PDF file from Bitmap object(s) 21

Combine two or more PDF files to a new PDF file

Combine two PDF files to a single PDF file

```
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
String[] inputFilePaths = new String[2] { inputFilePath1, inputFilePath2 };
//  Comebine two PDF files
PDFDocument.CombineDocument(inputFilePaths, outputFilePath);
```

Combine three PDF files to a single PDF file

```
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
String inputFilePath3 = Program.RootPath + "\\" + "3.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
String[] inputFilePaths = new String[3] { inputFilePath1, inputFilePath2, inputFilePath3 };
//  Comebine three PDF files
PDFDocument.CombineDocument(inputFilePaths, outputFilePath);
```

Split a PDF file to two files

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFileName = "Output";
int splitIndex = 1;     //  valid value: 1 to (Page Count - 1)

List<String> outputFilePaths = new List<String>();
outputFilePaths.Add(Program.RootPath + "\\" + outputFileName + "_0.pdf");
outputFilePaths.Add(Program.RootPath + "\\" + outputFileName + "_1.pdf");

//  split input file to 2 files, a file contains the first page of the input file,
//  and another file contains all remained pages.
PDFDocument.SplitDocument(inputFilePath, splitIndex, outputFilePaths.ToArray());
```

Split a PDF file to files

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFileName = "Output";
int[] splitIndex = new int[3] { 1, 3, 5 };    //  valid value for each index: 1 to (Page Count - 1)

//  create output file path list
List<String> outputFilePaths = new List<String>();
for (int i = 0; i <= splitIndex.Length; i++)
{
   outputFilePaths.Add(Program.RootPath + "\\" + outputFileName + "_" + i.ToString() + ".pdf");
}

//  split input file to 4 files:
//  file 0: page 0
//  file 1: page 1 ~ 2
//  file 2: page 3 ~ 4
//  file 3: page 5 ~ last page
PDFDocument.SplitDocument(inputFilePath, splitIndex, outputFilePaths.ToArray());
```

Split a PDF document by file size

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";

//  set split option

SplitOptions options = new SplitOptions(SplitMode.BySize);

//  limit the size of each file to 0.1M bytes

options.MaxSize = 0.1F;

//  set output option

SplitOutputOptions outputOps = new SplitOutputOptions();

outputOps.OutputFolder = Program.RootPath;

outputOps.Mode = 2;

outputOps.Label = @"Part";

outputOps.Separator = '_';

//  split a PDF file with options

PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

Extract pages from an exist PDF file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";

//  Selects 4 pages (page 0, 1, 3, 6) from the input file and form a PDF file with new order
//  Page 0 use page 1 of the input file
//  Page 1 use page 0 of the input file
//  Page 2 use page 6 of the input file
//  Page 3 use page 3 of the input file
int[] pageIndexes = new int[4] { 1, 0, 6, 3 };
PDFDocument.ExtractDocument(inputFilePath, pageIndexes, outputFilePath);
```

Append all pages in a PDF file to the PDFDocument object

```
String appendedPDFfilePath = Program.RootPath + "\\" + "2.pdf";
//  load the file that would be appended
PDFDocument appendDoc = new PDFDocument(appendedPDFfilePath);

String inputFilePath = Program.RootPath + "\\" + "1.pdf";
//  get PDFDocument object from a source file
PDFDocument doc = new PDFDocument(inputFilePath);
//  apply appending
doc.AppendDocument(appendDoc);

//  save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Add a page (from a PDF file) to a PDFDocument object

```
//  load the PDF file that provides the page object
String resFilePath = Program.RootPath + "\\" + "2.pdf";
PDFDocument resDoc = new PDFDocument(resFilePath);
//  get the 1st page in the document
PDFPage page = (PDFPage)resDoc.GetPage(0);

//  get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  append selected page to the end of the PDFDocument
doc.AddPage(page);

//  save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Add pages (from a PDF file) to a PDFDocument object

```
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
String outPutFilePath = Program.RootPath + "\\" + "Output.pdf";

PDFDocument doc1 = new PDFDocument(inputFilePath1);
PDFDocument doc2 = new PDFDocument(inputFilePath2);

//  get page 0, page 1 and page 3 from the first document
PDFPage page0 = (PDFPage)doc1.GetPage(0);
PDFPage page1 = (PDFPage)doc1.GetPage(1);
PDFPage page2 = (PDFPage)doc1.GetPage(2);
PDFPage[] pages = new PDFPage[3] { page0, page1, page2 };

//  append selected pages to the second document
doc2.AddPages(pages);
//  output the new document
doc2.Save(outPutFilePath);
```

Insert a page (from a PDF file) to a PDFDocument object at specified position

```
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
String outPutFilePath = Program.RootPath + "\\" + "Output.pdf";

PDFDocument doc1 = new PDFDocument(inputFilePath1);
PDFDocument doc2 = new PDFDocument(inputFilePath2);

//  get a page from first document
PDFPage page = (PDFPage)doc1.GetPage(0);

int pageIndex = 2;
//  insert selected page to the second document at the specified position
doc2.InsertPage(page, pageIndex);
//  output the new document
doc2.Save(outPutFilePath);
```

Insert pages (from a PDF file) to a PDFDocument object at specified position

```
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
String outPutFilePath = Program.RootPath + "\\" + "Output.pdf";

PDFDocument doc1 = new PDFDocument(inputFilePath1);
PDFDocument doc2 = new PDFDocument(inputFilePath2);

//  get page 0, page 1 and page 3 from the first document
PDFPage page0 = (PDFPage)doc1.GetPage(0);
PDFPage page1 = (PDFPage)doc1.GetPage(1);
PDFPage page2 = (PDFPage)doc1.GetPage(2);
PDFPage[] pages = new PDFPage[3] { page0, page1, page2 };

int pageIndex = 1;
//  insert selected pages to the second document at the specified position
doc2.InsertPages(pages, pageIndex);
//  output the new document
doc2.Save(outPutFilePath);
```

Insert an empty page to a PDFDocument object at specified position

```
//  get PDFDocument object from a file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

//  insert an empty page (with default page size) after the 2nd page
doc.InsertPage(2);

//  save the PDFDocument object to a file
String outPutFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outPutFilePath);
```

Replace a page (in a PDFDocument object) by a page object

C#

```
// load the PDF file that provides the page object
String resFilePath = Program.RootPath + "\\" + "2.pdf";
PDFDocument resDoc = new PDFDocument(resFilePath);
// get the 1st page in the document
PDFPage page = (PDFPage)resDoc.GetPage(0);

// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// replace the 3rd page by the PDFPage object
int pageIndex = 2;
doc.UpdatePage(page, pageIndex);

// save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

VB

```
' load the PDF file that provides the page object
Dim resFilePath As String = Program.RootPath + "\\" + "2.pdf"
Dim resDoc As PDFDocument = New PDFDocument(resFilePath)
' get the 1st page in the document
Dim page As PDFPage = resDoc.GetPage(0)

' get PDFDocument object from a source file
Dim inputFilePath As String = Program.RootPath + "\\" + "1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
' replace the 3rd page by the PDFPage object
Dim pageIndex As Integer = 2
doc.UpdatePage(page, pageIndex)

' save the PDFDocument
Dim outputFilePath As String = Program.RootPath + "\\" + "Output.pdf"
doc.Save(outputFilePath)
```

Delete a page in a PDFDocument object at specified position

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// delete the 3rd page
int pageIndex = 2;
doc.DeletePage(pageIndex);

//  save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Delete consecutive pages in a PDFDocument object

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// detele consecutive 3 pages from the 2nd page
int pageIndex = 1;
int pageCount = 3;
doc.DeletePages(pageIndex, pageCount);

// save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Delete pages in a PDFDocument object

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// delete 5 pages by their page indexes
int[] pageIndexes = new int[] { 1, 3, 5, 7, 9 };
//delete pages
doc.DeletePages(pageIndexes);

// save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Reorder all pages in a PDFDocument object

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// show page count of the document
int pageCount = doc.GetPageCount();
Console.WriteLine("Page Count: " + pageCount);

// define the new order for all pages
// 1. the length of the array MUST BE equal to pageCount
// 2. each page index SHOULD be in the array and only once
// otherwise, the method would throw exception
int[] pageOrders = new int[] { 1, 3, 0, 5, 4, 6, 2 };
doc.SortPage(pageOrders);

// save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Swap two pages in a PDFDocument object

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// swap two pages in the document
int pageIndex1 = 0;    // the first page
int pageIndex2 = 1;    // the second page
doc.SwapTwoPages(pageIndex1, pageIndex2);

// save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc.Save(outputFilePath);
```

Extract pages from a PDFDocument object

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";

PDFDocument doc = new PDFDocument(inputFilePath);

// extract pages by page indexes
int[] pageindexes = new int[] { 1, 2, 4 };
doc.ExtractPages(pageindexes, outputFilePath);
```

Duplicate a page from a PDFDocument object

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// duplicate the second page in the document
int pageIndex = 1;
PDFPage page = (PDFPage)doc.DuplicatePage(pageIndex);

// do something ...
```

Extract pages (in a PDFDocument object) to build a new PDFDocument object

```
// get PDFDocument object from a source file
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// select pages
List<int> pageIndexes = new List<int>();
pageIndexes.Add(2);    // the 3rd page
pageIndexes.Add(0);    // the 1st page
pageIndexes.Add(3);    // the 4th page
// create the new document with 3 pages
PDFDocument newDoc = (PDFDocument)doc.GetMultiDocument(pageIndexes);

// save the PDFDocument
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
newDoc.Save(outputFilePath);
```

Merge two PDFDocument objects

Merge two documents to a new document:

```
// get PDFDocument object from one file
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc1 = new PDFDocument(inputFilePath1);
// get PDFDocument object from another file
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
PDFDocument doc2 = new PDFDocument(inputFilePath2);

// merge two documents to a new document
PDFDocument newDoc = (PDFDocument)doc1.MergeDocument(doc2);

// ave the new document
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
newDoc.Save(outputFilePath);
```

Append another document:

```
//  get PDFDocument object from one file
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc1 = new PDFDocument(inputFilePath1);
//  get PDFDocument object from another file
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
PDFDocument doc2 = new PDFDocument(inputFilePath2);

//  append the 2nd document
doc1.AppendDocument(doc2);

//  save the document
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
doc1.Save(outputFilePath);
```

Add an empty page to a PDF file

Output to a new file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
int pageIndex = 0;
//  insert an empty page before the first page
PDFDocument.AddEmptyPage(inputFilePath, pageIndex, outputFilePath);
```

Overwrite the original file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
int pageIndex = 0;
//  insert an empty page before the first page
PDFDocument.AddEmptyPage(inputFilePath, pageIndex);
```

Add empty pages to a PDF file with the specified start page index and count

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
int startPageIndex = 1;
int numberOfPages = 5;
//  insert 5 empty pages after first page
PDFDocument.AddEmptyPages(inputFilePath, startPageIndex, numberOfPages);
```

Delete pages in a PDF file

Output to a new file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
int[] pageIndexes = new int[3] { 0, 2, 3 };
//  delete page 0, page 2 and page 3 from the file
PDFDocument.DeleteDocumentPages(inputFilePath, pageIndexes, outputFilePath);
```

Overwrite the original file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
int[] pageIndexes = new int[3] { 0, 2, 3 };
//  delete page 0, page 2 and page 3 from the file
PDFDocument.DeleteDocumentPages(inputFilePath, pageIndexes);
```

Delete consecutive pages in a PDF file with the specified start page index and count

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
int startPageIndex = 0;
int pageCount = 2;
//  delete first two pages
PDFDocument.DeleteDocumentPages(inputFilePath, startPageIndex, pageCount);
```

Extract pages from a file to generate a new file

Output to a new file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
int[] pageIndexes = new int[4] { 1, 0, 2, 3 };
//  reform a new PDF document with 4 pages
PDFDocument.ExtractDocument(inputFilePath, pageIndexes, outputFilePath);
```

Overwrite the original file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
int[] pageIndexes = new int[4] { 1, 0, 2, 3 };
//  reform a new PDF document with 4 pages
PDFDocument.ExtractDocument(inputFilePath, pageIndexes);
```

Rotate all pages in a PDF file

Output to a new file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
//  Rotate 180 in clockwise
int rotateInDegree = 180;
//  rotate all pages
PDFDocument.RotateAllPages(inputFilePath, rotateInDegree, outputFilePath);
```

Overwrite the original file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
//  Rotate 90 in clockwise
int rotateInDegree = 90;
//  rotate all pages
PDFDocument.RotateAllPages(inputFilePath, rotateInDegree);
```

Rotate a specified page in a PDF file

Output to a new file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
int pageIndex = 0;
//  Rotate 180 in clockwise
int rotateInDegree = 180;
//  rotate the first page
PDFDocument.RotatePage(inputFilePath, pageIndex, rotateInDegree, outputFilePath);
```

Overwrite the original file

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
int pageIndex = 0;
//  Rotate 270 in clockwise
int rotateInDegree = 270;
//  rotate the first page
PDFDocument.RotatePage(inputFilePath, pageIndex, rotateInDegree);
```

Create an empty PDF file

```
String outputFile = Program.RootPath + "\\" + "output.pdf";

//  create a PDF file with three empty pages
PDFDocument.CreatePDFFile(outputFile, 3);
```

Create an empty PDF document object

```
String outputFile = Program.RootPath + "\\" + "output.pdf";

//  Create a PDF Document object with 2 blank pages
PDFDocument doc = PDFDocument.Create(2);

//  Save the new created PDF document into file
doc.Save(outputFile);
```

Duplicate pages from a PDFDocument object

```
// get a PDFDocument object
String inputFilePath1 = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc1 = new PDFDocument(inputFilePath1);
// get another PDFDocument object
String inputFilePath2 = Program.RootPath + "\\" + "2.pdf";
PDFDocument doc2 = new PDFDocument(inputFilePath2);

// duplicate the 2nd page, 3rd page and 5th page from the first document
int[] pageIndexes = new int[] { 1, 2, 4 };
BasePage[] pages = doc1.DuplicatePages(pageIndexes);

// insert pages to the second document at the specified position
int pageIndex = 2;
doc2.InsertPages(pages, pageIndex);

// output the new document
String outputFilePath = Program.RootPath + "\\" + "output.pdf";
doc2.Save(outputFilePath);
```

Move a page to specified position

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";

// load the PDF file
PDFDocument doc = new PDFDocument(inputFilePath);

// move the 2nd page in the file
int moveFrom = 1;
// to the 6th position in the file
int moveTo = 5;
// move the page
doc.MovePage(moveFrom, moveTo);

// output the document
doc.Save(outputFilePath);
```

Split file by number of pages

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";

//  set split option
SplitOptions options = new SplitOptions(SplitMode.ByPage);
//  limit the pages of each file to 8 pages
options.MaxPages = 8;
//  set output option
SplitOutputOptions outputOps = new SplitOutputOptions();
outputOps.OutputFolder = Program.RootPath;
outputOps.Mode = 2;
outputOps.Label = @"Part";
outputOps.Separator = '_';
//  split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

Split file by output file size

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";

//  set split option
SplitOptions options = new SplitOptions(SplitMode.BySize);
//  limit the size of each file to 0.1M bytes
options.MaxSize = 0.1F;
//  set output option
SplitOutputOptions outputOps = new SplitOutputOptions();
outputOps.OutputFolder = Program.RootPath;
outputOps.Mode = 2;
outputOps.Label = @"Part";
outputOps.Separator = '_';
//  split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

Split file by bookmark

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";

//  set split option
SplitOptions options = new SplitOptions(SplitMode.ByBookMark);
//  set output option
SplitOutputOptions outputOps = new SplitOutputOptions();
outputOps.OutputFolder = Program.RootPath;
outputOps.Mode = 2;
outputOps.Label = @"Part";
outputOps.Separator = '_';
//  split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

Create a PDF file from Bitmap object(s)

C#

```
String imageFilePath = Program.RootPath + "\\" + "1.png";
String outputFile = Program.RootPath + "\\" + "output.pdf";

//  load image
Bitmap bitmap = new Bitmap(imageFilePath);

ImageToPDFSetting settings = new ImageToPDFSetting();
//  page size is same as the image size
settings.PageSize = SizeF.Empty;
//  use JBIG2 filter to compress monochrome image
settings.MonochromeCompression = PDFCompression.JBIG2Decode;
//  use Flate filter to compress grayscale image
settings.GrayscaleCompression = PDFCompression.FlateDecode;
//  use DCT filter to compress color image
settings.ColorCompression = PDFCompression.DCTDecode;
//  select quality level for DCT filter
settings.JPEGImageQualityLevel = JPEGImageQualityLevel.Highest;

//  create document
PDFDocument doc = PDFDocument.Create(bitmap, settings);

//  Save the new created PDF document into file
doc.Save(outputFile);
```

VB

C#

```
String[] imageFilePathes = new String[] {
    Program.RootPath + "\\" + "1.png",
    Program.RootPath + "\\" + "Test1bpp.bmp",
    Program.RootPath + "\\" + "Test8bppGrayScale.bmp",
    Program.RootPath + "\\" + "Test24bpp.bmp"
};
String outputFile = Program.RootPath + "\\" + "output.pdf";

//  load images
Bitmap[] bitmaps = new Bitmap[imageFilePathes.Length];
for (int i = 0; i < imageFilePathes.Length; i++)
{
    bitmaps[i] = new Bitmap(imageFilePathes[i]);
}

ImageToPDFSetting settings = new ImageToPDFSetting();
//  set page size to 16 inches * 12 inches
settings.PageSize = new SizeF(16F, 12F);
//  use JBIG2 filter to compress monochrome image
settings.MonochromeCompression = PDFCompression.JBIG2Decode;
//  use Flate filter to compress grayscale image
settings.GrayscaleCompression = PDFCompression.FlateDecode;
//  use DCT filter to compress color image
settings.ColorCompression = PDFCompression.DCTDecode;
//  select quality level for DCT filter
settings.JPEGImageQualityLevel = JPEGImageQualityLevel.Highest;

//  create document
PDFDocument doc = PDFDocument.Create(bitmaps, settings);

//  Save the new created PDF document into file
doc.Save(outputFile);
```

VB