Datalogics

Datalogics PDF2IMG™

User Guide

PDF2IMG version 4.7

For additional information, contact us at **http://www.datalogics.com/**.

Table of Contents

Introduction

Datalogics PDF2IMG is a conversion utility that allows you to transform pages in a PDF or XPS document into graphics image files like BMP, GIF or JPEG. You can also use PDF as one of the export formats. That means that you could take a 15-page PDF document and use PDF2IMG to turn it into 15 separate PDF documents, one PDF for each page in the original document. Alternatively the same 15-page PDF can be converted into a series of 15 different PNG files.

This product was created using the Adobe PDF Library, a Software Development Kit (SDK) based on Adobe Acrobat technology. The Adobe PDF Library is also available from Datalogics.

Deliverables

The installation includes:

1. pdf2img, the command line program.
2. Resources, dependencies used for proper rendering of PDF content.
3. pdf2imglib, the library itself.
4. pdf2img.vcproj, Visual Studio project file for command-line program

Win32 only: The legacy COM interface performs the same conversion task and options as the command line program using a COM object. It can be used within a .NET Framework application.

Installing the Software

Installing the PDF2IMG software is easy, simply run the installer on Windows or Linux.

Licensing

PDF2IMG is available for a free evaluation period. You can download and install it from our website and enter the activation key when prompted. The installation process generates a license file in your installation directory, and you will be ready to start using the software immediately. Please contact Datalogics to extend your evaluation.

Note that if you don't enter the activation key value when you first install the product, or enter it incorrectly, you will be prompted to enter the value again the next time you run the command line executable.

The license file should be stored in the same directory where you place pdf2img.

To generate a new license file, run your application that using pdf2img and enter the activation key again when prompted. Or you can copy the original license key file to the directory where you store your executable.

Note: The Windows installation adds the location of the pdf2img dependencies to the %PATH% Environment Variables, so you can run "pdf2img.exe" from anywhere.

Linux

On Linux, append to $PATH the location of pdf2img.

Note: GCC library v4.8 or higher is required to link against the libpdf2img.so shared library on Linux.

You can also use the "fontlist" command line argument to specify the location of font resources you want to use when you use PDF2IMG to complete a PDF document conversion.

Running

pdf2img offers several options for controlling Resolution, Color Model, Color Bit Depth and other settings, depending on output format selected.

For details on working with the library APIs, see the Appendix.

Command Line Syntax

The syntax is:

```
pdf2img [options] <inputFile> <outputFormat>
```

Only the **inputFile** and **outputFormat** arguments are required.

There are many other *optional* arguments to allow finer grain control over conversion.

Command Line Summary

Required Arguments

Argument name	Description
inputFile	Input PDF file name
outputFormat	Output graphic format–BMP/EPS/GIF/JPG/PDF/PNG/RAW/TIF

Optional Arguments

Use the following optional arguments in any order, preceding the required <inputFile> and <outputFormat> arguments. For example:

```
pdf2img -firstonly -colormodel=gray -bpc=1 -jpegquality=40 -resolution=72
input.pdf jpg
```

The two required arguments appear at the end, “input.pdf” as the file name (inputFile) and “jpg” as the output format (outputFormat).

Argument name	Description
asprinted	If specified, reverse the Annotation handling to suppress Image-only annotations and allow Print-only annotations
blackisone	If specified, reverse the PhotometricInterpretation setting to be black=1/white=0 *TIFF only*

Argument name	Description
blendingspace	Specify a blending color space by naming a profile description or providing a name and path of the profile (icc) file. *Default = CMYK*
bpc	Provide the number of bits used to represent each output color channel. *Default = 8*
colormodel	Define the color model, cmyk/gray/lab/rgb/rgba *Default=rgb*
colorprofile	Define the output ICC color profile. *Default determined by color space*
compression	Define the compression method, no/jpg/lzw/g3/g4 *Default=lzw*
digits	If provided, use to specify the number of digits to use in the sequential output filename counter
firstonly	Convert only the first page of the input file. *Default=all pages*
fontlist	If specified, provide a quoted semicolon-delimited list of alternate directories for font resources
help	If specified, provide a help list of available commands
ignoredefaultfonts	If specified, ignore default font resource locations when searching for fonts at start up (default directories and current directory)
ignorewarn	If specified, suppress warnings for non-renderable content
intent	Define the goal or priority for rendering intent for colors: perceptual/relative/saturation/absolute/profile *Default= profile if color profile is specified, otherwise perceptual*

Argument name	Description
jpegquality	Set the JPEG compression quality from 1 to 100 Higher values produce a better image but also a larger output file size. *JPEG only*, *Default=75*
maxbandmem	Define the maximum memory to use per band of multiband conversion output in bytes. *JPEG or TIFF only*, *Default=*300000000
multipage	If specified, produce one multipage TIFF output file of the requested name rather than the default of single-page sequentially named output files *TIFF only*
noannot	If specified, suppress viewable annotations
nocmm	If specified, suppress Color Management Module
noenhancethinlines	If specified, do not enhance thin lines when rendering
OPP	If specified, enable Overprint Preview (OPP) in output
output	Provide a prefix for output filename(s) to be created. *Default=input PDF file name plus sequence number*
outputintent	If specified, use the output intent dictionary value found in the source PDF document to define the output color profile to use when rasterizing a document
pages	If specified, provide a page or range of pages to process, such as *14-last* or *2-9*
password	If specified, enter a password string required to open the document for conversion
pdf-rasterize	Rasterize each page and export the content to a single PDF output file.
pdf-rasterize-split	Rasterize each page and export the content to a series of PDF output files, one for each page in the source document.

Argument name	Description
pdfregion	Define the region of PDF page to rasterize. Options include art, bleed, bounding, crop, media, or trim. *Default=crop*
pdf-split	Split a PDF document into a series of separate PDF documents, one PDF document for each page in the source file.
pixelcount	If specified, define an absolute picture size expressed as horizontal by vertical number of pixels. This can be used to specify a fixed number of pixels for the width and/or height of an output image, and thus scale the image up or down as needed to a specific output size and dimension.
profileCMYK	Provide the name and location of the CMYK color profile or description of the color profile to use for CMYK values specified in uncalibrated (device) color values in the input PDF file. *Default=Adobe Reader 9 CMYK*
profileGray	If specified, provide the name and location of the grayscale color profile or description of the color profile to use for grayscale values specified in uncalibrated (device) color values in the input PDF file. *Default=Gray Gamma 2.2*
profileRGB	If specified, provide the name and location of the RGB color profile or description of the color profile to use for grayscale values specified in uncalibrated (device) color values in the input PDF file. *Default=sRGB*
relaxParseSyntax	If specified, relaxes parsing restrictions to fix minor syntax issues in PDF source documents.
removewhitespace	If specified, direct PDF2IMG to remove the white space margins around the content on a page in the PDF document
resampler	Apply bicubic resampling: auto, bicubic, or none. *Default=auto*

Argument name	Description
resolution	Define output resolution, from 12 to 2400 in Dots per Inch *Default=300*
reverse	If specified, use to create a negative image. *Grayscale output formats only*
smoothing	Specify to apply Image antialiasing. Options include none, text, line, image, or all.

Arguments and Options

Note: You can always enter the command pdf2img -help to list details of the command line syntax.

Required

inputFile

```
inputFile
```

The inputFile argument is the name of your input PDF or XPS file. Specify a path to its location if it is not in your present working directory.

Note: By default, the software will place your output files in the same folder as your input file unless you specify otherwise using the -output option. See "Redirecting Output to Another Location." If an existing version of the same output file is stored in the output directory, PDF2IMG will overwrite that earlier version of the file with the new output file if the two files share the same name.

outputFormat

```
outputFormat
```

This argument defines the output graphic format you request: BMP, EPS, GIF, JPG, PDF, PNG, RAW or TIF.

Note: The limit for JPEG output image size is 65535 x 65535 pixels. TIF output has been tested up to a band of 68898 x 34449 pixels in size.

Optional

asprinted

```
-asprinted
```

By default, PDF2IMG renders the document to image format in the form as you would see it on screen, not as you would see it on paper. Normally, print annotations are omitted from the rendering process and only those annotations intended for viewing on screen are included, but -asprinted allows you to override that and reverse the distinction. That way you can render printable annotations, or those annotations which have been flagged as printable by the document author, when converting the document to an image.

Further, the use of -asprinted will suppress annotations flagged by the document author as for viewing on screen only (non-printing). Those will no longer appear in output images if this argument is selected.

Note: This command can be overridden by the -noannot flag.

blackisone

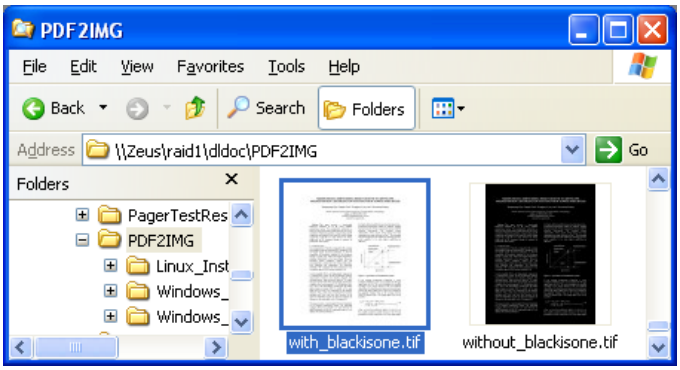
`-blackisone`

(*TIFF Only: Defaults to CMYK*)

Use the -blackisone argument to direct PDF2IMG to declare a reversed PhotometricInterpretation value of black=1; white=0 in the header of TIFF image file. The TIFF image file is defined with an output of 1-bit (black and white). This argument corrects a problem existing in some third-party PDF documents, where the converted TIFF output may unexpectedly display in reversed (negative) format when viewed via certain display utilities.

Consider the screenshot below. The white on black thumbnail image at right shows one such problem document which was converted without the -blackisone flag.

The corrected equivalent is on the left:



You normally will not need this switch. Also, -blackisone will have no visible effect on systems or utilities that read and understand the PhotometricInterpretation setting in the image header, since they will reverse their interpretation to correspond to that setting, and their output will look the same either way. This only affects systems having a fixed interpretation that does not agree with the original encoding of the image and gives you the ability to reverse the image encoding if needed.

Note: This switch is intended to correct a problem of unwanted display reversals seen in documents produced by some third-party PDF products. It is not intended for generating negative output. If you want to generate reverse or negative images, use the reverse switch.

blendingspace

```
-blendingspace=[description|.icc file name]
-blendingspace="Adobe RGB (1998)"
-blendingspace=c:\Windows\System32\spool\drivers\color\CoatedFOGRA39.icc
```

(*Default: CMYK*)

Users can specify a blending color space, by naming a profile description, or providing a name and path of the profile (.icc) file.

PDF files can have objects that are partially or fully transparent, and thus can blend in various ways with objects behind them. Transparent graphics or images can be stacked in a PDF file, with each one contributing to the result that appears on the page. With a stack of transparent images, the final colors shown are the result of blending the colors of all the overlapping objects. The flattening process merges a stack of transparent objects or graphics images into a single image on the page. Flattening images in a PDF file is necessary before you can render the page as a graphics image.

If a page in a PDF document has transparencies that need to be flattened, the page must go through an intermediate blending space before it can be rendered as a graphic output file. By default, that blending space is CMYK in PDF2IMG, but you can use the blendingspace option to select your own color profile for flattening transparencies, in the form of an icc file.

bpc

```
-bpc=[1 | 8]
-bpc=1
-bpc=8
```

(*Default: 8*)

This argument refers to the Image Depth expressed as Bits per Color channel, or the number of bits used to represent a color channel sample in the selected output format. Typically, the bpc is 8 for color or grayscale images, and 1 for black and white.

This dictates the number of different values or levels that each color channel may have, by specifying how many bits can be allocated for the color channel value. Thus, a bpc of 1 indicates that the color can only be either all present or all absent (such as black and white), since the single bit can only represent 0 or 1, the presence or absence of that color. A bpc of 8 indicates that eight bits are allocated for each color channel level, representing 256 gradients from None to full saturation (from 0 to 255) for that color.

Note: When producing TIFF g3 or g4 compressed output, you must include a colormodel argument (colormodel=gray) and a bpc argument (bpc=1).

In color images, each pixel is made up of three color channels, in the form of RGB (Red/Green/Blue), or four-color channels, as CMYK (Cyan/Magenta/Yellow/Black). Bit depth for each of these is always 8 and indicates that there may be 256 shades of each color.

colormodel

```
-colormodel=[cmyk | gray | lab | rgb | rgba]
-colormodel=gray
```

(*Default: rgb*)

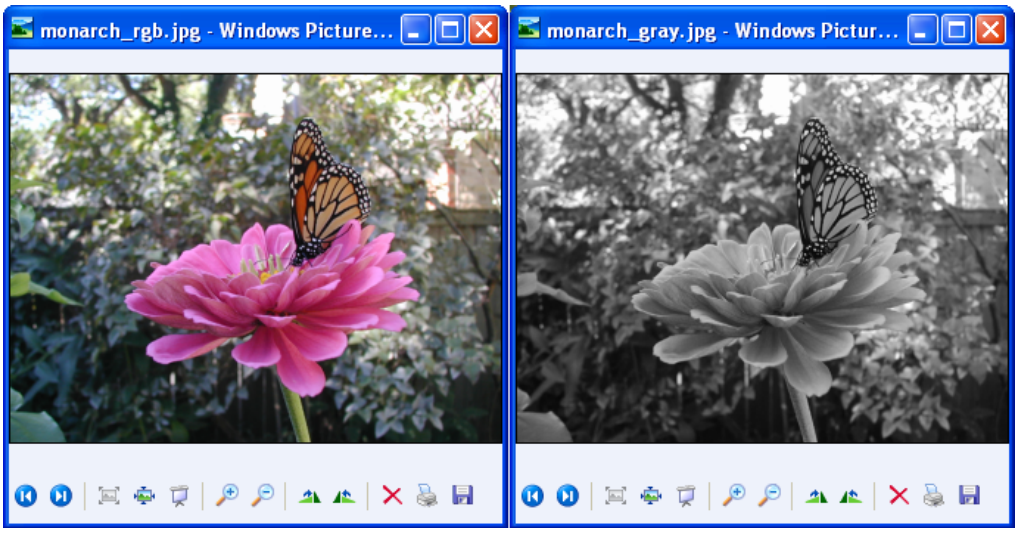
The colormodel will be cmyk, gray, lab, rgb, or rgba. The valid colormodel choices are determined by the output format selected.

Note: You can't define a color model for PDF or XPS output. If you select PDF or XPS as the output format and include the "-colormodel" option in your command line statement, the software will ignore it.

Output Format	Color Models Available
BMP	gray or rgb
GIF	gray or rgb
JPG	cmyk/gray/rgb
PNG	gray/rgb/rgba
TIFF	cmyk/gray/lab/rgb/rgba

For lab (valid in TIFF output only), 24-bit CIELAB images will be drawn, using 8 bits per channel. Values in that device-independent color model are relative to the D50 white point. PDF2IMG will return an error message if the requested colormodel is invalid for the selected format.

Note: When producing TIFF g3 or g4 compressed output, you must include a colormodel argument (colormodel=gray) and a bpc argument (bpc=1).



Note: Adobe Photoshop reads and writes CMYK JPEG files in a slightly non-standard format. As a result, if you use PDF2IMG to generate a JPEG in CMYK, the image may appear somewhat discolored when viewed in Photoshop.

colorprofile

```
-colorprofile =[filename or description]
-colorprofile=AdobeRGB1998.icc
```

(*Default determined by colorspace)*

Color profiles are standards for managing colors, used to guarantee that the colors for text or graphics in a file remain the same regardless of the hardware or software used to display, edit, or print that file. A color profile is usually included in the software or driver for an installed printer, scanner or other hardware device, or in software used to edit a file that is to be displayed or printed. The hardware device or software product uses the color profile to interpret the colors provided in a file, so that those colors can be presented accurately across more than one platform.

PDF2IMG selects default color profiles for both the input PDF and the target image rendering. The default input profiles (working spaces) apply to PDF elements that are not explicitly calibrated, such as DeviceCMYK, DeviceGray, DeviceRGB. The default output profile is chosen based on the target colormodel specified. The default ICC color profiles are provided in this table:

Color Space	**ICC Color Profile (Input)**	**ICC Color Profile (Output)**
CMYK	Adobe Reader 9 CMYK	Adobe Reader 9 CMYK
Gray	Gamma 2.2	Gray Gamma 2.2
L*a*b		CIE 1976 (l*a*b*) color specification with a D50 white point
RGB/RGBA	sRGB	sRGB

Use the colorprofile option to select the output profile, and profileCMYK/profileRGB/profileGray to select the input profile. The profile's filename refers to the name of the ICC Color Profile file, usually ending with an ".icc" suffix, as in "AdobeRGB1998.icc." If the program is calling an ICC file from another directory, provide the path name for the file as well.

For the colorprofile option, you will probably want to enter the ICC filename, but you can also enter the profile description, corresponding to the description field of the selected ICC profile, as in "Adobe RGB (1998)."

compression

```
-compression=[no | jpg | lzw | g3 | g4 ]
-compression=no
-compression=jpg
```

```
-compression=lzw
-compression=g3
-compression=g4
```

(*TIFF Only*)

(Default: lzw)

Note: When producing TIFF g3 or g4 compressed output, you must include a colormodel argument (colormodel=gray) and a bpc argument (bpc=1).

Use this option to specify output compression of TIFF images as needed. Valid values depend on the type of TIFF images being processed.

For color images you can select "no" (no compression), jpg, or lzw, to turn compression on or off.

For black and white images (1 channel, 1 bit), you can also specify g3 or g4 compression.

digits

```
-digits=[0 to 9]
```

The -digits command line argument allows you to specify the number of digits to be used for the sequential output filename numbering suffix. For example, if you enter:

```
-digits=3
```

as part of a command, the export files generated from your input PDF document will be named FILE001.JPG, FILE002.JPG, and so on.

If you don't use the "-digits" argument no numbers will be added to the export file names.

Normally, leading zeroes are only added as required to maintain the sorting order of the files. That is, a PDF input file with 200 pages will be processed to create a series of JPEG or PNG output files with names like FILE_001.JPG and FILE_002. But you can use the digits argument to force a specific number of digits regardless of the input page count. So, if you enter "-digits=4" the system will generate graphics output files with file names like FILE0001 and FILE0002, even if the PDF input file only has 12 pages, making the extra leading zeroes, strictly speaking, unnecessary.

Note: If you use the digits command line argument the system will not include the underscore character ("_") in the output file names, even though this would normally come before the sequence number otherwise (as in "FILE_001.JPG").

If you set the -digits value equal to zero (-digits=0) the system will return to normal sequential numbering and file naming logic.

If you enter a -digits value equal to one (-digits=1) the system will not add any leading zeroes to the output file names, and it will also suppress the use of the underscore character (FILE1.JPG, FILE2.JPG).

Note: You must specify a -digits value at least as high as what the input file would require by default. For example, a 200-page input file requires a "-digits" value of 3 or higher (-digits=3). No error will occur, but later file names in the output sequence may be one or more digits longer than earlier file names as a result, leading to problems listing the files in order. For example, you might see FILE98, FILE99, and FILE100 instead of FILE098, FILE099, and FILE100.

firstonly

```
-firstonly
```

The firstonly command directs PDF2IMG to convert only the first page of the input PDF file. Otherwise, the software converts every page found in the document. This option does not accept a value.

fontlist

```
-fontlist="directory1;directory2;directoryN"
-fontlist="C:\Test\Client\Fonts"
-fontlist="C:\Alternate\Fonts\Test;C:\Application\Resources"
```

(Adobe Systems standard search locations)

The -fontlist argument passes to the PDF2IMG system an alternate list of directories where the system can find font files. PDF2IMG will use the first instance of each font that it finds.

The following rules apply:

- The values included in the -fontlist argument--that is, the names of font directories--must be separated by semicolons.
- Wildcard characters such as tildes (~) or asterisks (*) are not allowed.
- The list of values must be enclosed in double quotes.
- You can list up to 16 locations with the -fontlist argument.

You generally do not need the -fontlist option unless you are using font files not actually installed on your machine. For example, this might be font files that are stored on your machine but not installed or recognized as fonts by the operating system. Also, you might want to use the -fontlist option if you have also specified the -ignoredefaultfonts flag with a PDF document that may not have embedded all its necessary font resources.

In addition, the -fontlist option will replace the default search for fonts. For example, if the -fontlist option is *not* given, PDF2IMG on a Windows machine will look for font files in the folders provided with the PDF2IMG software installation:

```
C:\Program Files\Datalogics\PDF2IMG Pro\Resources\CMap
C:\Program Files\Datalogics\PDF2IMG Pro\Resources\Font
```

Regardless of whether the -fontlist option is given, both Windows and UNIX versions always search the following locations, relative to the present PDF2IMG installation directory, unless the command also specifies the -ignoredefaultfonts flag:

- Resources
- Resources/CMap
- Resources/Font

Note: A list of directories given here replaces the default Adobe search locations; it does not append to them. Platform default resource locations such as C:\Windows\Fonts or similar on Windows are always searched. The -fontlist argument only overrides the Adobe location search.

help

```
-help
```

If you would like to list basic information about the proper syntax and accepted values for every one of the PDF2IMG command line arguments, enter the -help option. This option does not accept a value.

ignoredefaultfonts

```
-ignoredefaultfonts
```

Normally, when PDF2IMG starts it searches for local font resources, looking in the default system font directories and the current working directory. Use the -ignoredefaultfonts option to suppress this search. Instead, the system will only use the resources specified in the supplied -fontlist option statement. This option does not accept a value.

ignorewarn

```
-ignorewarn
```

The -ignorewarn command suppresses the warning normally returned whenever non-renderable content is encountered in the input file. This option does not accept a value.

intent

-intent=**[perceptual | relative | saturation | absolute | profile]**

(Default: profile *if a profile is supplied; otherwise,* perceptual*)*

-intent=relative

Use the -intent option to specify the color translation method for colors that are outside the gamut of the color profile. The intent feature is useful if you are converting a PDF page to a graphic file. If the resulting file includes a color or colors that cannot be represented directly on a specific hardware device, the intent lets the PDF2IMG software determine how to substitute a color that can be written to the file.

With the intent option, when you use PDF2IMG to convert a PDF document to a graphic file or files, you can select from a list of standard strategies to apply when converting the colors in that original PDF document. Thus, when you print or display a graphic output file, the colors in the output file will match as closely as possible the original color found in the source PDF document.

You can provide both color profile and intent options with your request, or either one. You do not need to define a color profile with your intent option. See the description of "Rendering Intents" in ISO 32000-1:2008, Document Management-Portable Document Format-Part 1: PDF 1.7, section 8.6.5.8, page 154.

Values you can enter for intent include:

Value	Description
perceptual	Generally used for photography. This method does not map colors one for one but estimates to match colors. Hence it often provides the most pleasing result but not necessarily the most accurate. If you do not specify a color profile in the "colorprofile" option, the intent value defaults to perceptual.
relative	Generally used for photography. The relative method uses an algorithm to select the closest possible color map to be true to the specified color.
saturation	Commonly used in charts and diagrams with a limited palette of colors where hue is not as important.
absolute	Often used to select a specific color or set of colors for drawings or designs. For PDF2IMG absolute will serve to reproduce the exact colors provided in the original PDF document. A common reason for using absolute would be to reproduce the color used in a corporate logo such as IBM Blue. The color is changed by selecting a defined match. This method does not use a conversion algorithm to select the closest color available.
profile	If you specify a color profile in the color profile option the intent value defaults to profile. In that case PDF2IMG will use the rendering intent provided with the ICC color profile currently in use.

For example, the Adobe RGB 1998 color profile uses Relative Colorimetric as its rendering intent. If PDF2IMG specifies Adobe RGB 1998 as the color profile (and an alternate intent option is not specified) the PDF2IMG software will use relative.

jpegquality

```
-jpegquality=[1 to 100]
-jpegquality=50
```

(Default: 75)

The jpegquality option is a value from 1 to 100, representing the quality/size value for the JPEG compressor. A higher value will produce a higher quality image, though also a larger output file. Lower values will produce lower-quality images but smaller and more efficient output graphic files.

Note: Lowering the JPEG Quality value will not only lower the detail of the image, but also lower the precision of the colors as compared with the original input. For example, rendering a JPEG image at 50% quality rather than some value significantly higher may yield a result that not only shows less detail but also contains slightly different shades of color.

maxbandmem

```
-maxbandmem=[100000000 to 2100000000]
```

(JPEG or TIFF only; Default: 300000000*)*

When generating JPEG or TIFF output, PDF2IMG checks to see if it has enough memory to rasterize a PDF page in one pass. If not, it rasterizes the page in bands (strips) to use less memory, then reassembles the bitmaps into the finished output image. This banding approach (if needed) will have no effect on the final output appearance and will be transparent to the user; the memory allocation or banding is handled internally.

You will typically not need this call. The -maxbandmem option allows you to fine tune the process to convert PDF document pages to JPEG or TIF files, if you find that your application's performance is enhanced by making the size of the rendering bands either larger or smaller within the technical limits of your machine.

multipage

```
-multipage
```

(TIFF only; No default)

Normally, processing a multipage PDF input file will result in a series of sequentially ordered, single-page output files, each carrying the given or default output file name prefix, an underscore ("_") and a sequential number. If performing a conversion to TIFF, -multipage directs PDF2IMG to produce one, multipage TIFF output file instead.

noannot

`-noannot`

Specify -noannot in a command line to prevent displayable annotations from appearing in output files.

Note: Use -noannot with care. Many page objects can be various forms of annotation, some more obvious than others, so you should check your output carefully to ensure that you are suppressing only those annotations that you want to block.

Note: This command will override the -asprinted option.

nocmm

`-nocmm`

The -nocmm option suppresses the use of the Color Management Module (CMM) and embedded color profiles during conversion to selected output graphic image formats.

For PDF2IMG, color management is normally in effect, so the product assumes an output profile of Adobe Acrobat CMYK, sRGB, or Gamma 2.2 (as appropriate for the output format) and will assume that Device colors on input are calibrated as Adobe Acrobat CMYK, Adobe 1998 RGB, or Gamma 2.2 respectively. When generating TIF, JPEG, PNG or BMP output, PDF2IMG will embed the corresponding profile in the output image.

When -nocmm is s specified, PDF2IMG will draw the output to a device color, embed no profile in the image written, and presume no default color model for input device colors.

noenhancethinlines

`-noenhancethinlines`

PDF2IMG uses the "Enhance thin lines" rendering option by default. This process is also found in Adobe Reader and Adobe Acrobat. If this is not the effect you want, use the -noenhancethinlines option to turn it off.

OPP

```
-OPP
```

(Default: false)

The OverPrint Preview (OPP) option allows you to generate a graphic that represents the Overprint content that would otherwise be lost during the rendering process. The graphic will show what the input PDF page would look like after being printed, accounting for ink overprinting. See the pdf2img_set_OPP reference.

output

```
-output=[filename]
-output=alternate_name
-output="C:Converted_Filesalternate_name"
```

(Default: Input PDF file name plus sequence number)

Use -output to define the prefix to add to the name of the output file or files you seek to create. The value given here will be used for the output file name, with a sequence number and an appropriate extension appended to indicate the output file type, such as sample_1.gif and sample_1.jpg. For a PDF input document with more than one page, the system will create sequential, separate output files for each page of the input file, with a sequence number appended to each, such as sample_1.gif, sample_2.gif, sample_3.gif, and so on.

See “Multi-Page Processing.”

You can also use the output option to redirect output files to the local or server directory that you select. This is described below.

Note: The Underscore character ("_") that normally precedes the sequence number is not inserted if the -digits command line argument is used.

If you do not include the -output option in a command, PDF2IMG will assign the name of the input PDF file name to each output graphic file and add a sequence number. For example, if you are starting with a five-page PDF document called Test.PDF, and are exporting the pages to JPEG files, the five output files will be named Test_1.JPG, Test_2.JPG, and so on.

Redirecting Output to Another Location

You can use the -output option to save your output file or files to another location, but the following notes apply:

- If you copy an output file to a folder, and the folder already has an existing output file with the same name as the new file, the new file will overwrite the existing file in that directory.

- If you want to store your output files in a folder that you choose, you must create that folder first.
- You need to provide the name of your output file along with the path name of the folder where you want to save that file. When you redirect output files to your own directory, PDF2IMG will not use the name of the input file by default.

outputintent

```
-outputintent [description]
-outputintent=GTS_PDFX
```

This option defines the output color profile for PDF2IMG to use when rendering a page from a PDF document to a rasterized graphic image. The argument tells the software to find the profile to use in the output intent stored in the PDF document itself.

An output intent is set of dictionaries stored in the PDF document's OutputIntents array. More than one output intent may be imbedded in an output intent array within a PDF document, each with its own color profile and characteristics. This allows the PDF document to adapt to a variety of workflows or production environments. Each output intent features several dictionary key values, including OutputCondition and an OutputConditionIdentifier. Both values are text strings that describe the intended output device for this PDF document or the production environment.

The output intent dictionary also offers a SubType (S) dictionary key, an optional value that further describes the PDF document format.

Three output intent subtypes are defined:

- GTS_PDFX for PDFX, or Graphics Exchange documents
- GTS_PDFA for PDF Archive documents
- ISO_PDFE for PDF Engineering documents

See the description of "Output Intents" in ISO 32000-1:2008, Document Management-Portable Document Format-Part 1: PDF 1.7, section 14.11.5, page 633.

When you create the outputintent argument in a PDF2IMG command statement, the Description variable will be a description of a color profile embedded in an OutputCondition, OutputConditionIdentifier, or Subtype dictionary.

```
pdf2img -outputintent=JC200103 TestFile.pdf tif
```

PDF2IMG looks at the OutputConditionIdentifier entries in the output intent array for the color profile JC200103. Then, PDF2IMG applies that profile when converting a PDF document called TestFIle.PDF to a TIF file.

This command:

```
pdf2img -outputintent=GTS_PDFA sample_jc.pdf tif
```

Tells PDF2IMG to use the color profile used with the PDFA subtype and create a TIF file from TestFile.PDF. The conversion will be governed by the requirements defined for the PDF Archive format.

If you don't specify an input color profile or an output color profile, PDF2IMG will assign a default color profile for each. You can use profileCMYK, profileRGB, or profileGray to assign an input color profile when you convert a PDF document to graphic images. To select a specific color output profile to use when converting a PDF document page to a graphics file, you can use the colorprofile argument. Or you can use the outputintent argument to find and use a color profile embedded in the PDF document itself.

PDF2IMG will use the color profile specified with the outputintent argument for both the output color profile *and* the input color profile unless you define an input color profile using profileCMYK, profileRGB, and/or profileGray. You can use the color profile values (such as colorprofile or profileCMYK) and an outputintent value in the same argument. If PDF2IMG cannot find the color profile specified in the outputintent argument within the PDF document, it will ignore the outputintent statement and select a different color profile or profiles to use. Either it will use the color profiles offered using the colorprofile argument or profileCMYK/RGB/Gray, or it will select the appropriate default color profile.

pages

```
-pages=[range]
-pages=3
-pages=2,5
-pages=1-10
-pages=1,3,5-9,21
-pages=15-last
```

Normally, when PDF2IMG processes a PDF document it converts every page found in that document. Use the -pages option to specify a list of selected input pages to process. You can give single page numbers, separated by commas; the first and last page of a range, separated by a hyphen; or some combination of the two.

Pages are identified by their sequential order within the file, not by their folio. For example, you would specify "2" for the second page of the input file, even if the document identifies it as "Page 57" or "page xiv" on paper. If you are working with a PDF document that has a cover sheet, a page with copyright data, and four pages of a table of contents, the first page of the document, page 7, would actually be page 7 in the file. So, in this case you would specify "7" for page one in this document. Page ranges must be given in increasing order; don't enter a command like "-pages=15,19-21,5." Do not include spaces.

If you do not know the exact page count of the input file, use the keyword "last" to indicate a page range that should run to the end.

password

```
-password=[string]
-password=sesame
```

Use the -password option to pass the User or Owner password needed to open a password-protected PDF document. The password may be any character string up to 127 characters in length, without spaces.

Note: A document with a User password but no other restrictions can be processed by providing the User password. If the PDF document has any security beyond a User password to restrict viewing, you will need to specify its Owner password instead. Opening with an Owner password will override the document's security restrictions and allow PDF2IMG conversion to proceed.

pdf-rasterize

```
-pdf-rasterize
```

Use this option to rasterize each page in a PDF input document into graphics images and then save these pages to a single PDF output file.

A rasterized graphic uses pixels, or points of color, to create a bitmapped graphic image, expressed in Dots per Inch. A common example would be a photograph, presented as a JPG file. Standard PDF documents are vector documents, where the text or images are based on a mathematical formula.
If you want to convert the pages in a PDF document to a series of graphic files, such as PNG or TIF, you are said to be "rasterizing" that PDF document.

In effect, with this option PDF2IMG converts the input PDF document into a series of graphic image files, and then saves these rasterized image files into a single PDF output file.

pdf-rasterize-split

```
-pdf-rasterize-split
```

Use this option to rasterize each page in a PDF input document into graphics images and then save these pages to a series of PDF output files, one PDF output file for each page in the original document.

A rasterized graphic uses pixels, or points of color, to create a bitmapped graphic image, expressed in Dots per Inch. A common example would be a photograph, presented as a JPG file. Standard PDF documents are vector documents, where the text or images are based on a mathematical formula.
If you want to convert the pages in a PDF document to a series of graphic files, such as PNG or TIF, you are said to be "rasterizing" that PDF document.

In effect, with this option PDF2IMG converts the input PDF document into a series of graphic image files, and then saves each of these image files into a separate PDF output file, one PDF output document for each page in the PDF source document.

pdfregion

```
-pdfregion=[art | bleed | bounding | crop | media | trim]
-pdfregion=art
-pdfregion=bleed
-pdfregion=bounding
-pdfregion=media
-pdfregion=trim
-pdfregion=0,300,300,0
```

(Default: crop)

Use the pdfregion option to select a region of the input page or pages in a PDF document to rasterize. Elements not within the indicated area are ignored and do not appear in output. The values for pdfregion correspond to page boundary definitions as given in section 14.11.2, "Page Boundaries," in ISO 32000-1:2008, Document Management-Portable Document Format-Part 1: PDF 1.7, page 627.

This is summarized below.

Region	Coverage Area
art	Defines the logical extent of the content of the graphic as intended by the page creator. This is the smallest of margins. With “art” only the graphic itself is exported. No space is included around the image.
bleed	Defines the region where the contents of the page will be clipped. It may include an extra area surrounding the graphic to allow for the physical limitations of printing equipment. This value is the second widest margin around the graphics image after media.
bounding	Usually the smallest possible rectangle that can hold all of the content on the page (all the area within the declared bounding dimensions). It is possible for Bounding to include objects that fall outside the borders of the PDF page such as a particularly wide Bezier curve.
crop	(*Default*) Defines the region for clipping or cropping the graphic for display or print. Unlike the other settings for region crop has no default defined size or geometry. With the crop value it is possible to provide additional information to manually define the margins of the image that are selected and exported. If these values are not provided the crop value will match the media value.
media	Defines the boundaries of the actual page where the graphics image will be printed. In this case the media setting may include an extended area around the graphic on the printed page. This area can be used for printing marks on a proof copy for example. The media value provides the widest possible margins around the graphics image.
trim	Defines the intended dimensions of the finished graphic after trimming to fit the page. This will be smaller than the media and bleed settings but wider than the art setting.

It is also possible to create a box with a custom set of four coordinates, [left],[top],[right],[bottom] in PDF units. The same syntax applies for PDF2IMG and PDF2IMG .NET. In this example, the size and placement of the box is defined as zero units from the left side of the page, 300 units down from the top, 300 units in from the right, and zero units up from the bottom:

```
-pdfregion=0,300,300,0
```

A PDF unit here is a form of measurement used with PDF documents to define the placement of text of graphic on a page. There are 72 PDF units per inch.

pdf-split

```
-pdf-split
```

This option simply takes a PDF input document and splits it into a series of separate PDF documents, one PDF document for each page in the source file. The output is not converted into graphics images.

pixelcount

```
-pixelcount=[width x height]
-pixelcount=w:[width]
-pixelcount=h:[height]
-pixelcount=2550x3300
-pixelcount=660X300
-pixelcount=w:200
-pixelcount=h:300
```

Use the -pixelcount option to specify the exact dimensions of your output width and/or height in pixels. Both width and height values are optional, though if you provide both, you must give the width first, height second, separated by the "x" character. Do not include spaces.

This allows you to scale the image up or down as desired to a specific output size and dimension, and either maintain the original proportions or override them as you like: you can resize the original input without alteration or distort it horizontally or vertically as you prefer. If you don't use the -pixelcount option PDF2IMG will use the original size of the image when generating output.

If you do not want to alter the proportions or aspect ratio of the page or pages in your original input document, you should give only one -pixelcount dimension for output, either width or height, whichever one is more critical for correct output positioning or sizing. The other dimension will be scaled as necessary to maintain the original aspect ratio of the input PDF file.

Remember that the -resolution argument also affects output size by setting the Dots per Inch value, so be sure that your -pixelcount and -resolution arguments together will produce the output size that you want.

Note: The limit for JPEG output image size is 65535 x 65535 pixels. TIF output has been tested up to a band of 68898 x 34449 pixels in size.

Using pixelcount for RAW Output

When converting to RAW output, the process will create an output byte stream to generate the page image according to its original input height and width (the PDF page dimensions), and at the current resolution value (its default value, unless directed otherwise). However, the image will not contain any embedded information on its correct dimensions in pixels. Thus, you should specify the desired pixelcount dimensions yourself, in order to ensure that the output image size is what you expect.

profileCMYK/profileRGB/profileGray

```
-profileCMYK [file name or description of CMYK input color profile]
-profileRGB [file name or description of RGB input color profile]
-profileGray [file name or description of grayscale input color profile]
-profileRGB=AdobeRGB1998.icc
```

PDF2IMG offers three command line arguments to select an input color profile, one each for CMYK, RGB, or grayscale. These input color profile arguments are similar to colorprofile. For example, you would use the profileCMYK argument to select an input profile for CMYK and the colorprofile or the outputintent argument to select an output color profile. It is common practice to define both the input and output profiles when rasterizing a PDF document to a graphic file.

The output color profile is used for rasterizing the PDF document to a bitmap graphics file, or series of graphics files. The output profile can be added directly to the output file. For example, if you are converting a PDF document to a series of PNG files, one PNG file per page, you can define an output color profile and then add that output profile to the metadata for each of those PNG output files. This applies to PNG and TIFF files; you can't add color profile metadata to a BMP file.

When you select the input color profile, you are assigning that profile to objects and elements within the PDF document that do not already have native color profiles assigned to them. PDF2IMG copies your input color profile to a buffer, and the API calls it from there.

The elements and objects in a PDF document can be specified in a device-specific color space like DeviceCMYK, or they can be expressly assigned to an ICC color profile. If a color profile is already assigned to an element in a PDF document, that element is said to be *calibrated* to that color profile. PDF2IMG only applies a default color profile to an element in a PDF that is not already calibrated. For example, suppose a person creates a PDF document and embeds a photograph on one of the pages in that document. If that person selects an explicit ICC color profile to assign to that photograph, PDF2IMG will not change it. If the person who creates the PDF does not assign a profile to the image, however (the image is not already calibrated), PDF2IMG will assign a default input color profile to that image.

You can use profileCMYK, profileRGB, or profileGray, to assign your own input color profile to a document, and thus override the default color profile provided by PDF2IMG. But these three input color arguments will also not change the color profile assigned to an element or object that is already calibrated. If you don't use one of these three arguments to select a color profile, PDF2IMG will select a default instead.

Color Space	Default ICC input color profile provided by PDF2IMG
CMYK	Adobe Reader 9 CMYK
Gray	Gray Gamma 2.2
RGB	sRGB

The filename for the input profile argument refers to the ICC color profile file, usually ending with an ".icc" suffix, as in "HPO63000.icc." If the program is calling an ICC profile from another directory, provide the path name for the file as well. PDF2IMG will read the file you provide into a buffer and try to use it when rasterizing the PDF document. You can also enter the profile description, corresponding to the description field of the ICC profile you want to use, as in:

```
"HP OJ 6300-Premium Paper(tri-color+black)"
```

If you like you can use two or three of these color profile arguments in a single PDF2IMG command statement to assign separate profiles to individual objects or elements within a PDF document. You could also include a colorprofile or outputintent argument in the same statement to define the output profile to assign to the graphic file.

Suppose you have a PDF document with a single page that features grayscale text, a vector line drawing with a RGB color space, and a photograph (JPEG image) using CMYK. PDF2IMG will convert this PDF page to a single export graphic file, such as a TIF or PNG file. But you could write a PDF2IMG command statement that will assign a separate input color profile to each of these graphics, and you could also add a colorprofile or outputintent argument to define the output color profile. The command might look something like this:

```
pdf2img –profileCMYK=Probev1_ICCv4.icc –profileRGB=USWebCoatedSWOP.icc  –
colorprofile=AdobeRGB1998.icc TestFile.PDF PNG
```

In this example, PDF2IMG would convert the PDF document called TestFile.PDF to a single PNG file.

It will assign the color profile Probev1_IDCCv4.icc to the CMYK graphic (the JPEG photograph) in TestDocument.PDF, and the color profile USWebCoatedSWOP.icc to the RGB vector line drawing. For the text, PDF2IMG will use its own default Grayscale profile, because the –profileGray argument is not used in the statement. And it will assign AdobeRGB1998.icc to the PNG output file.

relaxParseSyntax

```
-relaxParseSyntax
```

This option can be used to correct a problem where PDF2IMG is not processing a PDF document properly. It can correct minor PDF syntax errors by relaxing a parsing restriction, allowing the software to continue to process the PDF document. The option sets the PDPrefSetAllowRelexedSyntax flag to True; by default, this flag is turned off.

removewhitespace

```
-removewhitespace
```

This command directs PDF2IMG to remove the white space margins around the content on a page in the PDF document. Most documents feature black text and color images on a white background. This option is similar to the Crop feature in Adobe Acrobat, in that you can use it to remove the white space surrounding an image or block of text on a page. With the removewhitespace option, however, PDF2IMG determines how much white space to remove for you.

This option does not accept a value.

resampler

```
-resampler=[auto | bicubic | none]
-resampler=bicubic
-resampler=none

(Default: auto)
```

If images are converted without resampling, it can in some cases cause unwanted artifacts or loss of detail in small-sized or low-resolution output images, such as thumbnails. Automatic resampling was introduced to PDF2IMG to enhance the quality of images when they are converted.

If you use the default value of automatic resampling, if any of the following conditions are true, the images will first be rasterized to 150 DPI.

After that a bicubic downsampling to the desired target values will be applied:

- -pixelcount:h is less than one half of the default input height
- -pixelcount:w is less than one half of the default input width
- -resolution is less than 150

Specifying bicubic will apply the resampler unconditionally. That is, every image will be resampled regardless of the pixel count and resolution. Specifying none will turn it off completely, so that the images will not be resampled at all.

resolution

```
-resolution=[12 to 2400]
-resolution=600
```
(Default: 300*)*

```
-resolution=[horizontal 12 to 2400 x vertical 12 to 2400]
-resolution=1200x600
```
(Default: 300x300*)*

Use the -resolution option to set the Dots per Inch value for the output file or files, from 12 to 2400. You can enter one or two values. If you only provide one value, it will be applied to both horizontal and vertical, as in 600 x 600. If you provide two values, the first will be applied horizontally and the second will be applied vertically. Do not include spaces between the numbers in the command line statement.

For best results, provide a value that serves as a multiple of the DPI resolution of the intended output device. Try to match the resolution of your image file to the device that will display it.

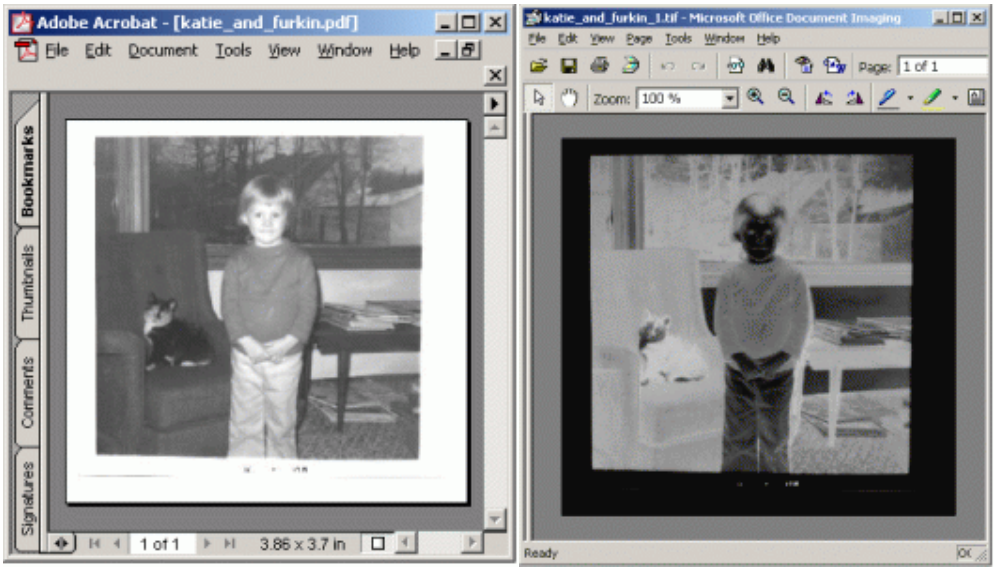
Note: If the specified value is less than 150 and resampling has not been disabled, images will be resampled for improved appearance during conversion. See “Resampler” for more details. Also see “PixelCount” for another way to control the specific output size of your image.

reverse

`-reverse`

(Grayscale only; No default)

If you are converting to grayscale output files, use the -reverse option to direct PDF2IMG to reverse the grayscale values to produce a "negative" image:



Note: This option is intended for generating reverse or negative images only. If you are trying to correct a problem of unwanted display reversals seen in documents produced by some third-party PDF products, use the “blackisone” option instead.

smoothing

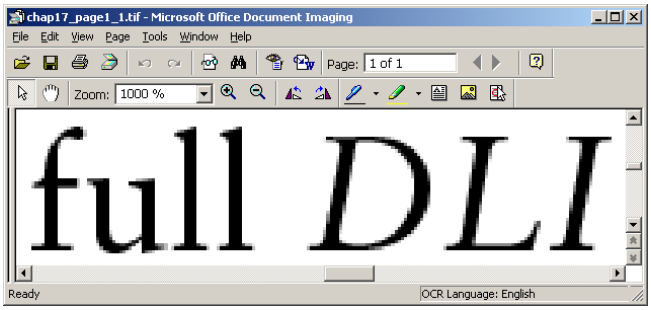
```
-smoothing=[none | text | line | image | all]
-smoothing=none
-smoothing=image
-smoothing=line,text
-smoothing=all
```

(Default: no smoothing)

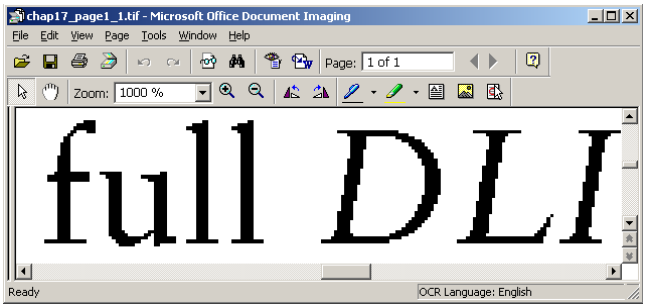
User-controlled anti-aliasing, or "smoothing," can be controlled individually for text, line art, images, or any combination as you like. Multiple selections should be separated by commas for PDF2IMG.

The smoothing option for PDF2IMG accepts "none," "text," "line," "image" or "all" values. You can combine "text," "line" or "image" with commas.

Text Smoothing (1000% Enlargement)



No Smoothing (1000% Enlargement)



Smoothing is most helpful when creating low-resolution outputs. But we don't recommend it if you want to create image files that you plan to print. It is also not recommended for black-and-white (1bpp) output files, to preserve sharpness at high magnifications.

Color Management

How printers and other hardware present colors will vary from one device to another. Each device independently defines the colors used. Even if two printers process a shade of green using the same CMYK color values—Cyan 31, Magenta 0, Yellow 80, Black 5—the appearance of the two printed documents still might not match.

The lack of a standard for managing color became a problem with the complex technology that started to appear in the mid-20th century. A wide variety of input devices were introduced by multiple manufacturers, including scanners, printers, digital cameras, and mobile devices. These devices needed to communicate with an equally wide variety of output devices, such as monitors, presses, laser, ink jet, and dot-matrix printers, and copy machines. This created a vast number of possible color conversions from one hardware device to another.

In response, color management was introduced, using color profiles. The International Color Consortium (ICC) developed a color specification in 1993 that works across all operating systems and software packages and applies regardless of the hardware involved. All color profiles are based on this ICC specification. A color profile is a table that specifies standard values for a range of colors, and that works as a translation matrix between devices. Any two devices involved in a transaction that requires content to be printed or displayed will share a color profile and convert their internal color values to match the standard provided in that profile.

A variety of color profiles have been defined, presented in the form of .icc files. Some of these profiles are specific to hardware devices, and define what a camera can detect, or a printer print, or a monitor display. Others are based on software and thus can be used across many kinds of devices. For example, USWebCoatedSWOP is a standard CMYK color profile, commonly used with Adobe Systems software products like PhotoShop and InDesign. The standard RGB color space, sRGB, was developed by Microsoft and Hewlett Packard to describe colors available on most monitors and other displays. This color space is also commonly used for web graphics. And the Adobe RGB color profile (AdobeRGB1998.icc) was designed by Adobe Systems to hold all of the colors that are likely to be available on any color CMYK printer. It is considerably larger than standard RGB.

A single PDF document can support a wide variety of elements using different color models. Often a PDF file is produced and saved with elements in the DeviceRGB, DeviceCMYK and DeviceGray spaces that have an associated ICC profile. An element in a PDF that has an associated profile is considered calibrated. Elements that do not have embedded profiles are considered un-calibrated. PDF processing software will often assign default profiles (referred to as working spaces) to un-calibrated elements. Graphics files, however, such as PNG, TIF, or JPEG, can only hold a single color profile. When PDF2IMG rasterizes a page from a PDF document to create a graphic file, it will assign default profiles for un-calibrated elements in the PDF, or you can specify the input and output color profiles you want to use from a stream or file. When PDF2IMG initializes, it identifies the color profiles present in the subject PDF document, and that are available on the host machine.

Conversions with ICC Color Profiles

PDF2IMG will honor calibrated colorspaces in PDF files, if output color management is in effect. The product will also write target ICC profiles to the TIF, JPEG, PNG, or BMP output files.

Color management can be suppressed using either the Color Management Module nocmm command line call or the pdf2img_set_colormanagement API call.

For non-calibrated spaces, PDF2IMG will use the defaults provided by the Adobe PDF Library for conversions, depending on the colorspace.

Default Conversion Color Profiles

Colorspace	ICC Color Profile (input)	ICC Color Profile (output)
RGB	Adobe 1998 RGB	sRGB
CMYK	Adobe Reader 9 CMYK	Adobe Reader 9 or later CMYK, based on Simplified US web coated SWOP v2
Gray	Gamma 2.2	Gamma 2.2
L*a*b		CIE 1976 (L*a*b*) color specification with a D50 white point

Conversions with Missing Resources

When creating a PDF document, the best practice is to embed all the fonts that document will need to use in the document itself. That way, the software used to open that PDF document (such as Adobe Reader or Acrobat) does not need to try to find fonts that it needs from the local workstation or laptop or provide substitutes.

Likewise, before PDF2IMG starts the process to convert a PDF document to a graphic file or series of graphic files, the product will look for the needed fonts within the PDF document itself.

If the fonts are not embedded in the document PDF2IMG will try to make use of font resources found on the system. PDF2IMG can use environment variables to determine where to find those font resources.

Or the software can provide its own fonts. PDF2IMG offers internal font and CMap resources that are shipped with the product. These font resource files are copied to the installation directory for the PDF2IMG software (where the executable is stored) when you install the product.

Font and CMap Location Searches

Providing Custom Locations for Font Files

By default, PDF2IMG looks for fonts installed and used by Adobe products, such as Adobe Acrobat, and will use them if they are present. However, you can provide your own list of directories to search for font files (up to 16 local or server file folders) using the optional fontlist command line parameter. The fontlist option directs the software to look in the directories you provide instead of the default Adobe sites.

After searching the font locations you provide it then searches the font folders provided upon installation of the product, in the following locations, relative to the present installation directory:

- Resources
- Resources/CMap
- Resources/Font

PDF2IMG searches first in the locations you provide with the fontlist command line option. Any locations for fonts that you provide using the fontlist argument will supersede another of the same name that may appear in one of the default locations.

It is important to understand the PDF2IMG can convert a PDF document that references fonts that are not in fact embedded in that document. But if the software cannot find those fonts on the local machine, PDF2IMG will substitute its own. The product will not display an error message, but this might lead to results you might not want, especially if the document is calling fonts with obscure or decorative typefaces or special symbols. See “Conversions with Missing Resources” on page 31.

Multi-Page Processing

By default, PDF2IMG converts the pages in a PDF file into multiple single-page image files. The system assigns a name to each image file with an underscore character (_) added between the file name prefix and the sequence counter number. For example, the five pages in a PDF input file called Convert.PDF would be exported to form five bitmap (BMP) graphic files:

- convert_1.BMP
- convert_2.BMP
- convert_3.BMP
- convert_4.BMP
- convert_5.BMP

If you are working with a long PDF document, the software will by default pad the sequence number with leading zeroes as needed, to keep the export graphic files in order by name. So if Convert.PDF has 128 pages, the first few BMP export files will be named:

- convert_001.BMP
- convert_002.BMP
- convert_003.BMP

Followed, later, by:

- convert_043.BMP
- convert_044.BMP
- convert_045.BMP

And finally by:

- convert_126.BMP
- convert_127.BMP
- convert_128.BMP

Remember:

1. You can select PDF as your export format for a PDF document, rather than a graphic file format like PNG or BMP. So if you have a 15-page PDF document and want to use PDF2IMG to export it to PDF, you can set up the software to provide a series of 15 PDF documents as the export result. The software will use the page-split feature. To this end you will need to select the pdf2img_set_pdf_output_type API call, or the command line statement pdf-rasterize-split or pdf-split.
2. You can use the optional **firstonly** command line argument to direct PDF2IMG to convert only the first page of the input PDF file rather than every page in the document. So you can use this argument to test a conversion before using the product to process a series of very long PDF documents.

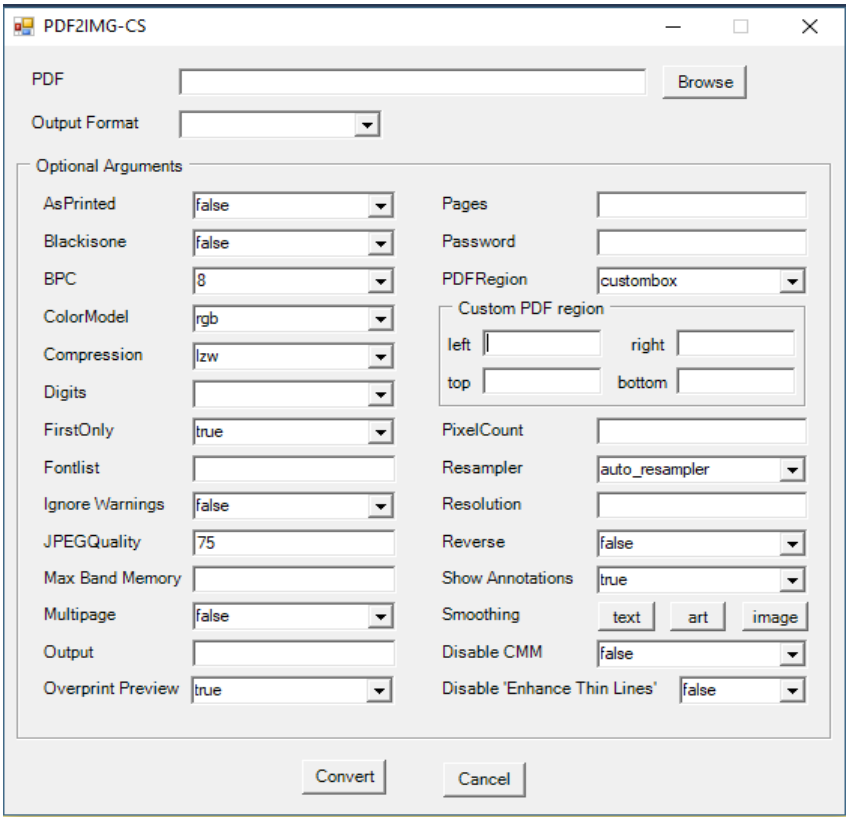
3. Or you can use the optional multipage command line argument to convert long, multipage PDF document into a single multipage TIFF document.
4. Use the optional digits command line argument to specify the number of digits for numbering the output files and suppress the underscore character (_). For example, if you enter "-digits=3" as part of a command, the export files generated from your input PDF document will be named FILE001.JPG, FILE002.JPG, and so on.
5. Finally, use the optional output command line argument to specify an alternate output file name prefix. The value given here will be used for the output file name, with a sequence number and an appropriate extension appended to indicate the output file type, such as sample_1.gif and sample_1.jpg. For a PDF input document with more than one page, the system will create sequential, separate output files for each page of the input file, with a sequence number appended to each, such as sample_1.gif, sample_2.gif, sample_3.gif, and so on.

Note: An alternate file folder location can be specified for the output file(s) to be created, but that folder must already exist; PDF2IMG will not create it.

Working with the legacy COM Interface *(Windows 32 only)*

You will see a C# application subfolder under the PDF2IMG directory. This folder will include the PDF2IMG COM modules, source code, and sample program files. You may alter the sample program source code to build your own applications.

When you run the PDF2IMG COM sample application it presents a Graphic User Interface that looks like this:



Working with the .NET Interface *(Windows 64)*

The .NET interface is also provided as part of the Windows 64-bit version of the product.

To use the PDF2IMG interface for .NET follow these steps:

1. Create an instance of the PDF2IMG class.
2. Load your input PDF or XPS document, to be converted to an image file or series of image files.
3. Select the image conversion options you need.
4. Convert each page of the input document to an image file.
5. Dispose of the object.

The public methods provided with the PDF2IMG .NET interface include:

- CheckForMissingAppearances()
- ConvertPageToImage()
- ConvertAllPagesToTIFFImage()
- GetPageBoxWithWhiteSpaceRemoved
- LoadInput()
- SetImageConversionOptions()

For more detail see the description of the API calls for the .NET Interface.

Troubleshooting

Problems with Opening a PDF Document

Opening a PDF file for conversion is not the same as opening it for viewing. In some cases, efforts to secure a PDF document may prevent PDF2IMG from completing the conversion process.

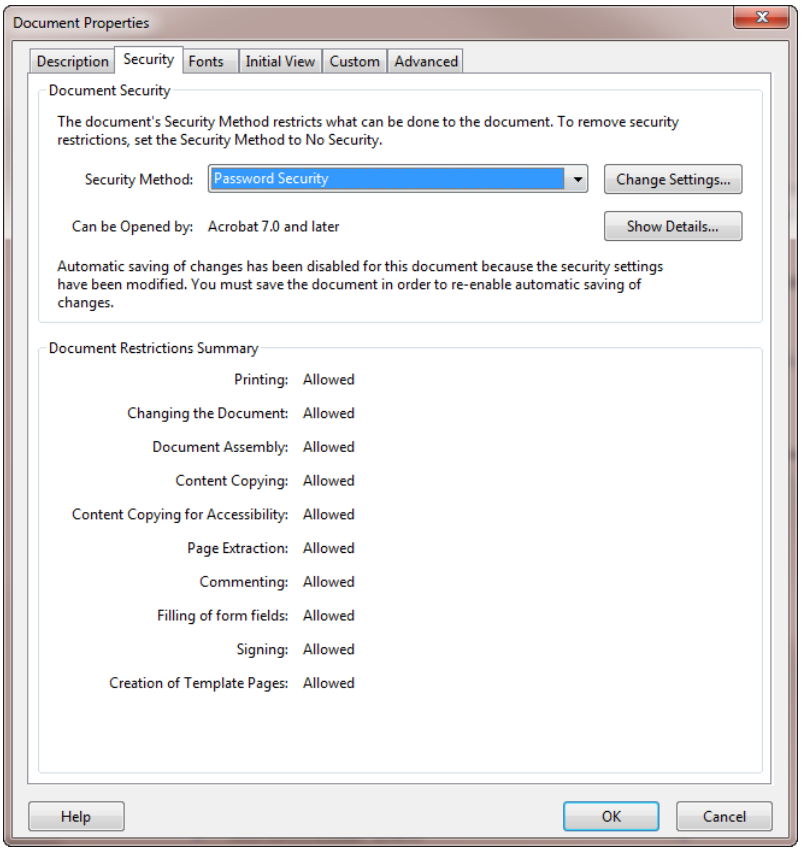
If a PDF document is password protected, you will need to provide that password before PDF2IMG can work. To that end you can use the optional password command line argument if you are using a console application.

Or you can use one of these API calls:

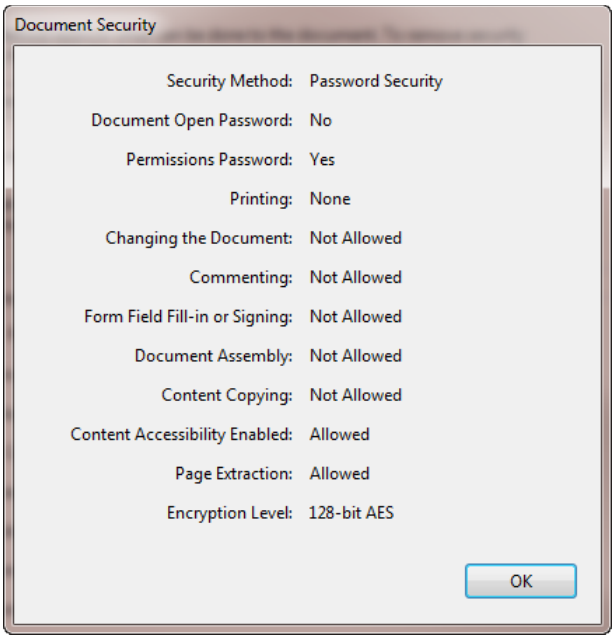
- pdf2img_new_conversion_with_password
- pdf2img_new_memconversion_with_password

Also, the PDF document must be set to allow Content Copying and Page Extraction before PDF2IMG can process it.

Open the file in Adobe Acrobat and click File/Properties, and then the Security Tab. Make sure that Content Copying and Page Extraction are allowed for this document.



If the Security Method for the file is set to “Password Security,” as shown above, click Show Details.



The “Document Open Password” value should be set to No. This means that the file does not have a Document Open password.

If this value is set to Yes, it means that to open the PDF document you need to enter this Document Open Password, and you will need to provide this password to PDF2IMG.

PDF2IMG may be able to process a PDF document with a Permissions Password, however, depending on what Security settings are in force. You can open a PDF document without entering the Permissions Password, which is used to lock the PDF document so that the security settings for that document cannot be changed. These settings might include the ability to print the PDF or to insert or delete pages.

If PDF2IMG cannot process a PDF document because of a security or permission problem, you will see an error message:

```
return code = 13, The document security settings do not permit this operation
```

You will need to investigate the document security or permission settings on the PDF document before continuing.

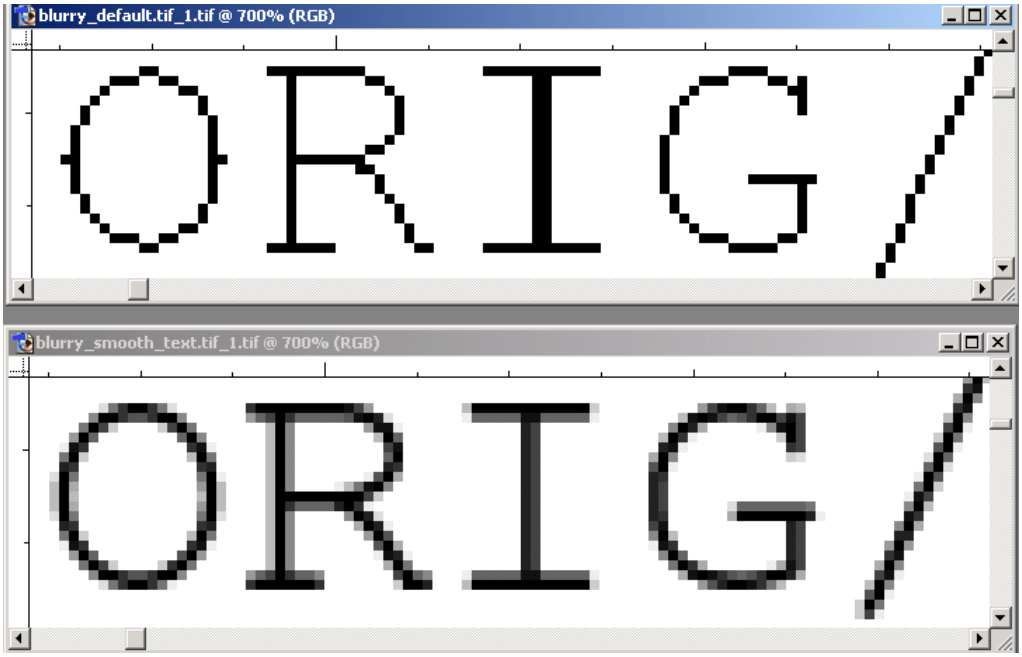
Problems with High-Resolution Conversions

Large PDF files processed at high resolution settings may cause PDF2IMG to fail; you might see a message indicating that a raster port could not be created. This typically means that there is not enough system memory available to generate raster output for the input PDF document. Rasterizing PDF documents at high resolution requires a lot of free memory.

Blurry Text Appears in Output File

When converting a PDF document with a small typeface, such as 8-point Courier, a blocky typeface with a 1-point stroke width, to TIFF output, the resulting output may appear grainy and pixelated. Some characters may develop visible gaps despite using the -smoothing command line argument.

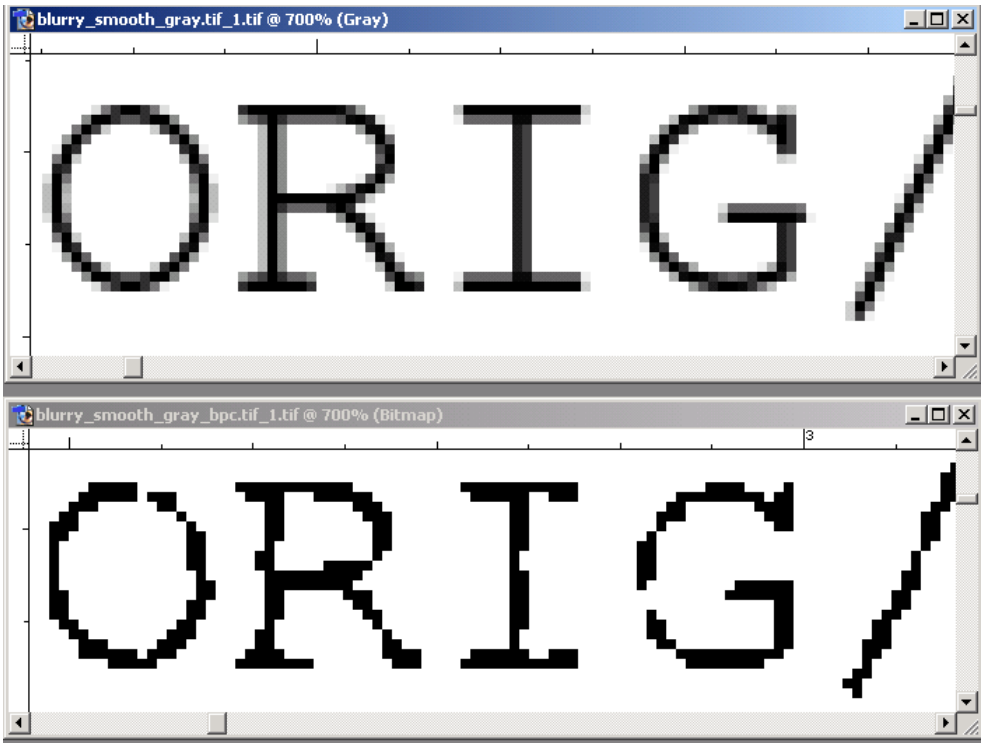
When small text is rendered as a TIFF image without any smoothing attempts, the very narrow stroke thicknesses for the characters are dramatically affected by whether they are squarely aligned over the image pixels or not. Some legs of the glyphs will be rendered with a single line of pixels, while others will be drawn with two:



Compare the vertical strokes of the “R” and “I” characters in the upper window above. These characters have no smoothing. The “R” is centered over the pixel grid and uses one vertical column, while the “I” is straddling the grid and ends up using two columns. When smoothing is applied, as seen in the lower window, it limits this effect. If a typeface that uses bold characters or characters that are shaped differently, especially a typeface without long, straight horizontal or vertical strokes, the problem might not be visible at all, even without any smoothing.

However, if you add too many arguments to your command line statement you can make matters worse. In this case, shades of gray will be needed for effective smoothing.

While you could add a Bits per Color channel (-bpc=1) setting to the conversion, we don’t recommend it.



In the images above, we see that when the Bits per Color Channel argument is set equal to 1 (-bpc=1) we lose all the gray values from the smoothing operation. The glyphs end up looking grainy and spotty as a result.

So, the best configuration for each conversion depends on the details of the document you want to convert. If you want to convert a document with a fine-grained text that needs a smoothing to preserve the small type, do not alter it using the “bpc” argument. You need to preserve the gray values necessary for rendering the characters.

Appendix: API Calls, C Language Interface

pdf2img_check_for_missing_appearances

(ImageConversion IC, int firstPage, Int lastPage)

Return Value: int

Description	This call checks pages within the specified range for annotations or form fields that cannot be rendered. It returns a value greater than 0 if the page range contains non-renderable content or 0 if the page range does not contain non-renderable content.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **int firstPage**: sequence number of first input file page to be inspected counting from 1 **int lastPage**: sequence number of last input file page to be inspected counting from 1
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_convert_page

(ImageConversion iC, unsigned int pageNum, const char *outputPath)

Return Value: int

Description	This call converts the specified page of the PDF file. It writes into the file specified by outputPath. Alternatively you can suppress writing an output file by passing a NULL in place of an output file argument and then use a pdf2img_get_pagemem call to load the converted graphic into memory.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int pageNum**: sequence number of input file page to be converted counting from 1

	const char *outputPath: name of file to receive converted output or NULL
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_get_pagemem pdf2img_get_pagememsize pdf2img_release_pagemem
Availability	All platforms

Technical Notes

1. While a NULL argument will suppress writing an output file, note that a work file will be temporarily created during the conversion process.
2. If the conversion is performed "in memory," then the ImageConversion will hold the memory for the conversion until the next call to pdf2img_convert_page, or until pdf2img_release_pagemem is called.

pdf2img_destroy_conversion

(ImageConversion IC)

Return Value: int

Description	This call frees the resources used by this image conversion. After this call has been made the ImageConversion is no longer valid and must not be used.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_new_conversion pdf2img_new_memconversion
Availability	All platforms

pdf2img_end_multipage

(ImageConversion IC)

Return Value: int

Description	This call ends a multipage image span.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_multipage pdf2img_start_multipage
Availability	All platforms

pdf2img_error_string

(ImageConversion IC)

Return Value: const char *

Description	This call returns an English-language string representing the last error encountered during PDF processing. Do not release this string.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	const char *
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_get_extended_error pdf2img_last_error
Availability	All platforms

pdf2img_get_extended_error

(ImageConversion IC)

Return Value: const char *

Description	This call returns a text string indicating the last error that occurred in a sub-component of the application. Typically, this is an error reported by the Adobe PDF Library.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	const char *
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_error_string pdf2img_last_error
Availability	All platforms

pdf2img_get_num_pages

(ImageConversion IC)

Return Value: int

Description	This call returns the number of pages in the PDF file to be used for the image conversion.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	int: Number of pages to be converted. Returns -1 in case of error.
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_get_pagemem

(ImageConversion IC, void *memBuf, unsigned int bufCapacity)

Return Value: Int

Description	After pdf2img_convert_page has converted a graphic into memory (by specifying NULL in place of its outputPath argument) this pdf2img_get_pagemem call will copy those results into memBuf. Note that memBuf is owned by the caller and must have been allocated to hold at least bufCapacity bytes. The pdf2img_get_pagememsize call can return the necessary size for this allocation.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **void *memBuf**: buffer allocated to receive the image from memory **unsigned int bufCapacity**: size of the allocated buffer
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_convert_page pdf2img_get_pagememsize pdf2img_release_pagemem
Availability	All platforms

pdf2img_get_pagememsize

(ImageConversion IC)

Return Value: int

Description	After pdf2img_convert_page has converted a graphic into memory (by specifying NULL in place of its outputPath argument) this call will return the number of bytes that will be required to hold the memory output. Results of this call will be needed to allocate the correct amount of memory buffer for a subsequent call to pdf2img_release_pagemem.

Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	**int**: number of bytes required to hold the memory representing the graphic created by pdf2img_convert_page
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_convert_page pdf2img_get_pagemem pdf2img_release_pagemem
Availability	All platforms

pdf2img_get_profiledata

(ImageConversion IC)

Return Value: char *

Description	If an input image in memory contains a color profile this call will return a pointer to a string containing the ICC Profile data (or NULL if a profile is not present.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	**char ***: pointer to a string containing the ICC Profile data
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_get_profilesize
Availability	All platforms

pdf2img_get_profilesize

(ImageConversion IC)

Return Value: size_t

Description	If an input image in memory contains a color profile this call will return its size (or 0 if a profile is not present).
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	**size_t**: the size of the ICC Profile data
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_get_profiledata
Availability	All platforms

pdf2img_get_region_remove_white_margins

(ImageConversion iC, unsigned int pageNum, double* left, double* top, double* right, double* bottom)

Return Value: int

Description	Returns coordinates of a tight-fitting bounding box encompassing all text, graphics, and images on the page. This effectively removes the white margins around the page (assuming black foreground on white background).
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int pageNum**: sequence number of input file page to be converted counting from 1 **double *left**: coordinate for left margin of bounding box **double *top**: coordinate for top margin of bounding box **double *right**: coordinate for right margin of bounding box **double * bottom**: coordinate for bottom margin of bounding box
Return Value	int
Exceptions	

Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_init

(const char *fontDirList[], unsigned int listLen)

Return Value: int

Description	This calling argument initializes PDF2IMG before use. If used in a multi-threaded application each calling thread must do its own initialization before use.
Parameters	**const char *fontDirList[]**: a list of null-terminated C strings containing directories in which PDF2IMG should search for fonts and font resources to use when rasterizing documents which have non-embedded fonts **unsigned int listLen**: number of strings in this list
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_init_ex pdf2img_term
Availability	All platforms

pdf2img_init_ex

(PDF2IMGInitParams *pinitParams)

Return Value: int

Description	This calling argument initializes *PDF2IMG* before use in the same manner as pdf2img_init. If used in a multi-threaded application each calling thread must do its own initialization before use. This argument is an alternative to the standard pdf2img_init call which scans default system font resource locations as well as its current directory at startup time. That font scanning time can be quite significant in some cases and this call offers the option of fine-tuning performance by

	suppressing the default scans and by providing a specific list of locations to be scanned instead. When the ignoreSysFonts and ignoreCurrentDirectory elements of its PDF2IMGInitParams argument structure are set to true (they default to false) this call will search only the indicated font resource locations passed within the fontDirList element and no other locations.
Parameters	**PDF2IMGInitParams *pInitParams**: structure for passing a list of font resource locations to be searched at startup along with flag settings to prevent searching the current directory or the system fonts areas.
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_init pdf2img_term
Availability	All platforms

pdf2img_last_error

(ImageConversion IC)

Return Value: int

Description	This call returns the last error number.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_get_extended_error
Availability	All platforms

pdf2img_new_conversion

(const char *inPDFPath)

Return Value: imageConversion

Description	This call creates a PDF-to-Image conversion. Each is local to the thread which makes this function call and should not be shared across threads.
Parameters	**const char *inPDFPath**: path to location of input image file
Return Value	ImageConversion
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_destroy_conversion pdf2img_new_conversion_with_password pdf2img_new_memconversion pdf2img_new_memconversion_with_password
Availability	All platforms

Technical Notes

1. This call maintains an open file handle for the inPDFPath argument.
2. Of the two calls pdf2img_new_conversion and pdf2img_new_memconversion, one or the other must be called to create an image conversion, but not both.

pdf2img_new_conversion_with_password

(const char *inPDFPath, const char *password)

Return Value: ImageConversion

Description	This call creates a PDF-to-Image conversion using the supplied password. Each is local to the thread which makes this function call and should not be shared across threads.
Parameters	**const char *inPDFPath**: path to location of input image file **const char *password**: password string for opening the PDF document
Return Value	ImageConversion
Exceptions	

Header	pdf2imglib.h
Related Methods	pdf2img_destroy_conversion pdf2img_new_conversion pdf2img_new_memconversion pdf2img_new_memconversion_with_password
Availability	All platforms

Technical Notes

1. This call maintains an open file handle for the inPDFPath argument.
2. A document with a User password but no other security restrictions can be processed by providing the User password if the document has any security restrictions on it beyond simply a User password to restrict viewing, you will need to specify its Owner password instead. Opening with an Owner password will override the document's security restrictions and allow the PDF2IMG conversion to proceed.
3. It is safe to call pdf2img_new_conversion_with_password with either a NULL pointer or a pointer to zero length string in the password In such cases, this call will behave exactly the same as its non-password counterpart, pdf2img_new_conversion.
4. Of the two calls, pdf2img_new_conversion_with_password and pdf2img_new_memconversion_with_password, one or the other must be called to create an image conversion, but not both.
5. The password argument is copied to an internal buffer. The client's copy may be released after this.
6. The password must grant the following permissions on the PDF document:

If using	**Permission Required**
Any output format	Open and Copy
EPS output format	High Quality printing
Color management	Modify

pdf2img_new_memconversion

(const void *inPDFMem, unsigned int numBytes)

Return Value: ImageConversion

Description	This call creates a PDF-to-Image conversion from a buffer of bytes. Each is local to the thread which makes this function call and should not be shared across threads.
Parameters	**const void *inPDFMem**: pointer to location of input image in memory **unsigned int numBytes**: number of bytes in this buffer
Return Value	ImageConversion
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_destroy_conversion pdf2img_new_conversion pdf2img_new_conversion_with_password pdf2img_new_memconversion_with_password
Availability	All platforms

Technical Notes

1. The memory in inPDFMem must be available until pdf2img_destroy_conversion is called for this conversion; it is not copied by PDF2IMG.
2. Of the two calls pdf2img_new_conversion and pdf2img_new_memconversion, one or the other must be called to create an image conversion, but not both.

pdf2img_new_memconversion_with_password

(const void *InPDFMem, unsigned int numBytes, const char *password)

Return Value: ImageConversion

Description	This call creates a PDF-to-Image conversion from a buffer of bytes using the supplied password. Each is local to the thread which makes this function call and should not be shared across threads.
Parameters	**const void *inPDFMem**: pointer to location of input image in memory **unsigned int numBytes**: number of bytes in this buffer

	const char *password: password string for opening the PDF document
Return Value	ImageConversion
Exceptions	pdf2img_destroy_conversion
Header	pdf2imglib.h
Related Methods	pdf2img_new_conversion pdf2img_new_conversion_with_password pdf2img_new_memconversion
Availability	All platforms

Technical Notes

1. The memory in inPDFMem must be available until pdf2img_destroy_conversion is called for this conversion; it is not copied by PDF2IMG.
2. A document with a User password but no other security restrictions can be processed by providing the User password if the document has any security restrictions on it beyond simply a User password to restrict viewing, you will need to specify its Owner password instead. Opening with an Owner password will override the document's security restrictions and allow PDF2IMG conversion to proceed.
3. It is safe to call pdf2img_new_memconversion_with_password with either a NULL pointer or a pointer to zero length string in the password In such cases, this call will behave exactly the same as its non-password counterpart, pdf2img_new_memconversion.
4. Of the two calls pdf2img_new_conversion_with_password and pdf2img_new_memconversion_with_password, one or the other must be called to create an image conversion, but not both.
5. The password argument is copied to an internal buffer. The client's copy may be released after this call.

If using	**Permission Required**
Any output format	Open and Copy
EPS output format	High Quality printing
Color management	Modify

pdf2img_release_pagemem

(ImageConversion IC)

Return Value: int

Description	This call releases the memory held by the ImageConversion for a graphic converted "into memory" by pdf2img_convert_page. This call is required only if pdf2img_convert_page was directed to return its output in memory instead of to an output file.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_convert_page pdf2img_get_pagemem pdf2img_get_pagememsize
Availability	All platforms

pdf2img_set_blackisone

(ImageConversion IC, unsigned short int blackisone)

Return Value: int

Description	Normal processing to TIFF output sets the photometric interpretation values as black=0; white=1. If calling pdf2img_set_blackisone with a non-zero value for the blackisone argument TIFF photometric interpretation will be transposed such that black=1 and white=0. This function is for 1-bit grayscale TIFF output only.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned short int blackisone**: 0: black = 0/white = 1 or 1 (or any non-zero) black = 1/white = 0 (*Default 0*)
Return Value	Int

Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_bpc

(ImageConversion IC, unsigned int bpc)

Return Value: int

Description	This call sets the Bits Per Channel (bpc) value for the output color space: either 1 (such as black and white) or 8.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int bpc**: 1 or 8 (Default 8)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_color_profile_from_buffer

(ImageConversion IC, void *data, size_t dataSize)

Return Value: int

Description	This call defines the color profile for use with PDF2IMG by selecting the profile in a buffer; the contents of the .icc file are stored in this buffer first.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **void *data**: pointer to the location of the specified color profile in memory **size_t dataSize**: the number of bytes to be read at the pointer location
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_color_profile_from_description pdf2img_set_color_profile_from_output_intent pdf2img_set_render_intent
Availability	All platforms

Technical Notes

1. Learn more about color management.
2. Any member of this group of functions can be called repeatedly:
   - pdf2img_set_color_profile_from_buffer
   - pdf2img_set_color_profile_from_description
   - pdf2img_set_color_profile_from_output_intent
3. A call succeeds if the specified profile exists, overriding a previous call if necessary. Thus, if several output profiles are specified, pdf2imglib uses the last one. Similarly, if several (RGB) input profiles are specified, pdf2imglib uses the last (RGB) input profile.
4. These functions are for setting the desired output color For specifying an input profile, see pdf2img_set_input_color_profile_from_description or pdf2img_set_input_color_profile_from_output_intent.

pdf2img_set_color_profile_from_description

(ImageConversion IC, const char *description)

Return Value: int

Description	This call defines the color profile for use with PDF2IMG by querying the system for the profile description.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **const char *description**: text string of profile description
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_color_profile_from_buffer pdf2img_set_color_profile_from_output_intent pdf2img_set_render_intent
Availability	All platforms

Technical Notes

1. Learn more about color management.
2. Any member of this group of functions maybe be called repeatedly:
   - pdf2img_set_color_profile_from_buffer
   - pdf2img_set_color_profile_from_description
   - pdf2img_set_color_profile_from_output_intent
3. A call succeeds if the specified profile exists, overriding a previous call if necessary. Thus, if several output profiles are specified, pdf2imglib uses the last one. Similarly, if several (RGB) input profiles are specified, pdf2imglib uses the last (RGB) input profile.
4. These functions are for setting the desired output color For specifying an input profile, see pdf2img_set_input_color_profile_from_buffer, pdf2img_set_color_profile_from_description, or pdf2img_set_color_profile_from_output_intent.

pdf2img_set_color_profile_from_output_intent

(ImageConversion IC, const char *description)

Return Value: int

Description	This call defines the color profile for use with PDF2IMG by pulling it from the output intent stored in the PDF document. An output intent is a dictionary array in the PDF document OutputIntents array. For more detail see the ISO 32000-1:2008 Document Management-Portable Document Format section 14.11.5 “Output Intents” on page 633.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **const char *description**: text string of profile description
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_color_profile_from_buffer pdf2img_set_color_profile_from_description pdf2img_set_render_intent pdf2img_set_input_color_profile_from_buffer
Availability	All platforms

Technical Notes

1. Learn more about color management.
2. Any member of this group of functions may be called repeatedly.
   - pdf2img_set_color_profile_from_buffer
   - pdf2img_set_color_profile_from_description
   - pdf2img_set_color_profile_from_output_intent
3. A call succeeds if the specified profile exists, overriding a previous call if necessary. Thus, if several output profiles are specified, pdf2imglib uses the last one. Similarly, if several (RGB) input profiles are specified, pdf2imglib uses the last (RGB) input profile.
4. These functions are for setting the desired output color For specifying an input profile, see pdf2img_set_color_profile_from_buffer, pdf2img_set_color_profile_from_description or pdf2img_set_input_color_profile_from_output_intent.
5. An output intent is a dictionary in the document’s OutputIntents array, as described in the PDF Reference, ISO 32000-1:2008, Document Management-Portable Document Format-Part 1: PDF

1.7, section 14.11.5, page 633. To use an output intent's color profile, pdf2imglib matches the description supplied by the client with its dictionary values as follows:

- use the first embedded profile with the specified OutputCondition value
- use the first embedded profile with the specified OutputConditionIdentifier value
- use the first embedded profile with the specified S (subtype) value

pdf2img_set_colormanagement

(ImageConversion IC, unsigned short int managecolors)

Return Value: int

Description	This call sets the Color Management Module (CMM) processing preference flag to determine whether color profile information should be embedded in the output images. When the managecolors flag is set to false PDF2IMG will draw the output to a device color. It will also embed no profile in the image written and presume no default color model for input device colors. When the managecolors flag is set to true (the default) PDF2IMG assumes that Device colors on input are calibrated as shown in the Technical Notes table below. The software also assumes a corresponding output profile to match. When generating TIF/JPG/PNG/BMP output PDF2IMG will embed the corresponding profile in the output image.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion unsigned short int managecolors: true or false *(Default true)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

For non-calibrated spaces, PDF2IMG will use the built-in defaults of its underlying Adobe PDF Library for conversions, depending on the colorspace.

Default Conversion Color Profiles

Colorspace	ICC Color Profile (input)	ICC Color Profile (output)
RGB	Adobe 1998 RGB	sRGB
CMYK	Adobe Reader 9 CMYK	Adobe Reader 9 CMYK (Simplified US web coated SWOP V2)
Gray	Gamma 2.2	Gamma 2.2

pdf2img_set_colorspace

(ImageConversion IC, ColorSpaceCode csCode)

Return Value: int

Description	This call sets the output colorspace. Not all combinations of colorspace with output format are valid; see Technical Notes below for more details.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion ColorSpaceCode csCode: Output colorspace value: CMYKcolor GRAYcolor LABcolor RGBcolor RGBAcolor (Default RGBcolor)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

1. When generating EPS output, calls to pdf2img_set_colorspace will have no effect, as the PDF file is not rasterized during processing.
2. Not all colorspaces are valid for all output See the table below:

Available Output Types per Colorspace

Colorspace	Output Type
CMYKcolor	JPEG or TIFF
GRAYcolor	BMP/GIF/JPEG/PNG/TIFF
LABcolor	TIFF
RGBcolor	BMP/GIF/JPEG/PNG/TIFF
RGBAcolor	PNG or TIFF

pdf2img_set_compression

(ImageConversion IC, CompressionCode cmCode)

Return Value: Int

Description	The call sets the output compression. This function is for TIFF output only. See Technical Notes below.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **CompressionCode cmCode**: Output compression value: NOcompression LZWcompression G3compression G4compression JPGcompression (Default LZWcompression)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

1. Compression codes are currently examined only when processing TIFF or JPEG graphics that use DCT compression, PNG graphics that use Flate compression, and GIF graphics that use LZW compression. BMP graphics are currently not compressed.
2. G3compression and G4compression values are only valid for B/W (1 channel, 1 bit) TIFF images.

pdf2img_set_enhance_thin_lines

(ImageConversion IC, in newVal)

Return Value: Int

Description	This call will toggle the Enhance Thin Lines setting (as used in Adobe Reader or Acrobat) on or off in the generated image. If set to true (the default) pdf2imglib will generate thin line renderings as they would appear when generated by Reader or Acrobat.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **int newVal**: Set enhance_thin_lines flag: 0: Do not set 1 (or any non-zero positive): Set enhance_thin_lines flag *(Default 1)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

This reflects a default rendering setting for enhance_thin_lines. It is intended to match the default behavior of Adobe Reader and Adobe Acrobat in enhancing thin document lines when rendering. This call allows you to turn off that behavior if you like.

pdf2img_set_horiz_res

(ImageConversion IC, unsigned Int hRes)

Return Value: int

Description	This call sets the horizontal resolution of the output expressed in dots/inch.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int hRes**: horizontal output resolution in dots per Valid range is 12 to 2400. *(Default 300)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_vert_res
Availability	All platforms

pdf2img_set_input_color_profile_from_buffer

(ImageConversion IC, void *data, size_t dataSize)

Return Value: Int

Description	This call defines the input color profile for use with PDF2IMG by selecting the profile in a buffer. The contents of the .icc file are stored in this buffer first.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **void *data**: pointer to the location of the specified color profile in memory **size_t dataSize**: the number of bytes to be read at the pointer location
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_input_color_profile_from_description pdf2img_set_input_color_profile_from_output_intent

	pdf2img_set_render_intent pdf2img_set_color_profile_from_buffer pdf2img_set_color_profile_from_description
Availability	All platforms

Technical Notes

1. Learn more about color management.
2. Any member of this group of functions may be called repeatedly:
   - pdf2img_set_input_color_profile_from_buffer
   - pdf2img_set_input_color_profile_from_description
   - pdf2img_set_input_color_profile_from_output_intent
3. A call succeeds if the specified (RGB) input profile exists, overriding a previous call if necessary. Thus, if several input profiles are specified, pdf2imglib uses the last one. Similarly, if several output profiles are specified, pdf2imglib uses the last output profile.
4. These functions are for setting the input (RGB) color. For specifying an output profile, see:
   - pdf2img_set_color_profile_from_buffer
   - pdf2img_set_color_profile_from_description
   - pdf2img_set_color_profile_from_output_intent

pdf2img_set_input_color_profile_from_description

(ImageConversion IC, const char *description)

Return Value: int

Description	This call defines the color profile for use with PDF2IMG by querying the system for the profile description.
Parameters	ImageConversion iC: the data structure containing the specifications for the active current conversion **const char *description**: text string of profile description
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_input_color_profile_from_buffer pdf2img_set_input_color_profile_from_output_intent pdf2img_set_render_intent pdf2img_set_color_profile_from_buffer pdf2img_set_color_profile_from_description

Availability	All platforms

Technical Notes

1. Learn more about color management.
2. Any member of this group of functions can be called repeatedly:
   - pdf2img_set_input_color_profile_from_buffer
   - pdf2img_set_input_color_profile_from_description
   - pdf2img_set_input_color_profile_from_output_intent
3. A call succeeds if the specified (RGB) input profile exists, overriding a previous call if necessary. Thus, if several input profiles are specified, pdf2imglib uses the last one. Similarly, if several output profiles are specified, pdf2imglib uses the last output profile.
4. These functions are for setting the input (RGB) color For specifying an output profile, see pdf2img_set_input_color_profile_from_buffer, pdf2img_set_color_profile_from_description or pdf2img_set_input_color_profile_from_output_intent.

pdf2img_set_input_color_profile_from_output_intent

(ImageConversion IC, const char *description)

Return Value: int

Description	This call defines the input color profile for use with PDF2IMG by pulling it from the output intent stored in the PDF document. An output intent is a dictionary array in the PDF document OutputIntents array. For more detail see the ISO 32000-1:2008 Document Management-Portable Document Format section 14.11.5 “Output Intents” on page 633.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **const char *description**: text string of profile description
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_input_color_profile_from_buffer pdf2img_set_input_color_profile_from_description pdf2img_set_render_intent pdf2img_set_color_profile_from_buffer pdf2img_set_color_profile_from_description

Availability	All platforms

Technical Notes

1. Any member of this group of functions pdf2img_set_input_color_profile_from_buffer, pdf2img_set_input_color_profile_from_description and pdf2img_set_input_color_profile_from_output_intent may be called repeatedly. A call succeeds if the specified (RGB) input profile exists, overriding a previous call if necessary. Thus, if several input profiles are specified, pdf2imglib uses the last one. Similarly, if several output profiles are specified, pdf2imglib uses the last output profile.
2. These functions are for setting the input (RGB) color For specifying an output profile, see pdf2img_set_color_profile_from_buffer, pdf2img_set_color_profile_from_description or pdf2img_set_color_profile_from_output_intent.
3. An output intent is a dictionary in the document's OutputIntents array, as described in the PDF Reference, ISO 32000-1:2008, Document Management-Portable Document Format-Part 1: PDF 1.7, section 14.11.5, page 633. To use an output intent's color profile, pdf2imglib matches the description supplied by the client with its dictionary values as follows:
   - use the first embedded profile with the specified OutputCondition value
   - use the first embedded profile with the specified OutputConditionIdentifier value
   - use the first embedded profile with the specified S (subtype) value

pdf2img_set_max_band_memory

(ImageConversion IC, unsigned int newVal)

Return Value: int

Description	PDF2IMG checks to see if it has enough memory to rasterize a PDF page to JPEG or TIF in one pass. If not, it rasterizes the page in bands (strips) in order to use smaller chunks of memory. Then the software reassembles the bitmaps into the finished output image. This call allows you to change the threshold value for the band rasterization process setting the size in bytes above which a banding conversion will be performed instead of attempting a single full-page conversion in one step.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int newVal**: number of bytes for the threshold above which banded conversion will be used. from 1000000 (one million) to 4200000000 (4.2 billion) with a default of 300000000 (300 million)
Return Value	Int

Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

1. You will typically not need the -maxbandmem and pdf2img_set_max_band_memory calls; the banding process is automatic and used only when needed, based on input page size and available memory.
2. The limit for JPEG output image size is 65535 x 65535 TIF output has been tested up to a band of 68898 x 34449 pixels in size.
3. The maximum allowed value for -maxbandmem as coded in the sample driver program pdf2img is approximately 2100000000 (2.1 billion), due to the ASCII-to-integer conversion process within the Higher values are allowed if passed in through this pdf2img_set_max_band_memory API call.

pdf2img_set_multipage

(ImageConversion IC, unsigned short int multipage)

Return Value: Int

Description	If a non-zero multipage value is provided as the second argument in this call pdf2imglib will create multipage TIFF output files. Each converted PDF page will correspond to a page in the multipage TIFF output. This function is for TIFF output only.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned short int multipage**: Multipage output flag: **0**: Single-page output **1 (or any non-zero)**: Multipage output *(Default 0)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h

Related Methods	pdf2img_end_multipage pdf2img_start_multipage
Availability	All platforms

pdf2img_set_OPP

(ImageConversion IC, int newVal)

Return Value: Int

Description	This call will toggle OverPrint Preview (OPP) on or off in the generated image. If set to true pdf2imglib will generate a graphic that represents what the input PDF page would look like after being printed. It will account for ink overprinting.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **int newVal**: Set OPP flag: **0**: Do not set **1 (or any non-zero positive)**: Set OPP flag *(Default 0)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_output_region

(ImageConversion IC, PDFRegionCode rCode)

Return Value: Int

Description	This call specifies the region of the PDF page which may be rasterized. All regions except BOUNDINGbox are defined in the "Page Boundaries" section in the PDF Reference.

	See ISO 32000-1:2008 Document Management-Portable Document Format-Part 1: PDF 1.7 section 14.11.2 page 627. The BOUNDINGbox region is the area enclosing all visible page markings.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **PDFRegionCode rCode**: Region of PDF page to rasterize: CROPbox MEDIAbox ARTbox TRIMbox BLEEDbox BOUNDINGbox (*Default CROPbox*)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

PDFRegionCode: rCode values below represent the following regions of the PDF page. For full details on all values for PDFRegionCode rCode except BOUNDINGbox. See the “Page Boundaries” section in the PDF Reference, ISO 32000-1:2008 Document Management-Portable Document Format-Part 1: PDF 1.7, section 14.11.2, page 627.

Page Boundary Values and Areas

PDF Region	Description	Default Value
CROPBox	Region to which the contents of the page are to be clipped (cropped) when displayed or printed	None
MEDIAbox	Dimensions of physical medium on which the page is to be printed. This includes any extended areas surrounding the finished page for bleed or printing marks or other purposes.	None
ARTbox	Extent of the page's meaningful content (including potential white space) as intended by the page's creator.	Page's Crop Box

PDF Region	Description	Default Value
TRIMbox	Intended dimensions of the finished page after trimming	Page's Crop Box
BLEEDbox	Region to which page contents should be clipped when output in a production environment. May include extra bleed area needed to accommodate physical limitations of cutting folding and trimming equipment.	Page's Crop Box
BOUNDINGbox	Area enclosing all visible page markings	None

pdf2img_set_output_type

(ImageConversion IC, OutputTypeCode oCode)

Return Value: Int

Description	This call specifies the output graphic type or types into which PDF2IMG will convert the document. See Technical Notes below.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **OutputTypeCode oCode**: Desired output graphic format for conversion: EPSoutput TIFFoutput JPEGoutput BMPoutput PNGoutput RAWoutput GIFoutput (No Default)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

If specifying GIFoutput as one of a group of two or more calls to pdf2img_set_output_type, specify GIFoutput last. Some internal functions assume that the GIF output format is always the last item in a list of multiple formats.

Output Format	Output Code	Format Description
EPS	EPSoutput	Encapsulated PostScript
TIFF	TIFFoutput	Tagged Image File Format
JPEG/JPG	JPEGoutput	Joint Photographic Experts Group
BMP	BMPoutput	Bitmap
PNG	PNGoutput	Portable Network Graphics
RAW	RAWoutput	Uncompressed format
GIF	GIFoutput	Graphics Interchange Format

pdf2img_set_pdf_output_type

(ImageConversion IC, PDFOutputType pCode)

Return Value: Int

Description	If selecting PDF as the output file type, select the method to use in generating the PDF output file or files.	
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion	
	PDFRegionCode pCode: Method for generating PDF output content:	
	Rasterize	Rasterize each page in the PDF source document and create a single PDF output file. (*Default*)
	RasterizeAndSplit	Rasterize each page in the PDF source document and export every rasterized page to a separate PDF output file.
	Split	Generate PDF output without rasterizing the source content. Create a single PDF output file for each page in the input document.
Return Value	Int	
Exceptions		

Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_quality

(ImageConversion IC, unsigned int quality)

Return Value: Int

Description	For JPG/JPEG output format the value set via this call will determine the output image quality. It represents the balance you want between generating a small output file (but a low-resolution image) versus a high-resolution image (but a large output file). Valid quality values range from 1 to 100. The smallest file would be 1 and 100 would be for the highest quality image.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int quality**: Desired output image quality from 1 to 100 *(Default 75)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

Technical Notes

Lowering the quality value will not only lower the detail of the image, but also lower the precision of the colors (as compared with the original input). For example, rendering a JPEG image at 50% quality rather than some value significantly higher may yield a result that not only shows less detail but also contains slightly different shades of color.

pdf2img_set_render_intent

(ImageConversion IC, RenderIntent riCode)

Return Value: Int

Description	This call sets the rendering intent for the ICC profile. If a profile is supplied the value defaults to the rendering intent provided within that ICC profile. If the profile is not supplied the intent defaults to perceptual.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **RenderIntent riCode**: one of the five intent values available to choose from perceptual relative saturation absolute profile
Return Value	Int: 0 for success
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_color_profile_from_buffer pdf2img_set_color_profile_from_description
Availability	All platforms

Technical Notes

The riCode parameter offers five intent values to choose from: perceptual, relative, saturation, absolute or profile.

Intent Values

perceptual	Generally used for photography. This method does not map colors one for one. Rather the method estimates to match colors. Hence it often provides the most pleasing result but not necessarily the most accurate. If you do not specify a color profile in the colorprofile option the intent value defaults to perceptual.

relative	Generally used for photography. The relative method uses an algorithm to select the closest possible color map to be true to the specified color.
saturation	Commonly used in charts and diagrams with a limited palette of colors. The saturation is used where hue is not as important.
absolute	Often used to select a specific color or set of colors for drawings or designs. For PDF2IMG absolute will serve to reproduce the exact colors provided in the original PDF document. A common reason for using absolute would be to reproduce the color used in a corporate logo such as IBM Blue. The color is changed by selecting a defined match. This method does not use a conversion algorithm to select the closest color available.
profile	If you specify a color profile via either pdf2img_set_color_profile_from_buffer or pdf2img_set_color_profile_from_description the riCode value defaults to profile. In that case PDF2IMG will use the rendering intent provided with the ICC color profile currently in use. For example, the Adobe RGB 1998 color profile uses Relative Colorimetric as its rendering intent. So if PDF2IMG specifies Adobe RGB 1998 as the color profile, and you don't enter a value via pdf2img_set_render_intent, the product will use relative as the color rendering intent.

pdf2img_set_resampler

(ImageConversion IC, ResamplerCode rsCode)

Return Value: Int

Description	Originally PDF2IMG would convert images without resampling. In some cases would cause unwanted artifacts or loss of detail in smaller or low-resolution output images such as thumbnails. Automatic resampling was introduced to enhance the quality of images when they were converted. If you use the default value of AutoResampler and if any of the following conditions are true images will first be rasterized to 150 PPI. After that a bicubic downsampling to the desired target values will be applied: -pixelcount:h is less than one half of the default input height

	-pixelcount:w is less than one half of the default input width -resolution is less than 150 Specifying an rsCode of BicubicResampler will apply the resampler unconditionally. Specifying NoResampler will turn it off completely for output consistent with prior releases of PDF2IMG.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **ResamplerCode rsCode**: DSelected resampler setting *(Default AutoResampler)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_reverse

(ImageConversion IC, unsigned short int reverse)

Return Value: Int

Description	If a positive non-zero reverse value is provided as the second argument in this call pdf2imglib will reverse black and white in the output image creating a negative image. This function is for grayscale output only.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned short int reverse**: Reverse output flag: 0: Do not reverse 1 (or any non-zero positive): Reverse output black/white values *(Grayscale output only; Default* 0*)*
Return Value	Int

Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_size_pixels

(ImageConversion IC, unsigned Int hPixels, unsigned Int vPixels)

Return Value: Int

Description	This call sets the page conversion to emit an image of exactly hPixels wide and vPixels tall. This call does not affect the image resolution but only the output size. Only one argument is required; the other argument can be zero (0) which will automatically scale the image proportionately using whichever value was given as a fixed dimension and floating the size of the other dimension as needed (No Default).
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int hPixels**: desired horizontal output width in pixels (or 0) **unsigned int vPixels**: desired vertical output height in pixels (or 0)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_smoothing

(ImageConversion IC, unsigned short Int Smoothing)

Return Value: Int

Description	This call sets Smoothing flags as desired for Text or Line Art and/or Image Smoothing. All settings are independent; valid values for this call are a logical OR of SmoothingCode values. For full Smoothing operation set all flags.

Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned short int Smoothing**: logical OR of available Smoothing flags: PDFSmoothNone: None PDFSmoothText: Text Smoothing PDFSmoothArt: Line Art Smoothing PDFSmoothImage: Image Smoothing (*No Default*)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_vert_res

(ImageConversion IC, unsigned Int vRes)

Return Value: Int

Description	This call sets the vertical resolution of the output expressed in dots/inch.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned int vRes**: vertical output resolution in dots per Valid range is 12 to 2400. *(Default 300)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_set_horiz_res
Availability	All platforms

pdf2img_setasprinted

(ImageConversion IC, Int newVal)

Return Value: Int

Description	By default a rendered page is converted to an image as it would be shown on screen but not on paper. Non-printing annotations will be shown and printable annotations will not. This call will set the asprinted flag and reverse those distinctions: the image will represent the PDF page in its printed form and printable annotations will appear. Non-printing annotations (those only used for display on a screen) will be suppressed.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **int newVal**: Set asprinted flag: 0: Do not set 1 (or any non-zero positive): Set asprinted flag (Default: 0)
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_setprintannot
Availability	All platforms

Technical Notes

This command can be overridden by the pdf2img_setprintannot

pdf2img_setprintannot

(ImageConversion IC, Int newVal)

Return Value: Int

Description	As with the command-line -noannot flag this call adds the capability to suppress displayable annotations from the converted output.

Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **int newVal**: Call with 0 value to suppress annotations *(No Default)*
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_setasprinted
Availability	All platforms

Technical Notes

1. This command should be used with Many page objects can be various forms of annotation, some more obvious than others, so you should check your output carefully to ensure that you are suppressing only those annotations that you want to block, and no others.
2. This command will override the pdf2img_setasprinted setting.

pdf2img_start_multipage

(ImageConversion IC, const char *ImageName)

Return Value: Int

Description	This call will start multipage TIFF output of the current input PDF document rather than the default of single-page sequentially- named output files. *(TIFF only; No default)*
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **const char *imageName**: File prefix to be assigned to output TIFF file name produced
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_end_multipage pdf2img_set_multipage

Availability	All platforms

pdf2img_term ()

Return Value: Int

Description	This calling argument terminates PDF2IMG after use. If used in a multi-threaded application each calling thread must do its own termination after use.
Parameters	
Return Value	Int
Exceptions	
Header	pdf2imglib.h
Related Methods	pdf2img_init pdf2img_init_ex
Availability	All platforms

pdf2img_verify_options

(ImageConversion IC)

Return Value: Int

Description	This call verifies that the conversion options supplied are valid in context. If so a zero (0) is returned; if any are not valid a non-zero value is returned.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion
Return Value	Int: 0 if conversion options are valid; non-zero if any are not
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_version_string()

Return Value: const char *

Description	This call returns a string representation (in English) of the pdf2imglib version in use. Do not release this string.
Parameters	
Return Value	**const char ***: Current pdf2imglib version in use
Exceptions	
Header	pdf2imglib.h
Related Methods	
Availability	All platforms

pdf2img_set_split_layers

(ImageConversion IC, unsigned short int newVal)

Return Value: int

Description	Split the document by Layers into separate output images. This function is for TIFF, JPEG, PNG, GIF, or BMP output only.
Parameters	**ImageConversion iC**: the data structure containing the specifications for the active current conversion **unsigned short int newVal**: 0 - Don't split by Layers 1 - Split by Layers (*Default 0*)
Return Value	Int
Exceptions	
Header	pdf2imglib.h

Related Methods	
Availability	All platforms

Appendix: API Calls, .NET Interface

We start with a description of the PDF2IMG constructor. The call opens with the PDF2IMG constructor, followed by LoadInput and one or more of the other method statements.

PDF2IMG

```
PDF2IMG(IList fonts = null, Boolean ignoreSystemFonts = false, Boolean
ignoreCurrentDirectory = false)
```

The PDF2IMG constructor creates a new instance of the PDF2IMG class.

Fonts	An optional parameter that specifies the list of local directories for the system to search to find font files and other resources. Defaults to null; provide a list of strings to define the directories as needed.
ignoreSystemFonts	This optional parameter is set to tell PDF2IMG to ignore the system font directories. Normally PDF2IMG will scan the default system resource directories to find font files that are stored locally, such as .otf, .ttf, and .cmap files. This process begins when the system starts, but the font scanning takes time. This option allows you to turn off the search process if it is not needed for a given PDF document or set of PDF documents. You can also direct PDF2IMG to search in a specific set of locations instead, a set of locations that you provide, defined using the Fonts parameter described above.
ignoreCurrentDirectory	This optional parameter is set to direct PDF2IMG to ignore the current directory when searching for font files.

An exception may be thrown if there is a problem in initializing the system.

Note that you can build your own executable file for PDF2IMG. But if you are working with an evaluation version of PDF2IMG, you must generate a license file, and store that license file in the same directory where you place your executable file. For more details on working with license files, see “**Error! Reference source not found.**” on page 2.

LoadInput

```
LoadInput(Byte[] inputBuffer, String userPassword = null)
```

This method loads the input PDF or XPS document into PDF2IMG.

inputBuffer	The memory buffer containing the input document to be imported into PDF2IMG.
userPassword	An optional parameter to hold the password of the PDF document if a password is required for the document to be opened.

This method returns the number of pages in the input file after it is loaded. An exception may be thrown if a problem appears in loading the document.

CheckForMissingAppearances

```
CheckForMissingAppearances(UInt32 firstPage, UInt32 lastPage)
```

This method checks pages within the specified range for annotations or form fields that cannot be rendered.

Define a range of page numbers for pages to be checked using the firstPage and lastPage parameters.

The method returns the number of appearances that cannot be rendered in an export graphics file. An annotation may specify one or more appearance streams that define how the annotation appears to a user. For example, a form field could change color if the user rolls the cursor over the field, or a link could change color after it is clicked.

An exception may be thrown if an error occurs while looking through appearances.

ConvertPageToImage

```
ConvertPageToImage(UInt32 pageNumber, String outputPath)
```

This method converts a specified page in a PDF document to an output image file, such as JPEG, using the Image Conversion Options that were previously set. The ImageConversionOptions class defines the available options.

pageNumber	Page to be converted
outputPath	Path of the image to be created

An exception may be thrown if a problem appears in converting the page to an image, or if the Image Conversion Options are not valid.

ConvertAllPagesToTIFFImage

```
ConvertAllPagesToTIFFImage(String outputPath)
```

This method converts all of the pages in a PDF input document to a single multi-page TIFF image file. Each page is stored in memory before it is converted.

outputPath	The local path and file name where the output TIFF image file will be saved.

An exception may be thrown if a problem appears in converting any of the pages in the PDF document to TIFF, or if the Image Conversion Options are not valid.

GetPageBoxWithWhiteSpaceRemoved

```
GetPageBoxWithWhiteSpaceRemoved()
```

This method retrieves four sets of coordinates—top, left, right, and bottom—for a CustomRegion on a page. These coordinates could be used to form a tight-fitting bounding box around all of the text, graphics, and images on the page, so that the white margins around the content on the page could be removed.

SetImageConversionOptions

```
SetImageConversionOptions(ImageConversionOptions options)
```

Use this method to define the Image Conversion Options to be used when converting a PDF document to graphic image file output. An exception may be thrown if the Image Conversion Options are not valid.

Use the “options” parameter to define the conversion options. The ImageConversionOptions class defines the available options.

outputType	The file format of the output image created. The default value is TIFF. Other values include EPS, JPG, BMP, PNG, RAW, PDF, and GIF.
pdfOutputType	If selecting PDF as the output file type, select the method to use in generating the PDF output file or files. This can be Rasterize, RasterizeAndSplit, or Split. Rasterize means that all of the pages in the PDF source document are converted into image files and then the content is output as a single PDF document. For RasterizeAndSplit, the pages in the source file are rasterized, and then output as separate

	individual PDF documents, one per page. Split means that the source PDF document is simply converted into a series of individual PDF documents, one per page, but the output is not rasterized. The default is Rasterize.
bitsPerChannel	The Bits Per Channel of the output image created. Defaults to 8.
horizontalResolution	The horizontal resolution of the output image created. The default value is 300 dots per inch. Valid values are between 12 and 2400.
verticalResolution	The vertical resolution of the output image created. The default value is 300 dots per inch. Valid values are between 12 and 2400.
colorSpace	The Colorspace of the output image created. The default value is RGB. Others include Gray, RGBA, CMYK, and LAB.
tiffConversionOptions	The options available for creation of the output TIFF. The options include setting the blackisone reference value (which is rarely used) and selecting a compression method, which defaults to lzw. Other compression options include None, G3, G4, and JPEG. If you want more detail, see the description of the **blackisone** optional command line argument. The concepts related to **blackisone** used here for the tiffConversionOptions option within the SetImageConversionOptions method in the .NET Interface are the same as described for the PDF2IMG C language interface.
jpegConversionOptions	The options available for creation of the output JPEG. This includes the output image quality; either a small output file with low resolution, or a larger file that has higher resolution. The quality value can range from 1 to 100, with the default set to 75. A file with a quality value of 100 would be the highest quality image possible.
overPrintPreview	Whether or not ink overprinting is applied to the output image created. If set to True the software will generate a graphic that represents what the input PDF page would look like after being printed and will account for ink overprinting. Defaults to false.
regionOfInterest	The region of the page rasterized for the output image created. The default value is CropBox. Other options include MediaBox, Artbox,

	Trimbox, Bleedbox, BoundingBox, and CustomBox. If you select CustomBox you need to define coordinates in customRegion.
outputColorProfileOptions	The options used for the Output Color Profile of the output image created.
inputColorProfileOptions	The options used for the Input Color Profile of the output image created.
useColorManagement	Sets whether color management is used during creation of the output image. The default value is true.
enhanceThinLines	Sets whether thin lines are generated in the generated output image. The default value is true, to generate thin line renderings as they would appear when generated by Adobe Reader or Acrobat.
resamplerUsed	Sets whether resampling is used in creation of the output image. The default value is Auto. Other options include Bicubic or none. Resampling is a method used to resize graphics image files, such as JPEG or PNG. The resampling process tends to create a smoother image when the image size increases or decreases. To increase the size of the image, more pixels are added to the image, and pixels are removed from an image to make it smaller. In each case the image loses resolution.
reverseGrayscale	Sets whether grayscale should be reversed in the created output image. The default value is false.
asPrinted	Sets if the document should be converted as if it was being printed when creating the output image. The default value is false.
printAnnotations	Sets if displayable annotations are suppressed the output image. The default value is false, annotations should be printed.
ouputWidth	The width of the output image created, in pixels. Defaults to 0.
outputHeight	The height of the output image created, in pixels. Defaults to 0.
smoothing	Smoothing used for the output image created. Smoothing, or anti-aliasing, is useful when creating low resolution outputs. The default value is All. Other options include none, text, line art, or image.

thresholdForBandRasterization	The threshold in bytes to use for determining if rasterization should be done in bands in order to use less memory. PDF2IMG checks to see if it has enough memory to rasterize a PDF page to JPEG or TIF in one pass. If not, it rasterizes the page in bands (strips) in order to use smaller chunks of memory. Then the software reassembles the bitmaps into the finished output image. This setting allows you to change the threshold value for the band rasterization process by defining the maximum size, in bytes. An image with a size greater than this value will be rasterized in a banding conversion, rather than attempting a single full-page conversion in one step. The default value is 300000000, or 30 million bytes. This is only applicable for TIFF or JPEG output.
customRegion	The custom region of the page rasterized for the output image created. When defining a region to rasterize on an input page, rather than using a standard value like cropbox or mediabox , it is possible to define a custom region for rasterizing by providing a set of four coordinates, [left],[top],[right],[bottom] in PDF units. Provide these coordinates in customRegion if you define regionOfInterest to be CustomBox.
SplitByLayers	Split the document by Layers into separate output images. This is for TIFF, JPEG, PNG, GIF, or BMP output only.