Datalogics

DATALOGICS PDF2IMG

PDF2IMG User Guide

For additional information, contact:
Datalogics, Incorporated
101 North Wacker Drive, Suite 1800
Chicago, Illinois 60606-7301

Phone: 312-853-8200
Fax: 312-853-8282

www.datalogics.com
support@datalogics.com

Table of Contents

Getting *Started*

This chapter introduces *Datalogics PDF2IMG*. Experienced users may want to skip directly to the section "What's New in This Release" on page 1.5 for information on the latest enhancements and additions. If you are a new user with an Evaluation release of *PDF2IMG*, please see "Notes for Evaluation Users" on page 2.2 for important details on use of the required `eval.lic` file.

An Introduction to *PDF2IMG*

Datalogics PDF2IMG provides a conversion utility that allows you to transform PDF (Portable Document Format) or (on non-Macintosh platforms) XPS (XML Paper Specification) documents into one or more image files, in your choice of graphic format and with your choice of options, to give as much flexibility as possible to your converted output for your different publishing needs. It can be built and run as a standalone executable, or incorporated into your C++, C#, .NET or Visual Basic application.

This product was created using the *Adobe PDF Library*, to facilitate the rapid creation of graphic output via a true Adobe internal interface for interpreting the input PDF or XPS documents. Sample applications are included to demonstrate its building and use in various development environments.

The *Adobe PDF Library*, plus accompanying *Datalogics Interface* (*DLI*) and *Datalogics Enhancements* (*DLE*), is also available separately from Datalogics, on multiple platforms, to enable you to add true Adobe PDF processing and output capability to your own applications. If you need more functionality beyond that which is provided by *PDF2IMG*, contact your Datalogics representative or visit the Datalogics website at `http://www.datalogics.com/products/pdfl/pdflibrary.asp` for more details on licensing the *Adobe PDF Library*, *DLI* and *DLE* for use in your own applications.

What You Should Know

This document is intended for programmers and other users who are familiar with PDF documents, graphic image file creation, graphic file manipulation and general computer processes, or by application designers who are constructing an application based on the *Adobe PDF Library* and *DLI* package.

How to Use this Book

This book has been created to guide you through the process of installing and operating the *PDF2IMG* utility.

This chapter, Getting Started, outlines the chapters to follow, explains the document conventions used here, and lists other related documentation which you may find useful for your work. Follow-on chapters will outline the steps needed to install *PDF2IMG* and perform conversion operations with it. Samples of input and output will be provided.

The following list provides an outline of the chapters as well as a brief description of their contents. Click on each Chapter title below to jump to its first page.

Chapter 1: **"Getting Started"** *(This chapter)* This introduces the *PDF2IMG* utility and describes the contents of this book.

Chapter 2: **"Installing PDF2IMG"** gives directions on how to install *PDF2IMG* on different system platforms, including file unpacking and environment-variable concerns.

Chapter 3: **"Building PDF2IMGCOM"** gives directions on how to build *PDF2IMGCOM* samples on Windows platforms, including file unpacking and invoking of the sample applications.

Chapter 4: **"Arguments and Options"** gives a general explanation of *PDF2IMG*, *PDF2IMGOM* and the available commands and options via the sample command-line and GUI interfaces.

The **"pdf2imglib Reference Appendix"** provides details on the calls available via `pdf2imglib` for incorporating *PDF2IMG* conversion operations directly in your application.

Document Conventions

The following documentation conventions appear throughout the manual to help you differentiate regular text from product and program names, and to distinguish command syntax.

- Product and program names are set in *italic* type.
- Multi-line examples are separated from the text and set in

```
Courier monospace
```

- Directory names and filenames are contained within the text and set in `Courier monospace`.
- Commands and arguments are contained within the text and set in `Courier monospace`. You can click on those in blue, as shown here, to jump directly to a full definition of that command or argument

in “Arguments and Options” on page 4.5. Similarly, a `pdf2imglib` function call in blue will jump you to that call’s full definition in the Reference Appendix of this guide.

- New terms are *italicized*.
- Page numbers in this book do not correspond to page numbers in the PDF file. The numbering scheme (*e.g.* 4.1 or A.10) indicates the chapter number (4) or appendix letter (A) first, followed by the page number (1 or 10), separated by a period.

What’s New in This Release

This section contains highlights of new additions and enhancements to *PDF2IMG* v2.2P1j and later. You should also check for any accompanying `readme.txt` files (one may accompany the software release files or the documentation files in their respective folders or directories). `readme.txt` files, if present, typically contain last-minute information on the current release of the software or the documentation files. A Documentation `readme.txt` file is not the same as a source-code `readme.txt` file; if both are present, be sure to review each one.

Minor version upgrades may be made as running changes rather than full releases, so the version or sub-version number of your release may be newer than those listed here. See the accompanying `readme.txt` files for the very latest changes and enhancements.

PDF2IMG v2.2 introduced the following changes, briefly outlined here.

New PDF Splitting Functionality

A new page-splitting enhancement has been introduced for PDF input files, allowing you to submit a multi-page input PDF document and receive individual, single-page output PDF document files in return.

This process runs by default whenever you submit a multi-page input PDF document and specify PDF for the output format. All *PDF2IMG* command-line options pertaining to multi-file output (*e.g.* `-digits`, `-firstonly`, `-output`) may be used in conjunction with this.

What’s New in Previous Releases

v2.1

Thin Line Enhancement On by Default

As of the v2.1P1b release, *PDF2IMG* now makes use of the "Enhance thin lines" rendering option (as seen in *Adobe Reader* or *Adobe Acrobat*) by default. As this is a change in default behavior, new options are provided for turning it off again to retain the past output appearance if desired. Select either one, depending on your method of using *PDF2IMG*:

- a new command line switch: `-noenhancethinlines`
- a new API function in *PDF2IMGCOM*: `pdf2img_set_enhance_thin_lines`

Banded Conversion of JPG and TIF Output

As of the v2.1P1a release, *PDF2IMG* will now use an enhanced conversion process to accommodate unusually large input PDF documents when converting to JPG or TIF.

Essentially, *PDF2IMG* now 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) in order to use smaller chunks of memory, then reassembles the bitmaps into the finished output image. This new banding approach (if needed for the input) will have no effect on the final output appearance, and will be transparent to the user; the memory allocation or banding is handled internally.

The *PDF2IMG* console application provided in this release now includes a new `-maxbandmem` command-line argument, and the *PDF2IMGCOM* interface now includes a new `pdf2img_set_max_band_memory` API call, used to adjust the default threshold value of 300,000,000 bytes, above which a banding process conversion will be used.

NOTE: 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.

v2.0

Macintosh OS X Platform Now Supported

As of the v2.0P1c release, *PDF2IMG* is now available for these Macintosh platforms:

- Tiger 10.4.8 or higher PPC
- Leopard 10.5.*x* PPC
- Tiger 10.4.8 or higher Intel
- Leopard 10.5.*x* Intel
- Snow Leopard 10.6.*x* Intel

Supported output formats include BMP, EPS, GIF, JPG, PNG, RAW and TIF. Conversion from XPS to PDF is not supported on this platform; input format must be PDF.

NOTE: BMP output is not supported on PPC machines.

Password Support

This release updated *PDF2IMG* to accommodate password-protected input PDF documents, allowing you to specify the password required to open an input PDF document for conversion. The *PDF2IMG* console application now includes a `-password` command-line argument, and the *PDF2IMGCOM* interface now includes `pdf2img_new_conversion_with_password` and `pdf2img_new_memconversion_with_password` API calls.

Rendering Enhancements Added via New *Adobe PDF Library* Build

This release updated *PDF2IMG* to use *Adobe PDF Library* v9.1.0PlusP2g, to reflect the latest rendering improvements and other ongoing enhancements to the PDF Library. Windows installers also include `pdf2imglib.lib` and Visual Studio Project files to enable you to build the `pdf2img.exe` driver.

XPS Input File Support

PDF2IMG v2.0 or higher now accepts XPS (XML Paper Specification) format files as input, offering the ability to convert these XPS files to PDF files, as well as to any of *PDF2IMG*'s other supported image types. Support for XPS input was added to the DLL and COM interfaces of *PDF2IMG*, and to the command-line and C# sample drivers.

PDF2IMG now includes the Standard `joboptions` file delivered with the *Adobe PDF Library* (under the name `xps2pdf.joboptions`), and supports converting XPS files with these settings. You may use other `joboptions` files, though Datalogics primarily supports the Standard `joboptions` file.

CMYK TIFF Output Format Support

PDF2IMG v2.0 added CMYK TIFF as a supported output configuration. It will be emitted to a default SWOP v2 output color profile by default.

Warnings on Nonrenderable Annotations and Forms

PDF2IMG 2.0 added a mechanism for users to query an opened document to determine whether a specified range of pages contains contents in annotations, AcroForms or XFA forms that lack normal appearance streams.

In such cases, rendered output from *PDF2IMG* will probably not match that of *Adobe Acrobat* or *Reader*: content that does appear when viewing the document with *Acrobat* or *Reader* will be missing from *PDF2IMG* output.

This capability is accessed in various ways, depending on the type of application:

DLL and COM Interfaces

A `pdf2img_check_for_missing_appearances` API in the *PDF2IMG* interfaces returns a Boolean indicator to report whether a supplied range of PDF pages has nonrenderable content. You may use the return result of the API to determine whether to proceed with rendering that range of pages.

C# and Command-line Sample Programs

These were updated to query the input PDF file for nonrenderable content over the pages specified for output. If nonrenderable content is found, you will be warned via a printed message to `stdout` (command-line example) or a warning dialog (C# example), but file processing will continue.

v1.5

Changes to Color Management

This release introduced some changes under the general category of Color Management, making use of the Color Management Module (*CMM*). In the past, *PDF2IMG* had been rendering pages to Device colorspaces, using *Adobe PDF Library* (*APDFL*) default settings for source colorspaces, and writing the resulting images without a color profile. However, *APDFL* default colorspace settings have been changing, and without a color profile in the output images, downstream procedures had no way to determine that a change in output had occurred, and no way of determining how to apply corrections (or whether any were needed).

Therefore, in order to better handle and track color management during conversion, we introduced two new enhancements to *PDF2IMG*:

1. There is a new interface to `pdf2imglib` of `pdf2img_set_colormanagement`, which may be set `True` or `False`. For the command-line sample application, a new `-cmm` switch was added (but note that this has now been replaced by a `-nocmm` switch as of the v1.6 release).
   See “Conversions with ICC Color Profiles” on page 4.29 for more details.
2. There were two new `pdf2imglib` interfaces, `pdf2img_get_profilesize` and `pdf2img_get_profiledata`, which enable an application wishing to access the image data in memory to access the profile data as well (assuming that there is indeed a profile associated with the image; these interfaces will return `0` and `NULL`, respectively, if there is no profile). They will return the size of the profile in bytes, and a pointer to a const string which contains the ICC Profile data.

New RGBa Output Colormodel Support

This release added support for RGBa (RGB + alpha channel) output for TIFF and PNG image formats. The `-colormodel` command-line option thus now acquires a new `"rgba"` choice, and the C# sample GUI application now offers a new `"rgba"` option in its ColorModel pulldown menu:

`colormodel=[gray | cmyk | rgb | rgba]` *(Default* `rgb`*)*

> **NOTE:** Please note that the COM component has changed to version 1.5 in this release. If you are using the COM component in a versioned manner, you will want to update your version to get this new COM v1.5 component with RGBa support.

New sRGB output of RGB images

This release now renders RGB images to sRGB output. The previous default setting used a value-preserving color path, but which tended to turn colors dull in some cases. The new sRGB output is intended to address this.

PDF2IMG Now Available on Win x64 Platforms

PDF2IMG is now available on Win x64 platforms, in addition to the original platform choices. There are two key differences between Win x64 and Win32 releases that you should be aware of:

- The Win x64 release does not include the COM interface (and therefore is not .NET compatible).
- It is provided as a self-extracting zipfile, rather than the MSI installer used for Win32 releases.

New Minimum Requirements for Linux & Solaris Releases

As part of the update to the underlying *Adobe PDF Library* components of *PDF2IMG* (see “PDF2IMG Now Using Adobe PDF Library v9.0” on page 1.10, below), releases of *PDF2IMG* v1.5P1a or higher on Linux or Solaris platforms now require gcc v4.1 and libstdc++.so.6. The gcc v3.2 compiler is no longer compatible.

PDF2IMG Now Using *Adobe PDF Library* v9.0

Releases of *PDF2IMG* v1.5P1a or higher are built with *Adobe PDF Library* v9.0, in keeping with the latest available *Adobe PDF Library* functionality.

NOTE: *Adobe PDF Library* modules are included for use in *PDF2IMG* work and graphic conversion, but are not intended for building PDF output applications. The Library and *Datalogics Interface* can be licensed separately for PDF generation or manipulation: see your Datalogics representative or `http://www.datalogics.com/products/pdfl/pdflibrary.asp` for more details.

Licensing Acknowledgements

The following open source packages are used in *DLI*. Click on any item below to jump to its licensing details below or on the following pages.

- libpng v1.0.8
- libjpeg
- libtiff

libpng v1.0.8

COPYRIGHT NOTICE, DISCLAIMER, and LICENSE

libpng v0.88

A `png_get_copyright` function is available, for convenient use in "about" boxes and the like:

```
printf("%s",png_get_copyright(NULL));
```

Also, the PNG logo (in PNG format, of course) is supplied in the files `pngbar.png` and `pngbar.jpg` (88x31) and `pngnow.png` (98x31).

libjpeg

libtiff

Installing

PDF2IMG

This chapter explains the process of installing *PDF2IMG*, including the unpacking process and the definition of environment variables where required. For releases of *PDF2IMG* v1.1P1p or higher, *PDF2IMGCOM* (when included in the release) will be installed simultaneously.

Installation

Installation of *PDF2IMG* and *PDF2IMGCOM* (where available) is straightforward. However, if desired you may alter or revise the `pdf2img.exe` front end as you see fit (the source code is provided), or incorporate it in your own application. (*PDF2IMG* is now distributed in `.DLL` or `.so` form, as appropriate for the platform. *PDF2IMGCOM* is currently available on Windows 32-bit platforms only.)

Notes for Evaluation Users

The Evaluation release of *PDF2IMG* is a time-limited version, and requires the included `eval.lic` file (accompanying the release files in your ftp download area) to be present at run time. If you have been provided this release as part of a product evaluation, the `eval.lic` file must be present when running your tests.

The evaluation release of *PDF2IMG* is license managed. You must either

- place the `eval.lic` file into the current working directory when running your *PDF2IMG* process, where the application will find it automatically, or
- set the `RLM_License` environment variable to point to it.

On Windows platforms, if you choose to set a Windows environment variable (a recommended strategy if running multiple applications in various locations), right-click on "My Computer" and go to "Properties". Click on the "Advanced" tab and select the "Environment Variables" button at the bottom. Click "New" and enter `RLM_License` as the name of the variable. For the value of the variable, enter the directory location of the `eval.lic` file; *e.g.*

```
RLM_License=C:\Datalogics\PDF2IMG\eval.lic
```

NOTE: The `eval.lic` file is time-limited, and will expire on the date shown in the file. (It is a plaintext file that you can view in any text editor.) Please contact your Datalogics Support representative if a new license file extension will be needed.

Macintosh Platforms

This *PDF2IMG* release is supported for these Macintosh platforms:

- Tiger 10.4.8 or higher PPC
- Leopard 10.5.*x* PPC
- Tiger 10.4.8 or higher Intel
- Leopard 10.5.*x* Intel
- Snow Leopard 10.6.*x* Intel

Supported output formats include BMP, EPS, GIF, JPG, PNG, RAW and TIF. Conversion from XPS to PDF is not supported on this platform; input format must be PDF.

NOTE: BMP output is not supported on PPC machines.

Installation

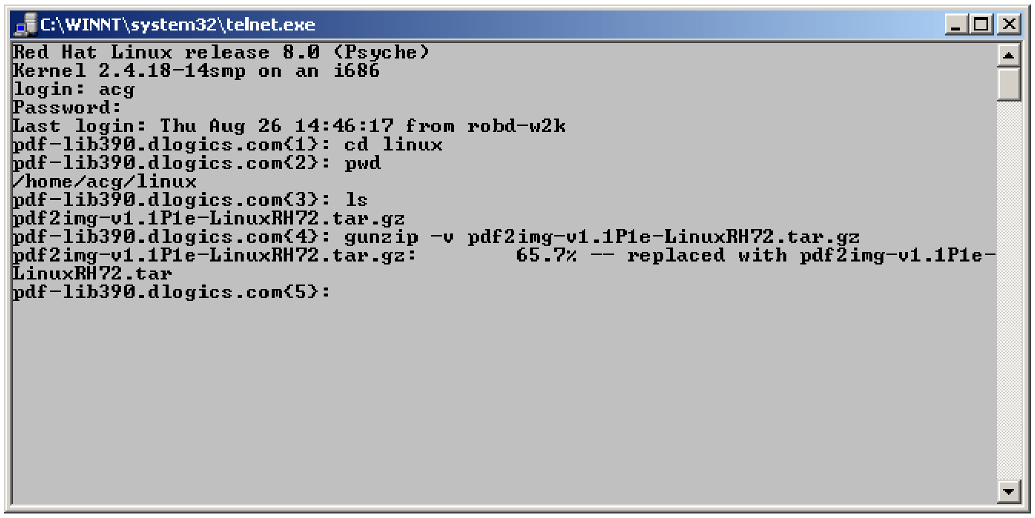
No formal installation is required. Double-click the `.dmg` file to expand and copy the folder to your Macintosh hard drive. As with Linux and Windows releases, *PDF2IMG* is run from the folder location. Open a terminal session and Change Directory to that location to run your application.

The *PDF2IMG* Macintosh Application will be located in a folder named `Applications` within the unpacked file (*i.e.* not the Macintosh `hd/Applications` folder).

The library files or Frameworks are in a folder named `Frameworks` within the unpacked file. The optional resource folders (`fonts` and `CMaps`) should be either copied into this location, or referenced on the command line via the `-fontlist` option.

UNIX Platforms

1 The release is packaged as a Gzipped `.tar` file. (*e.g.* `pdf2img-v1.1P1e-LinuxRH72.tar.gz` in the following sample screens)



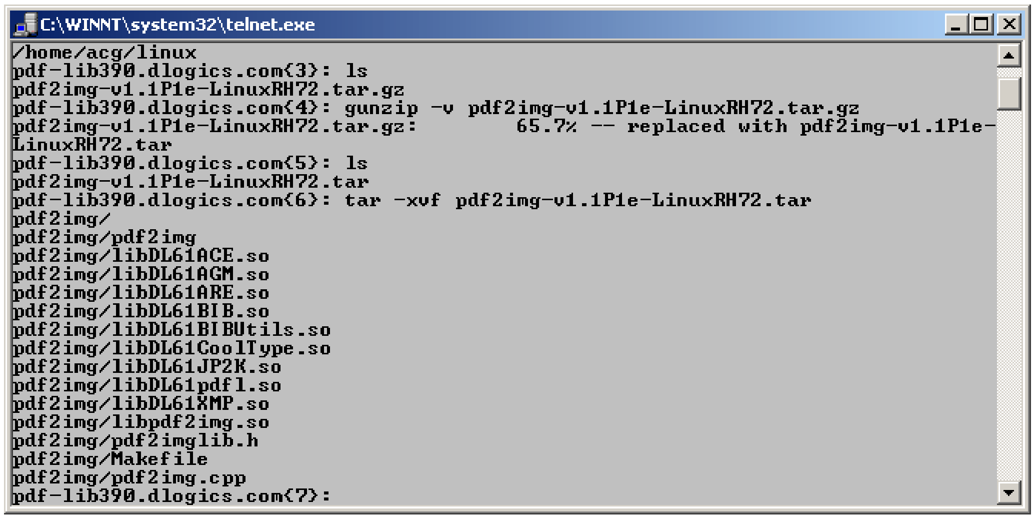
C:\WINNT\system32\telnet.exe

```
Red Hat Linux release 8.0 (Psyche)
Kernel 2.4.18-14smp on an i686
login: acg
Password:
Last login: Thu Aug 26 14:46:17 from robd-w2k
pdf-lib390.dlogics.com{1}: cd linux
pdf-lib390.dlogics.com{2}: pwd
/home/acg/linux
pdf-lib390.dlogics.com{3}: ls
pdf2img-v1.1P1e-LinuxRH72.tar.gz
pdf-lib390.dlogics.com{4}: gunzip -v pdf2img-v1.1P1e-LinuxRH72.tar.gz
pdf2img-v1.1P1e-LinuxRH72.tar.gz:        65.7% -- replaced with pdf2img-v1.1P1e-
LinuxRH72.tar
pdf-lib390.dlogics.com{5}:
```

After copying the file into the desired working directory, issue a `gunzip` command to access the `.tar` file as above; *e.g.*:

```
gunzip -v pdf2img-v1.1P1e-LinuxRH72.tar.gz
```

2 Next, unpack the .tar file itself:



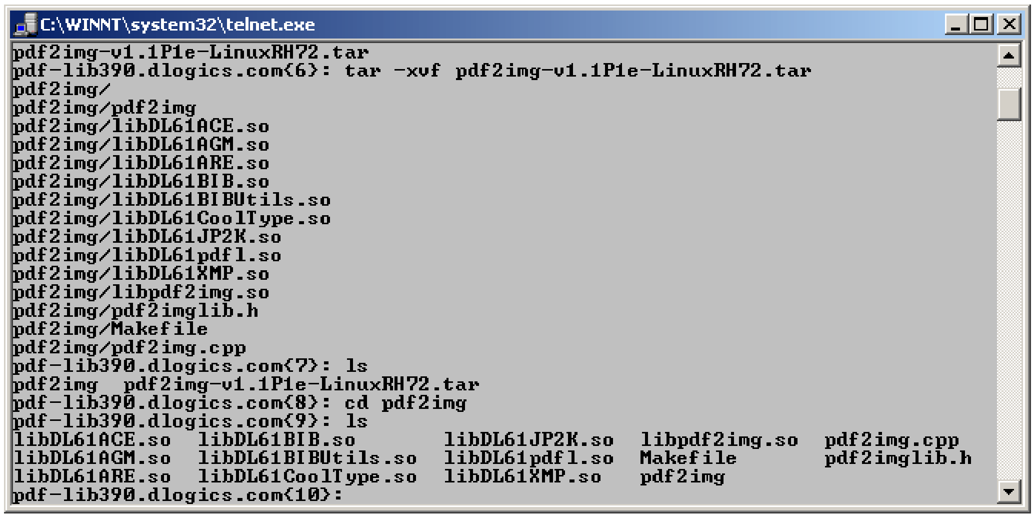
C:\WINNT\system32\telnet.exe

```
/home/acg/linux
pdf-lib390.dlogics.com{3}: ls
pdf2img-v1.1P1e-LinuxRH72.tar.gz
pdf-lib390.dlogics.com{4}: gunzip -v pdf2img-v1.1P1e-LinuxRH72.tar.gz
pdf2img-v1.1P1e-LinuxRH72.tar.gz:        65.7% -- replaced with pdf2img-v1.1P1e-
LinuxRH72.tar
pdf-lib390.dlogics.com{5}: ls
pdf2img-v1.1P1e-LinuxRH72.tar
pdf-lib390.dlogics.com{6}: tar -xvf pdf2img-v1.1P1e-LinuxRH72.tar
pdf2img/
pdf2img/pdf2img
pdf2img/libDL61ACE.so
pdf2img/libDL61AGM.so
pdf2img/libDL61ARE.so
pdf2img/libDL61BIB.so
pdf2img/libDL61BIBUtils.so
pdf2img/libDL61CoolType.so
pdf2img/libDL61JP2K.so
pdf2img/libDL61pdfl.so
pdf2img/libDL61XMP.so
pdf2img/libpdf2img.so
pdf2img/pdf2imglib.h
pdf2img/Makefile
pdf2img/pdf2img.cpp
pdf-lib390.dlogics.com{7}:
```

This will create a pdf2img directory and place all files in it; *e.g.*:

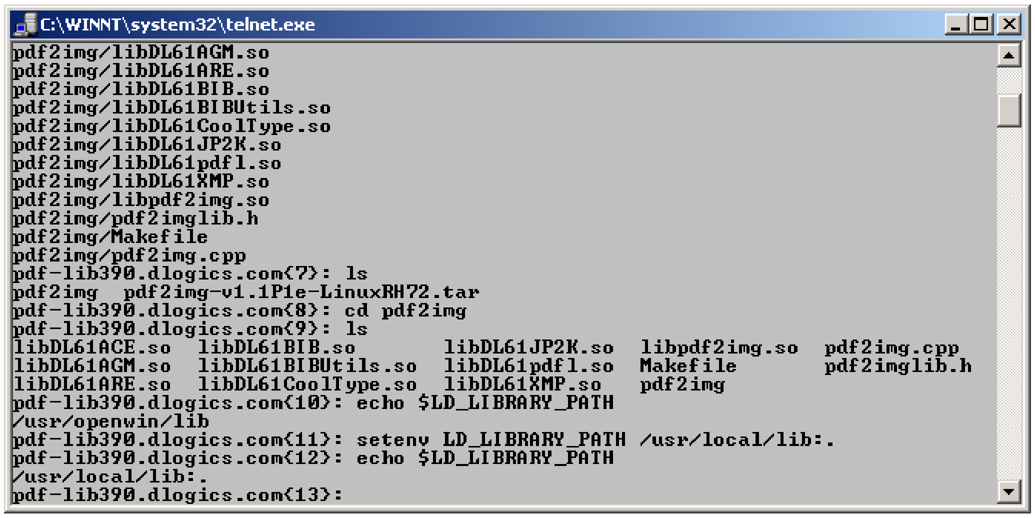
```
tar -xvf pdf2img-v1.1P1e-LinuxRH72.tar
```

3 Change your directory to the unpacked `pdf2img` folder and verify that the expected files are present:



```
pdf2img-v1.1P1e-LinuxRH72.tar
pdf-lib390.dlogics.com<6>: tar -xvf pdf2img-v1.1P1e-LinuxRH72.tar
pdf2img/
pdf2img/pdf2img
pdf2img/libDL61ACE.so
pdf2img/libDL61AGM.so
pdf2img/libDL61ARE.so
pdf2img/libDL61BIB.so
pdf2img/libDL61BIBUtils.so
pdf2img/libDL61CoolType.so
pdf2img/libDL61JP2K.so
pdf2img/libDL61pdfl.so
pdf2img/libDL61XMP.so
pdf2img/libpdf2img.so
pdf2img/pdf2imglib.h
pdf2img/Makefile
pdf2img/pdf2img.cpp
pdf-lib390.dlogics.com<7>: ls
pdf2img  pdf2img-v1.1P1e-LinuxRH72.tar
pdf-lib390.dlogics.com<8>: cd pdf2img
pdf-lib390.dlogics.com<9>: ls
libDL61ACE.so  libDL61BIB.so       libDL61JP2K.so  libpdf2img.so  pdf2img.cpp
libDL61AGM.so  libDL61BIBUtils.so  libDL61pdfl.so  Makefile       pdf2imglib.h
libDL61ARE.so  libDL61CoolType.so  libDL61XMP.so   pdf2img
pdf-lib390.dlogics.com<10>:
```

4 Before attempting to run the executable or execute the enclosed Makefile, you must define an `LD_LIBRARY_PATH` environment variable:



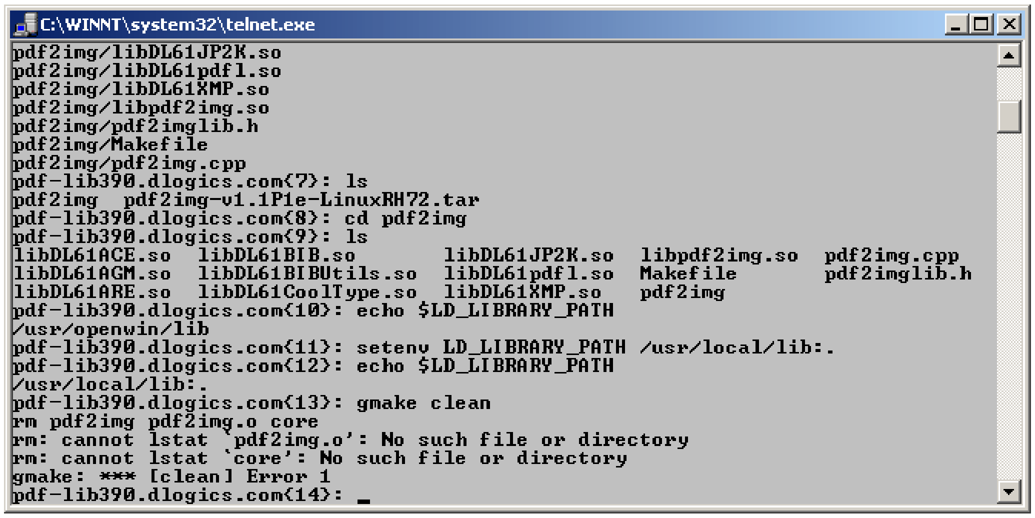
```
pdf2img/libDL61AGM.so
pdf2img/libDL61ARE.so
pdf2img/libDL61BIB.so
pdf2img/libDL61BIBUtils.so
pdf2img/libDL61CoolType.so
pdf2img/libDL61JP2K.so
pdf2img/libDL61pdfl.so
pdf2img/libDL61XMP.so
pdf2img/libpdf2img.so
pdf2img/pdf2imglib.h
pdf2img/Makefile
pdf2img/pdf2img.cpp
pdf-lib390.dlogics.com<7>: ls
pdf2img  pdf2img-v1.1P1e-LinuxRH72.tar
pdf-lib390.dlogics.com<8>: cd pdf2img
pdf-lib390.dlogics.com<9>: ls
libDL61ACE.so  libDL61BIB.so       libDL61JP2K.so  libpdf2img.so  pdf2img.cpp
libDL61AGM.so  libDL61BIBUtils.so  libDL61pdfl.so  Makefile       pdf2imglib.h
libDL61ARE.so  libDL61CoolType.so  libDL61XMP.so   pdf2img
pdf-lib390.dlogics.com<10>: echo $LD_LIBRARY_PATH
/usr/openwin/lib
pdf-lib390.dlogics.com<11>: setenv LD_LIBRARY_PATH /usr/local/lib:.
pdf-lib390.dlogics.com<12>: echo $LD_LIBRARY_PATH
/usr/local/lib:.
pdf-lib390.dlogics.com<13>:
```

This variable must locate all necessary files for building *PDF2IMG*; *e.g.*

```
setenv LD_LIBRARY_PATH /usr/local/lib:.
```

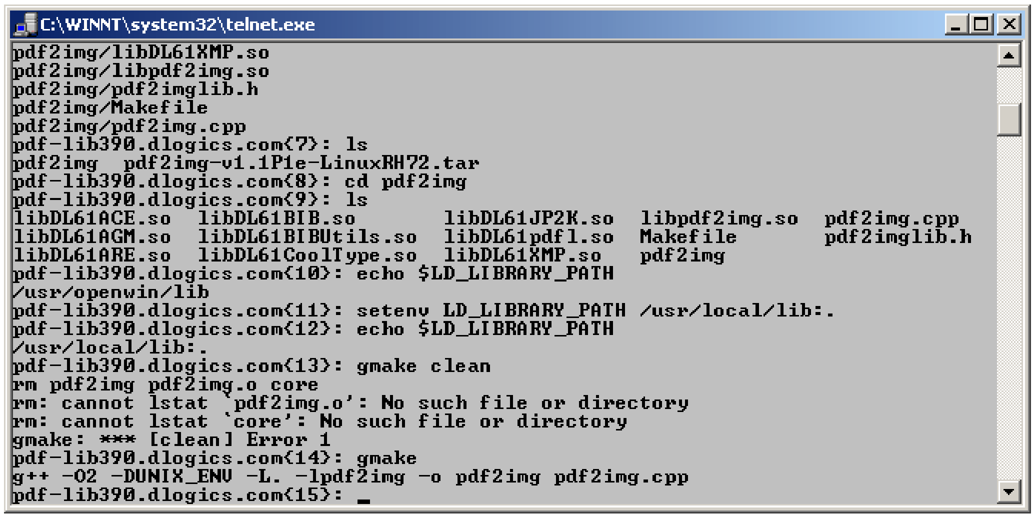
Please see “Environment Variable Definitions” on page 4.29 for full details. Alternatively, you can use the `-fontlist` command-line argument to specify resource location(s) at runtime. (See “Fontlist” on page 4.11 for full details.)

5 Issuing a `gmake clean` command will ensure that the initial *PDF2IMG* build will complete all necessary steps:



```
pdf2img/libDL61JP2K.so
pdf2img/libDL61pdfl.so
pdf2img/libDL61XMP.so
pdf2img/libpdf2img.so
pdf2img/pdf2imglib.h
pdf2img/Makefile
pdf2img/pdf2img.cpp
pdf-lib390.dlogics.com<7>: ls
pdf2img   pdf2img-v1.1P1e-LinuxRH72.tar
pdf-lib390.dlogics.com<8>: cd pdf2img
pdf-lib390.dlogics.com<9>: ls
libDL61ACE.so  libDL61BIB.so       libDL61JP2K.so  libpdf2img.so  pdf2img.cpp
libDL61AGM.so  libDL61BIBUtils.so  libDL61pdfl.so  Makefile       pdf2imglib.h
libDL61ARE.so  libDL61CoolType.so  libDL61XMP.so   pdf2img
pdf-lib390.dlogics.com<10>: echo $LD_LIBRARY_PATH
/usr/openwin/lib
pdf-lib390.dlogics.com<11>: setenv LD_LIBRARY_PATH /usr/local/lib:.
pdf-lib390.dlogics.com<12>: echo $LD_LIBRARY_PATH
/usr/local/lib:.
pdf-lib390.dlogics.com<13>: gmake clean
rm pdf2img pdf2img.o core
rm: cannot lstat `pdf2img.o': No such file or directory
rm: cannot lstat `core': No such file or directory
gmake: *** [clean] Error 1
pdf-lib390.dlogics.com<14>: _
```

6 If the preceding steps completed without problems, a `gmake` command here should produce a new build of *PDF2IMG*:

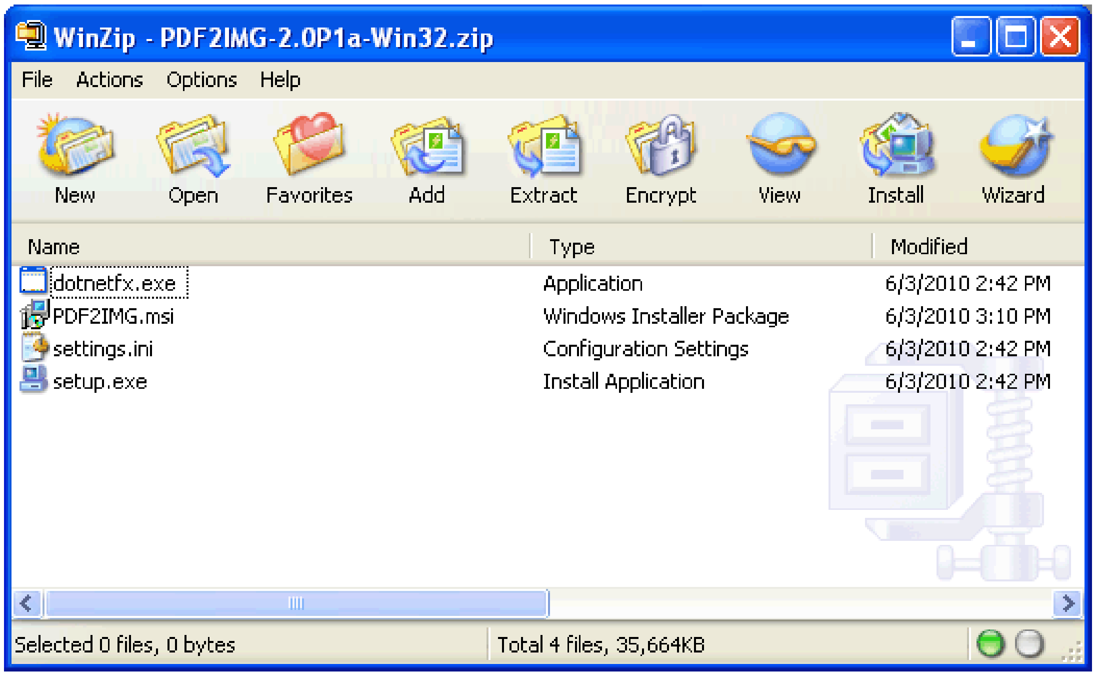


```
pdf2img/libDL61XMP.so
pdf2img/libpdf2img.so
pdf2img/pdf2imglib.h
pdf2img/Makefile
pdf2img/pdf2img.cpp
pdf-lib390.dlogics.com<7>: ls
pdf2img   pdf2img-v1.1P1e-LinuxRH72.tar
pdf-lib390.dlogics.com<8>: cd pdf2img
pdf-lib390.dlogics.com<9>: ls
libDL61ACE.so  libDL61BIB.so       libDL61JP2K.so  libpdf2img.so  pdf2img.cpp
libDL61AGM.so  libDL61pdfl.so      libDL61pdfl.so  Makefile       pdf2imglib.h
libDL61ARE.so  libDL61CoolType.so  libDL61XMP.so   pdf2img
pdf-lib390.dlogics.com<10>: echo $LD_LIBRARY_PATH
/usr/openwin/lib
pdf-lib390.dlogics.com<11>: setenv LD_LIBRARY_PATH /usr/local/lib:.
pdf-lib390.dlogics.com<12>: echo $LD_LIBRARY_PATH
/usr/local/lib:.
pdf-lib390.dlogics.com<13>: gmake clean
rm pdf2img pdf2img.o core
rm: cannot lstat `pdf2img.o': No such file or directory
rm: cannot lstat `core': No such file or directory
gmake: *** [clean] Error 1
pdf-lib390.dlogics.com<14>: gmake
g++ -O2 -DUNIX_ENV -L. -lpdf2img -o pdf2img pdf2img.cpp
pdf-lib390.dlogics.com<15>: _
```

NOTE: gcc v4.1 or higher is required to link against the `libpdf2img.so` shared library on UNIX systems.

Windows Platforms

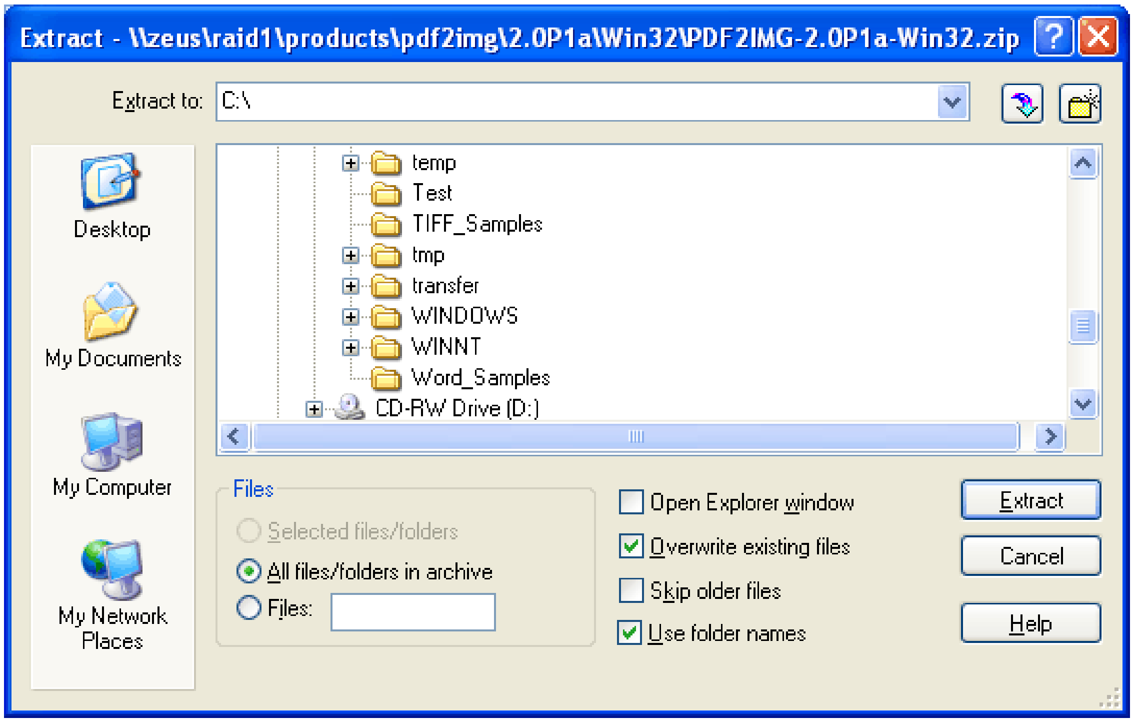
1 The release arrives as a .zip file. (*e.g.* `PDF2IMG-2.0P1a-Win32.zip` as shown in these examples):


WinZip - PDF2IMG-2.0P1a-Win32.zip
File Actions Options Help
New Open Favorites Add Extract Encrypt View Install Wizard

Name	Type	Modified
dotnetfx.exe	Application	6/3/2010 2:42 PM
PDF2IMG.msi	Windows Installer Package	6/3/2010 3:10 PM
settings.ini	Configuration Settings	6/3/2010 2:42 PM
setup.exe	Install Application	6/3/2010 2:42 PM

Selected 0 files, 0 bytes   Total 4 files, 35,664KB

2 Unzip the file in the desired location, such as (in this example) `C:\temp`:

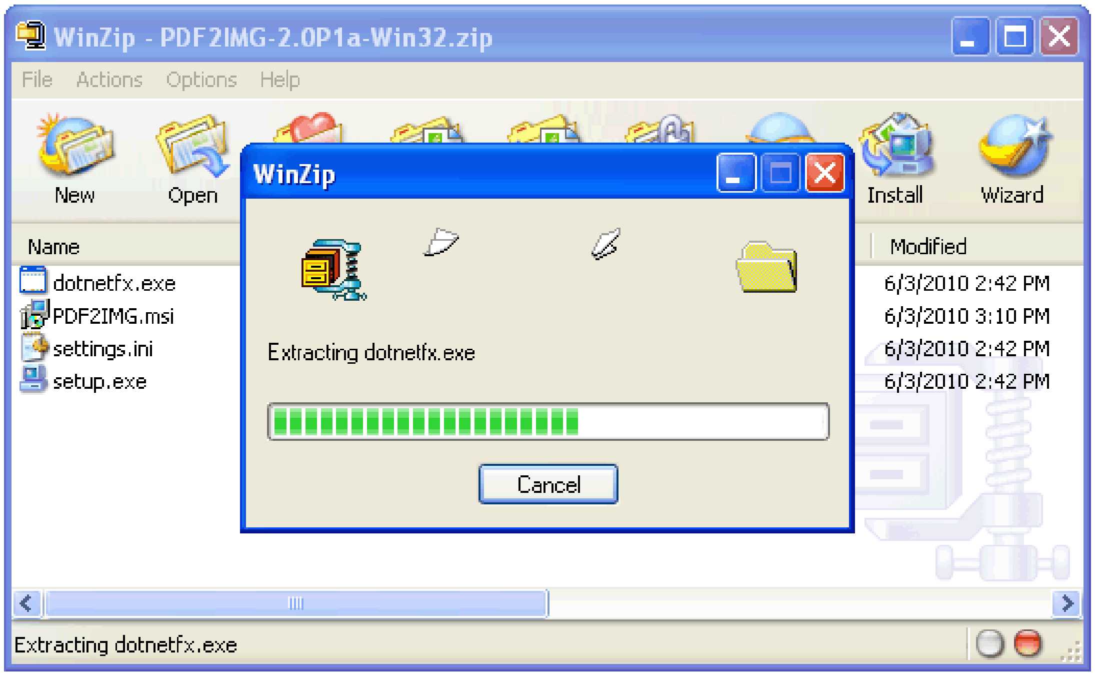


Extract to: C:\

Selected files/folders
[●] All files/folders in archive
Files:

[ ] Open Explorer window
[✓] Overwrite existing files
[ ] Skip older files
[✓] Use folder names

3 This will unpack all necessary files in that folder. A bar graphic will advance as it proceeds:

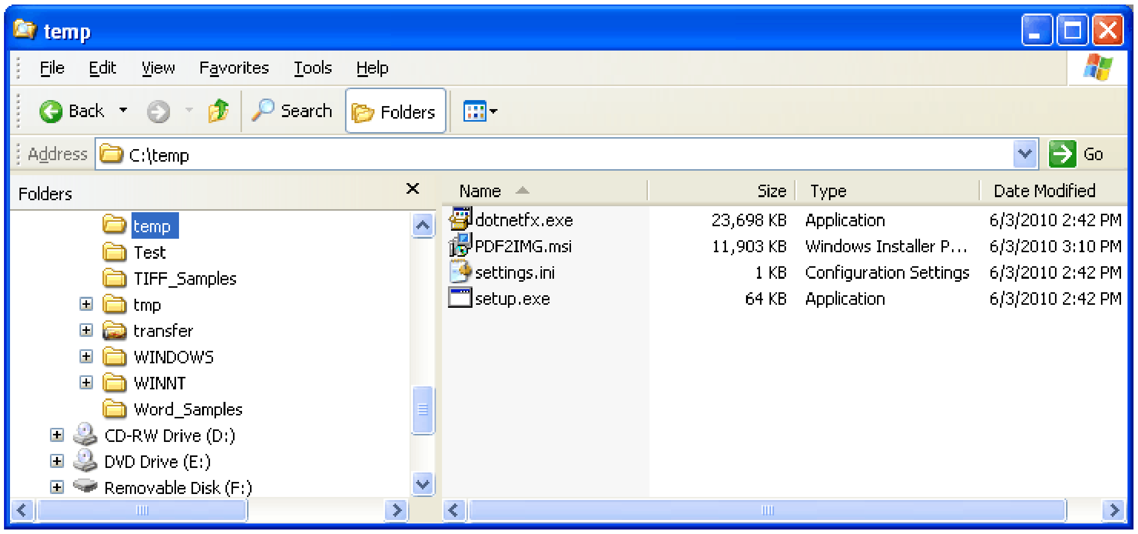


Name	Modified
dotnetfx.exe	6/3/2010 2:42 PM
PDF2IMG.msi	6/3/2010 3:10 PM
settings.ini	6/3/2010 2:42 PM
setup.exe	6/3/2010 2:42 PM

Extracting dotnetfx.exe

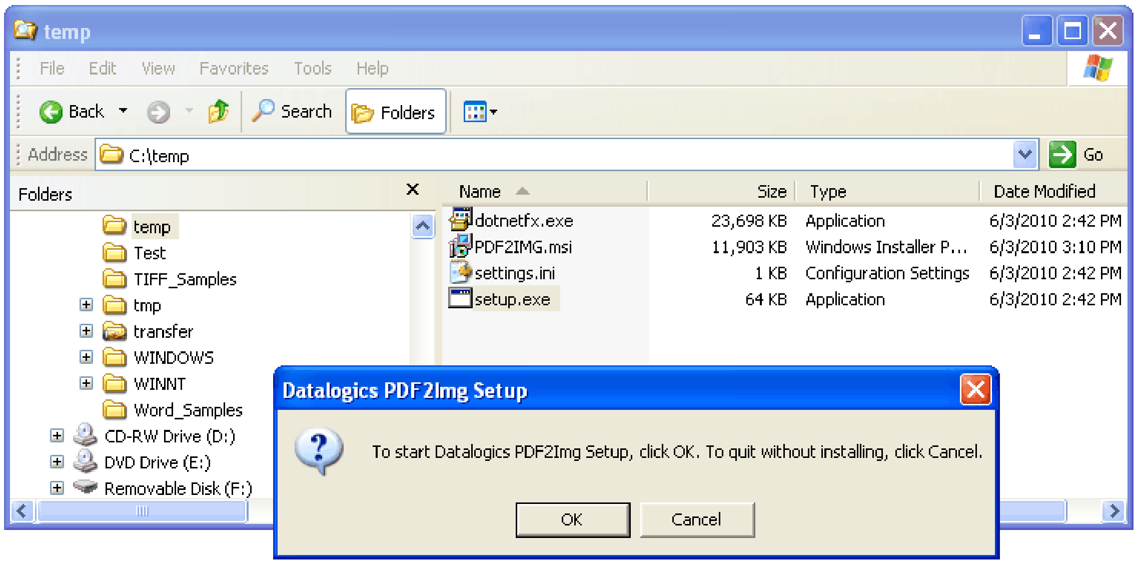
Extracting dotnetfx.exe

4 When complete, several files will have been unpacked, including the `setup.exe` installation program:



Name	Size	Type	Date Modified
dotnetfx.exe	23,698 KB	Application	6/3/2010 2:42 PM
PDF2IMG.msi	11,903 KB	Windows Installer P...	6/3/2010 3:10 PM
settings.ini	1 KB	Configuration Settings	6/3/2010 2:42 PM
setup.exe	64 KB	Application	6/3/2010 2:42 PM

5 At this point, Windows Vista and Windows 7 users should run the `PDF2IMG.msi` installation file directly. Windows XP users should continue on with these installation steps, to ensure that installation of the proper .NET frameworks will occur.
Doubleclick on the setup.exe to invoke it, and a Setup popup screen will appear:

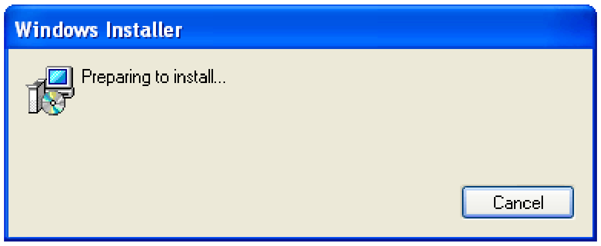


Name	Size	Type	Date Modified
dotnetfx.exe	23,698 KB	Application	6/3/2010 2:42 PM
PDF2IMG.msi	11,903 KB	Windows Installer P...	6/3/2010 3:10 PM
settings.ini	1 KB	Configuration Settings	6/3/2010 2:42 PM
setup.exe	64 KB	Application	6/3/2010 2:42 PM

Datalogics PDF2Img Setup

To start Datalogics PDF2Img Setup, click OK. To quit without installing, click Cancel.

6 Clicking the OK button will produce a popup message as the application loads:

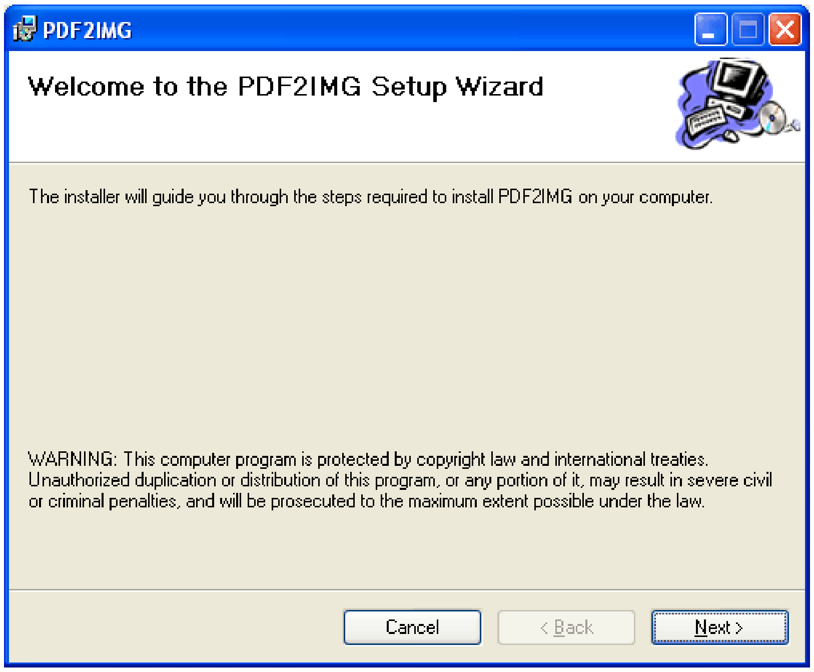


Windows Installer

Preparing to install...

At this point in the process, the installation process will check to see whether your machine has the required *Microsoft .NET Framework* installed. If not, it will branch off to install one (supplied with your release) before continuing with this process. If a *Microsoft .NET Framework* License Agreement screen appears at this point, skip to “Installing Microsoft .NET Framework” on page 2.14 (following). The procedure will continue automatically from this point once the *Microsoft .NET Framework* installation is complete.

7 A welcome screen will appear when the Setup Wizard starts:



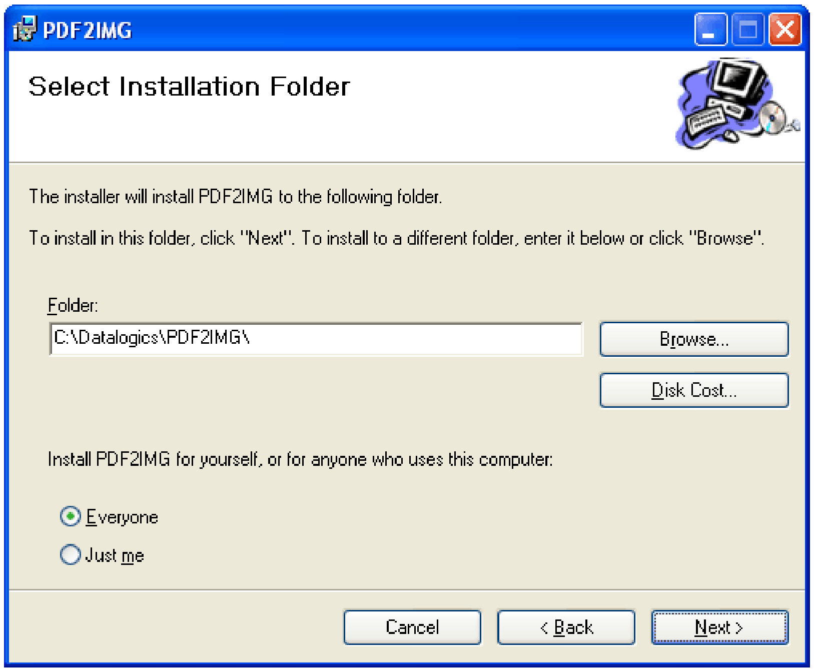
Welcome to the PDF2IMG Setup Wizard

The installer will guide you through the steps required to install PDF2IMG on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Click the Next button to proceed.

8 By default, *PDF2IMG* will install in its own folder under `C:\Datalogics\PDF2IMG`, but you are free to select a different location, either by typing it directly into the field or by clicking the button to select a different location:



Select Installation Folder

The installer will install PDF2IMG to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

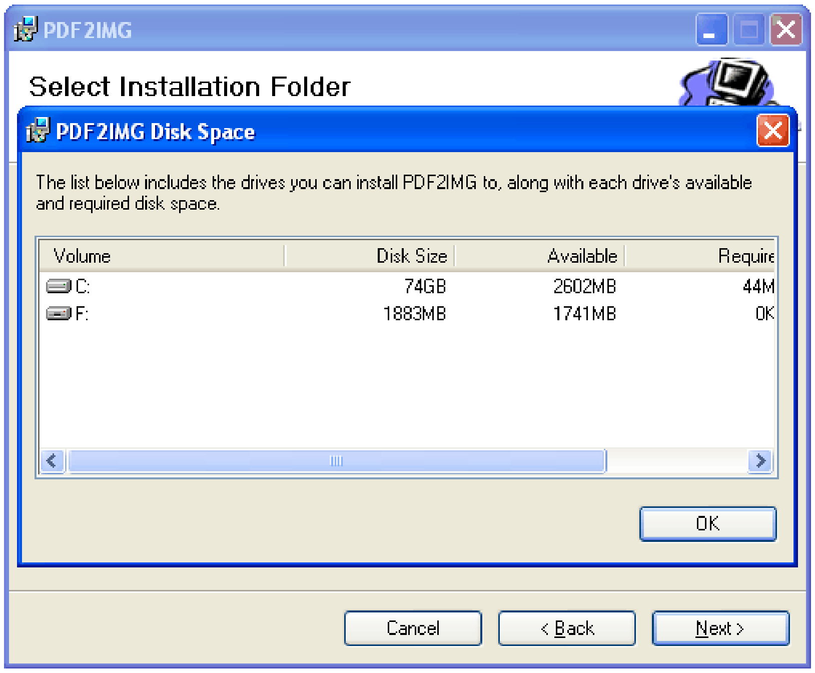
Folder: C:\Datalogics\PDF2IMG\

Install PDF2IMG for yourself, or for anyone who uses this computer:

[✓ Everyone]
Just me

By default, the Everyone radio button is set, to allow *PDF2IMG* use on this machine to all users. To limit its availability to you only, select the Just me option instead.

Finally, you can check available disk space on your machine before continuing, by clicking on the Disk Cost button first. This will bring up a listing of the available disk space on the drives and partitions of your machine:



PDF2IMG

Select Installation Folder

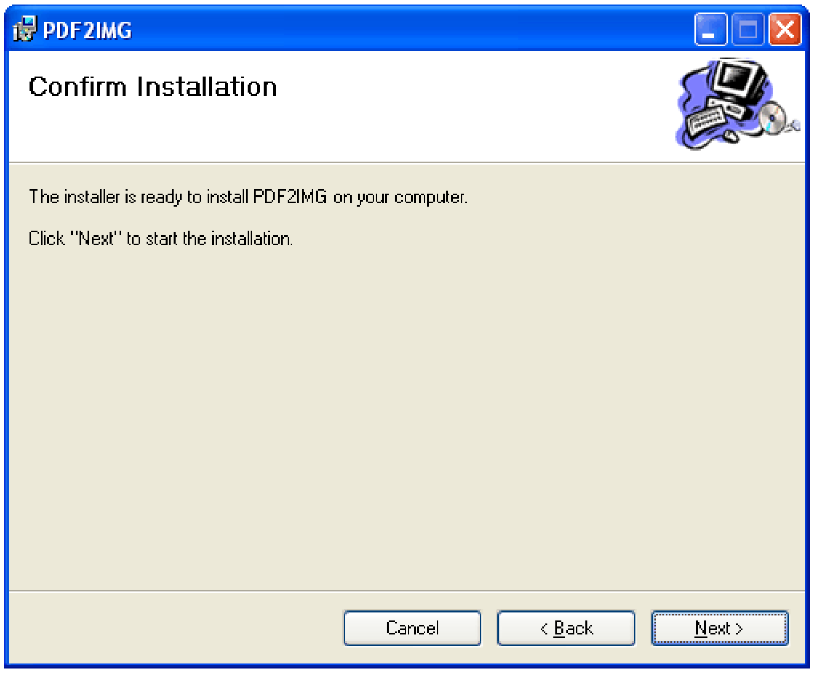
PDF2IMG Disk Space

The list below includes the drives you can install PDF2IMG to, along with each drive's available and required disk space.

Volume	Disk Size	Available	Require
C:	74GB	2602MB	44M
F:	1883MB	1741MB	0K

Click the OK button of this screen to return to the previous screen, and click the Next button there when you are ready to continue.

9 The confirmation screen will give you a chance to go back or cancel the installation process before file unpacking begins:



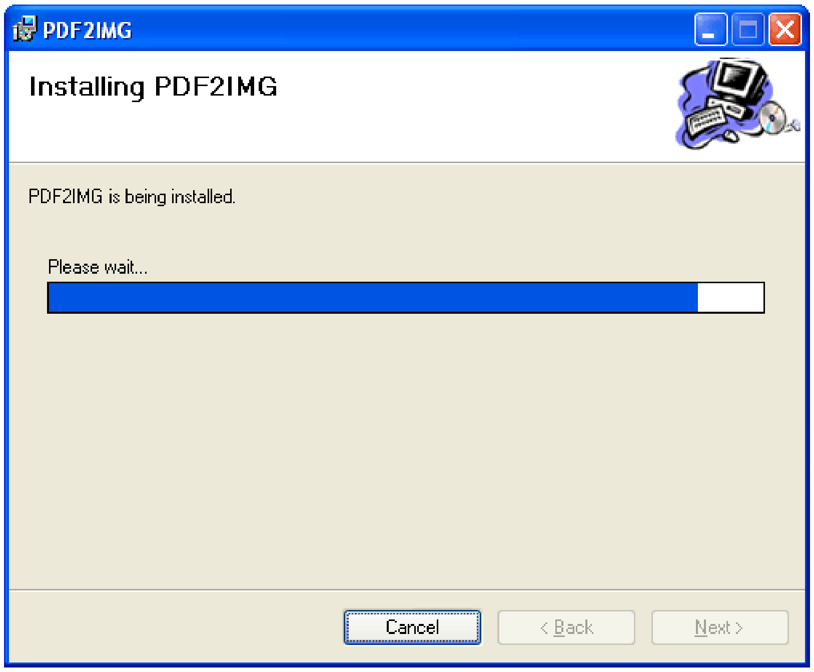
PDF2IMG

Confirm Installation

The installer is ready to install PDF2IMG on your computer.

Click "Next" to start the installation.

10 A bar chart will advance as file installation proceeds:

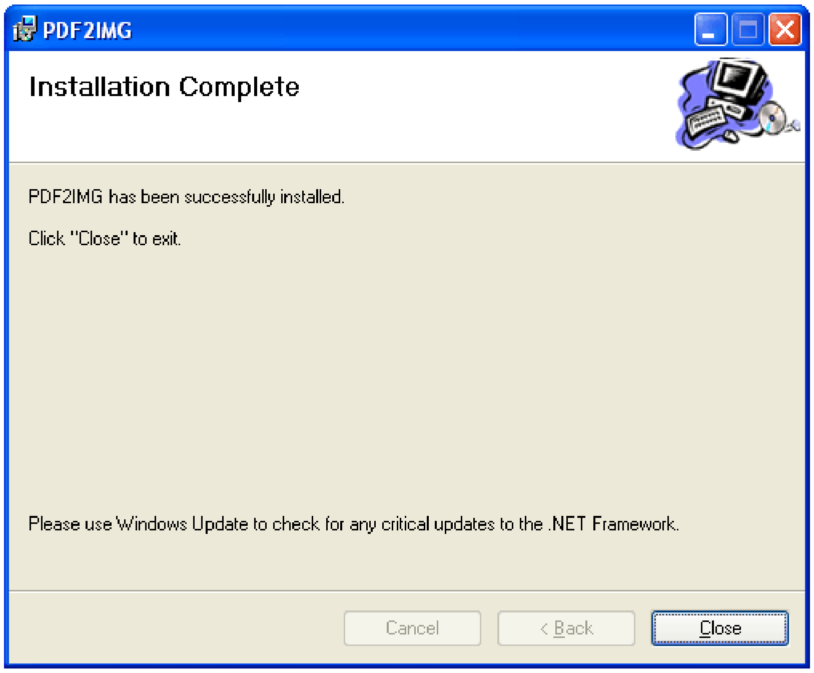


Installing PDF2IMG

PDF2IMG is being installed.

Please wait...

11 A completion screen will appear when the installation has finished successfully:



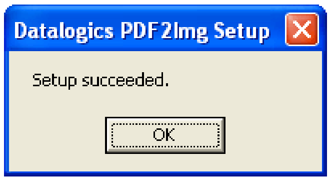
Installation Complete

PDF2IMG has been successfully installed.

Click "Close" to exit.

Please use Windows Update to check for any critical updates to the .NET Framework.

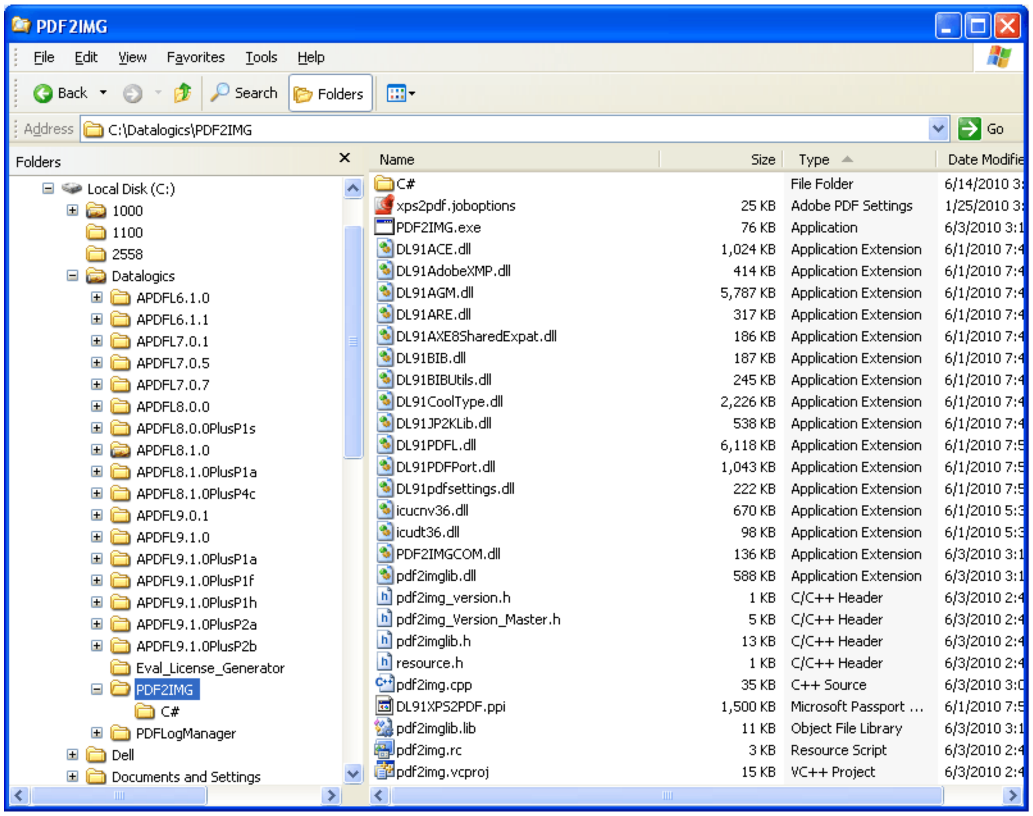
12 Click the OK button of the final popup screen to complete the installation procedure:



Datalogics PDF2Img Setup

Setup succeeded.

13 Viewing your unpacked location via *Windows Explorer* should now give you a listing of sample folders and supporting files for *PDF2IMG* and *PDF2IMGCOM*:



PDF2IMG
File Edit View Favorites Tools Help
Back Search Folders
Address C:\Datalogics\PDF2IMG Go
Folders

- Local Disk (C:)
  - 1000
  - 1100
  - 2558
  - Datalogics
    - APDFL6.1.0
    - APDFL6.1.1
    - APDFL7.0.1
    - APDFL7.0.5
    - APDFL7.0.7
    - APDFL8.0.0
    - APDFL8.0.0PlusP1s
    - APDFL8.1.0
    - APDFL8.1.0PlusP1a
    - APDFL8.1.0PlusP4c
    - APDFL9.0.1
    - APDFL9.1.0
    - APDFL9.1.0PlusP1a
    - APDFL9.1.0PlusP1f
    - APDFL9.1.0PlusP1h
    - APDFL9.1.0PlusP2a
    - APDFL9.1.0PlusP2b
    - Eval_License_Generator
    - PDF2IMG
      - C#
    - PDFLogManager
  - Dell
  - Documents and Settings

Name	Size	Type	Date Modifie
C#		File Folder	6/14/2010 3:
xps2pdf.joboptions	25 KB	Adobe PDF Settings	1/25/2010 3:
PDF2IMG.exe	76 KB	Application	6/3/2010 3:1
DL91ACE.dll	1,024 KB	Application Extension	6/1/2010 7:4
DL91AdobeXMP.dll	414 KB	Application Extension	6/1/2010 7:4
DL91AGM.dll	5,787 KB	Application Extension	6/1/2010 7:4
DL91ARE.dll	317 KB	Application Extension	6/1/2010 7:4
DL91AXE8SharedExpat.dll	186 KB	Application Extension	6/1/2010 7:4
DL91BIB.dll	187 KB	Application Extension	6/1/2010 7:4
DL91BIBUtils.dll	245 KB	Application Extension	6/1/2010 7:4
DL91CoolType.dll	2,226 KB	Application Extension	6/1/2010 7:4
DL91JP2KLib.dll	538 KB	Application Extension	6/1/2010 7:4
DL91PDFL.dll	6,118 KB	Application Extension	6/1/2010 7:5
DL91PDFPort.dll	1,043 KB	Application Extension	6/1/2010 7:5
DL91pdfsettings.dll	222 KB	Application Extension	6/1/2010 7:5
icucnv36.dll	670 KB	Application Extension	6/1/2010 5:3
icudt36.dll	98 KB	Application Extension	6/1/2010 5:3
PDF2IMGCOM.dll	136 KB	Application Extension	6/3/2010 3:1
pdf2imglib.dll	588 KB	Application Extension	6/3/2010 3:1
pdf2img_version.h	1 KB	C/C++ Header	6/3/2010 2:4
pdf2img_Version_Master.h	5 KB	C/C++ Header	6/3/2010 2:4
pdf2imglib.h	13 KB	C/C++ Header	6/3/2010 2:4
resource.h	1 KB	C/C++ Header	6/3/2010 2:4
pdf2img.cpp	35 KB	C++ Source	6/3/2010 3:0
DL91XPS2PDF.ppi	1,500 KB	Microsoft Passport ...	6/1/2010 7:5
pdf2imglib.lib	11 KB	Object File Library	6/3/2010 3:1
pdf2img.rc	3 KB	Resource Script	6/3/2010 2:4
pdf2img.vcproj	15 KB	VC++ Project	6/3/2010 2:4

You can run the `pdf2img.exe` application as it is provided, or use the enclosed *Microsoft Visual C++* Project files to do your own additional development and building work. See “Modifying the PDF2IMG Command Arguments” on page 4.2 for more details.

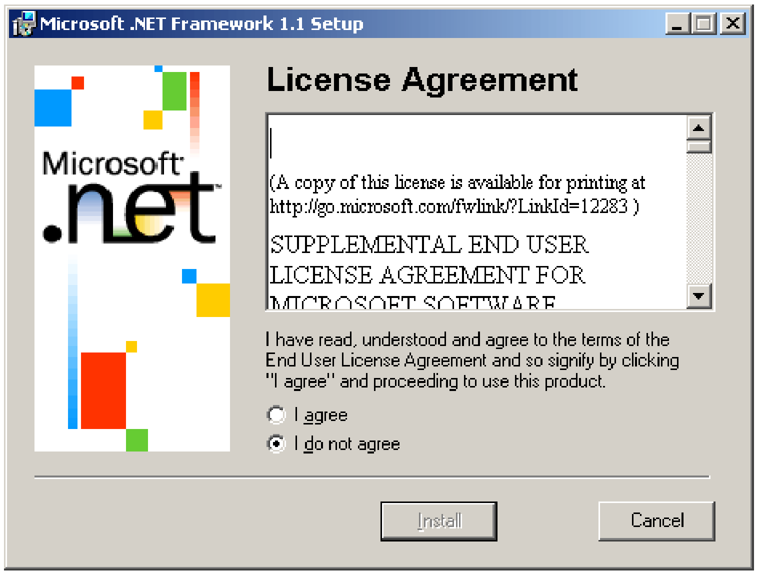
The *PDF2IMGCOM* modules will enable you to construct a COM object for similar work. See “The C# Sample Application” on page 3.3 for more details.

14 If your application will make use of other font resources besides those contained in the documents to be converted, be sure that you have defined your environment variables properly to account for all locations where your fonts and Cmap files reside. (See “Environment Variable Definitions” on page 4.29 for full details.) Alternatively, you can use the `-fontlist` command-line argument to specify resource location(s) at runtime. (See “Fontlist” on page 4.11 for full details.)

Installing *Microsoft .NET Framework*

Building of *PDF2IMG* or *PDF2IMGCOM* on Windows platforms requires *Microsoft .NET Framework* v1.1. A copy is provided with the Windows release, and the installation procedure will include the following steps if it detects that *Microsoft .NET Framework* is not already installed on your machine.

1 You must accept the License Agreement in order to proceed with the *Microsoft .NET Framework* installation. Click the "I agree" radio button and Install button to continue:


Microsoft .NET Framework 1.1 Setup

License Agreement

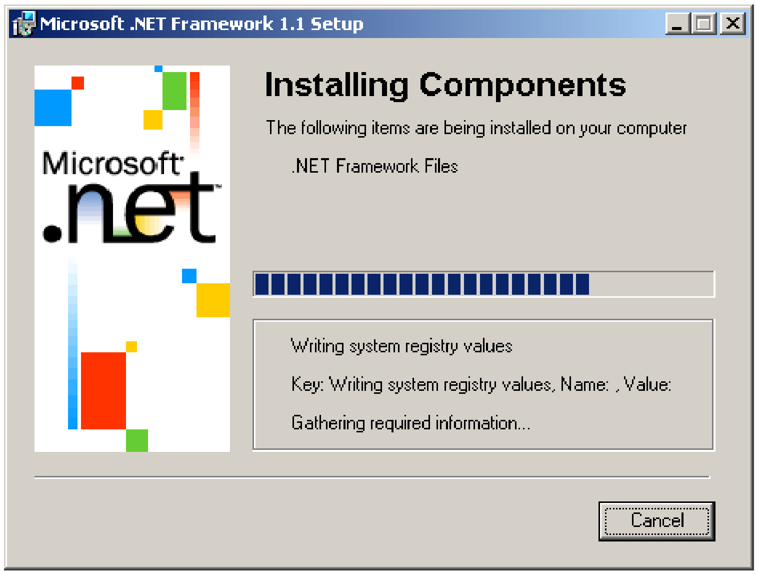
(A copy of this license is available for printing at http://go.microsoft.com/fwlink/?LinkId=12283)

SUPPLEMENTAL END USER LICENSE AGREEMENT FOR MICROSOFT SOFTWARE

I have read, understood and agree to the terms of the End User License Agreement and so signify by clicking "I agree" and proceeding to use this product.

( ) I agree
[✓ I do not agree]

2 The .NET installation procedure will collect some system information, and possibly warn you to exit certain running processes:


Microsoft .NET Framework 1.1 Setup

Installing Components

The following items are being installed on your computer

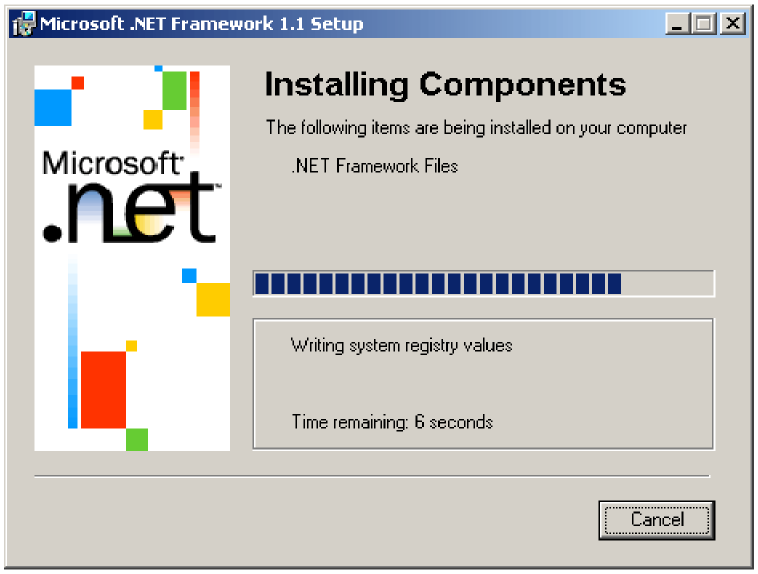
.NET Framework Files

Writing system registry values

Key: Writing system registry values, Name: , Value:

Gathering required information...

3 Eventually it will install its files and write registry values:



Microsoft .NET Framework 1.1 Setup

Installing Components

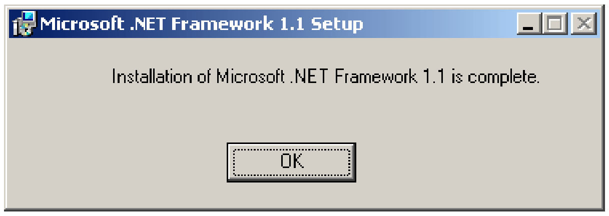
The following items are being installed on your computer

.NET Framework Files

Writing system registry values

Time remaining: 6 seconds

4 After you acknowledge this popup completion screen, the *PDF2IMG* installation will resume automatically.



Microsoft .NET Framework 1.1 Setup

Installation of Microsoft .NET Framework 1.1 is complete.

Building *PDF2IMGCOM*

This chapter outlines the process of building *PDF2IMGCOM*, including startup of the included sample application.

Installation

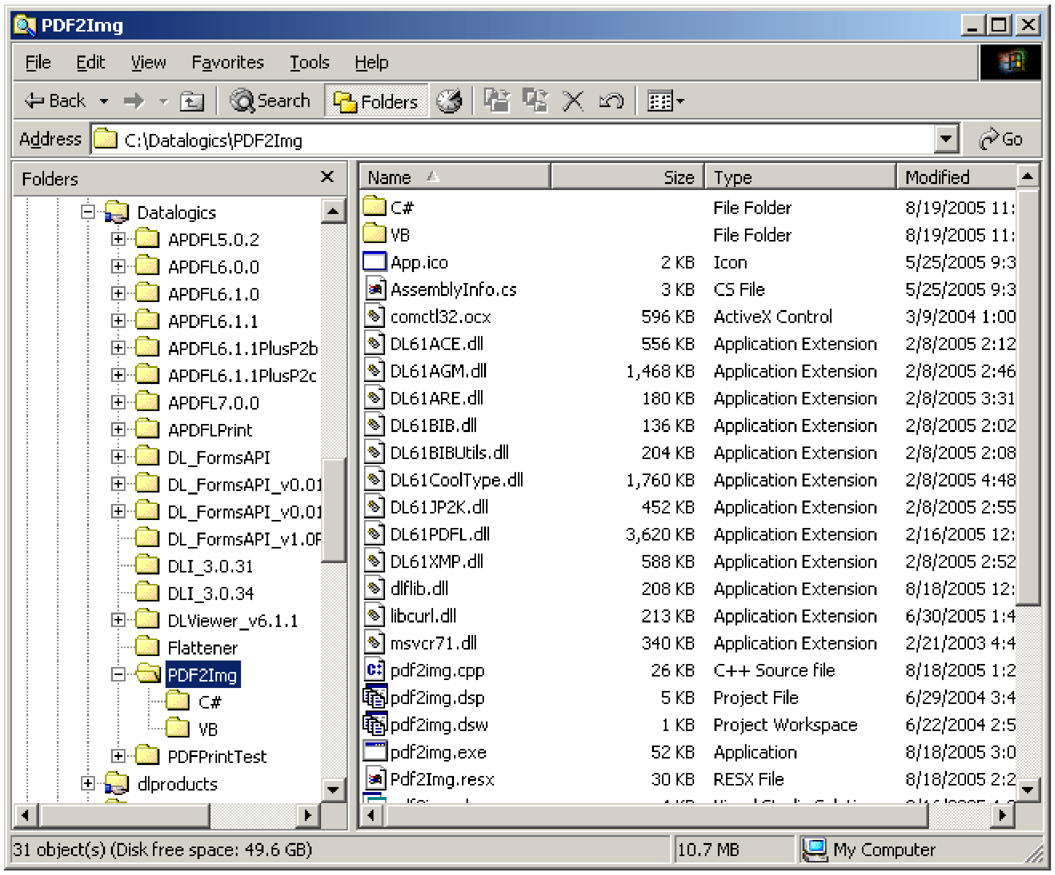
As of the v1.1P1p release of *PDF2IMG*, installation of *PDF2IMGCOM* is included with the same package as that of *PDF2IMG*; running the *PDF2IMG* installation procedure will (on Windows 32-bit platforms) deliver the *PDF2IMGCOM* components as well. When installation is complete, *PDF2IMGCOM* modules and sample folders will be found within the newly-unpacked collection of files.

Installation will create a subdirectory and sample area for developing applications in C#. A pre-built sample application showing a *PDF2IMGCOM* conversion interface in each environment is also provided. However, if desired you may alter or revise the *Graphical User Interface* (GUI) front end as you see fit (the source code is provided), or incorporate its functionality in your own application. (*PDF2IMG* is now distributed in `.DLL` or `.so` form, as appropriate for the platform. *PDF2IMGCOM* is currently available on Windows 32-bit platforms only.)

NOTE: *Microsoft .NET Framework* and *Visual Studio .NET* will be required for building C# *PDF2IMGCOM* applications.

The C# Sample Application

1 *PDF2IMGCOM* arrives with a sample application subfolder (below) holding a C# sample:



Address C:\Datalogics\PDF2Img

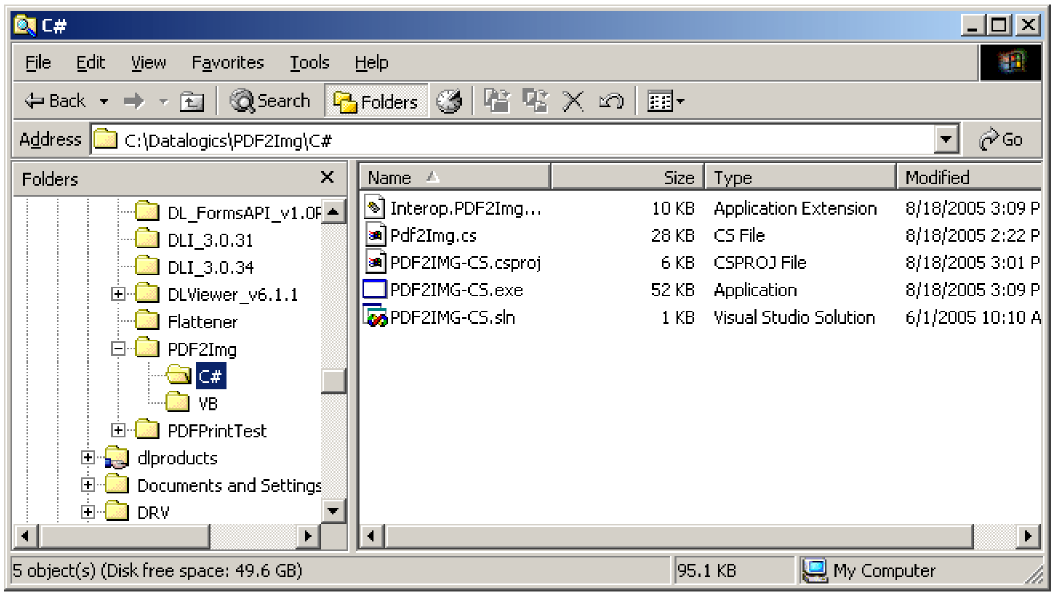
Folders

- Datalogics
  - APDFL5.0.2
  - APDFL6.0.0
  - APDFL6.1.0
  - APDFL6.1.1
  - APDFL6.1.1PlusP2b
  - APDFL6.1.1PlusP2c
  - APDFL7.0.0
  - APDFLPrint
  - DL_FormsAPI
  - DL_FormsAPI_v0.01
  - DL_FormsAPI_v0.01
  - DL_FormsAPI_v1.0F
  - DLI_3.0.31
  - DLI_3.0.34
  - DLViewer_v6.1.1
  - Flattener
  - PDF2Img
    - C#
    - VB
  - PDFPrintTest
- dlproducts

Name	Size	Type	Modified
C#		File Folder	8/19/2005 11:
VB		File Folder	8/19/2005 11:
App.ico	2 KB	Icon	5/25/2005 9:3
AssemblyInfo.cs	3 KB	CS File	5/25/2005 9:3
comctl32.ocx	596 KB	ActiveX Control	3/9/2004 1:00
DL61ACE.dll	556 KB	Application Extension	2/8/2005 2:12
DL61AGM.dll	1,468 KB	Application Extension	2/8/2005 2:46
DL61ARE.dll	180 KB	Application Extension	2/8/2005 3:31
DL61BIB.dll	136 KB	Application Extension	2/8/2005 2:02
DL61BIBUtils.dll	204 KB	Application Extension	2/8/2005 2:08
DL61CoolType.dll	1,760 KB	Application Extension	2/8/2005 4:48
DL61JP2K.dll	452 KB	Application Extension	2/8/2005 2:55
DL61PDFL.dll	3,620 KB	Application Extension	2/16/2005 12:
DL61XMP.dll	588 KB	Application Extension	2/8/2005 2:52
dlflib.dll	208 KB	Application Extension	8/18/2005 12:
libcurl.dll	213 KB	Application Extension	6/30/2005 1:4
msvcr71.dll	340 KB	Application Extension	2/21/2003 4:4
pdf2img.cpp	26 KB	C++ Source file	8/18/2005 1:2
pdf2img.dsp	5 KB	Project File	6/29/2004 3:4
pdf2img.dsw	1 KB	Project Workspace	6/22/2004 2:5
pdf2img.exe	52 KB	Application	8/18/2005 3:0
Pdf2Img.resx	30 KB	RESX File	8/18/2005 2:2

31 object(s) (Disk free space: 49.6 GB) 10.7 MB My Computer

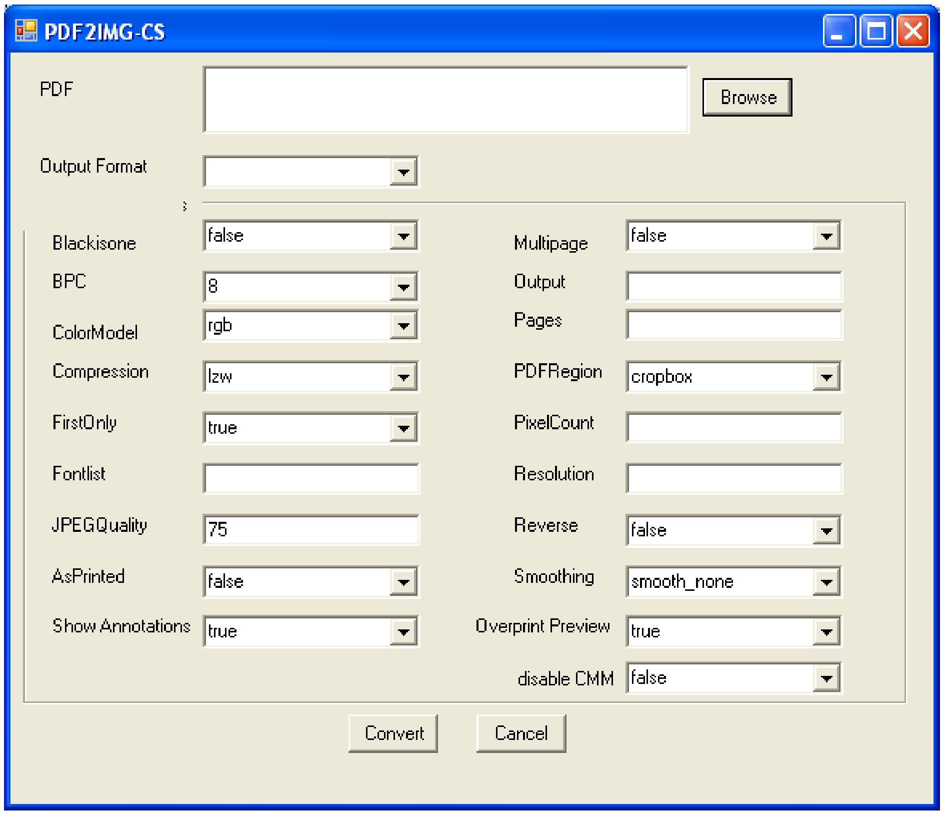
2 The C# subfolder (which installs in `C:\Datalogics\PDF2Img\C#` by default) contains the supporting source files and related modules for the sample application:



Name	Size	Type	Modified
Interop.PDF2Img...	10 KB	Application Extension	8/18/2005 3:09 P
Pdf2Img.cs	28 KB	CS File	8/18/2005 2:22 P
PDF2IMG-CS.csproj	6 KB	CSPROJ File	8/18/2005 3:01 P
PDF2IMG-CS.exe	52 KB	Application	8/18/2005 3:09 P
PDF2IMG-CS.sln	1 KB	Visual Studio Solution	6/1/2005 10:10 A

5 object(s) (Disk free space: 49.6 GB) 95.1 KB My Computer

3 Running that sample application will bring up this demonstration GUI utility:



PDF			
Output Format			
Blackisone	false	Multipage	false
BPC	8	Output	
ColorModel	rgb	Pages	
Compression	lzw	PDFRegion	cropbox
FirstOnly	true	PixelCount	
Fontlist		Resolution	
JPEGQuality	75	Reverse	false
AsPrinted	false	Smoothing	smooth_none
Show Annotations	true	Overprint Preview	true
		disable CMM	false

The supporting source code files for this utility are provided in the `C:\Datalogics\PDF2Img\C#` folder. (Screenshots shown are for illustration only and are subject to change.)

Running *PDF2IMG*

PDF2IMG (and by extension *PDF2IMGCOM*) is a Datalogics application designed to convert a selected PDF or XPS document file to one or more BMP, EPS, GIF, JPG/JPEG, PDF, PNG, RAW or TIFF image files.

Overview

PDF2IMG is a Datalogics application designed to convert a selected PDF or XPS document file to one or more BMP, EPS, GIF, JPG/JPEG, PDF, PNG, RAW or TIFF image files. (*PDF2IMGCOM* performs the same tasks via C# or VB GUI applications.) You also have several options for controlling Resolution, Color Model, Color Bit Depth and other settings, depending on output format selected. Although the interface for these commands appears different when using *PDF2IMGCOM* instead of *PDF2IMG*, their operation is the same in either case, except where noted otherwise below.

Basic Syntax

The basic command-line syntax for *PDF2IMG* is

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

Only the `<inputFile>` and `<outputForm>` arguments are required. Numerous optional arguments are available for tailoring the output as desired, explained on the following pages.

Modifying the *PDF2IMG* Command Arguments

This User Guide describes *PDF2IMG* as delivered. However, the source code for the command-line interpreter is also provided, so that you can tailor it and rebuild it as desired. (The command-line interpreter source code file is `pdf2img.cpp`.) While the following pages describe command-line argument names and default values in the product as shipped, you are free to modify those commands and their default behavior as you like, and build a new executable under your configuration, or embed the *PDF2IMG* engine itself within your own application.

You can run the `pdf2img.exe` application as provided, or use the enclosed *Microsoft Visual* C++ Project file (`pdf2img.vcproj`) to do your own additional development and building work.

NOTE: *PDF2IMG* and *PDF2IMGCOM* internals cannot be altered, although you can modify the command-line parameters and controls of *PDF2IMG*, since its command-line parsing wrapper `pdf2img.cpp` is provided in buildable form. `pdf2img.cpp` is intended to demonstrate how you can embed *PDF2IMG* within your own application, by replacing its default command-line environment with your own application calls.

Command Line Summary

Only the first two command-line arguments below are actually required; all others are optional and may be given in any order, but must precede <`inputFile`> and <`outputForm`>. Full details on each can be found in “Arguments and Options” on page 4.5 following, or click on the **boldface** argument below to jump directly to a full explanation.

inputF ile	Input PDF filename *(Required)*
outputForm	Output graphic format: `BMP`, `EPS`, `GIF`, `JPG`, `PDF`, `PNG`, `RAW` or `TIF` *(Required)*

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

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

Optional Arguments

asprinted	Reverse the Annotation handling, to suppress Image-only annotations and allow Print-only annotations *(Optional; No default)*
blackisone	Reverse the PhotometricInterpretation setting to be `black=1/white=0`. *(TIFF only; Optional; No default)*
bpc	Number of bits used to represent each output color channel *(Optional; Default* `8`*)*
colormodel	`rgb`, `rgba`, `cmyk` or `gray` *(Optional; Default* `rgb`*)*
compression	`no`, `jpg`, `lzw`, `g3` or `g4` *(TIFF only; Optional; Default* `lzw`*)*
digits	Specify the number of digits to use in the sequential output filename counter *(Optional; No default)*
firstonly	Convert only the first page of the input file *(Optional; Default* `All pages`*)*
fontlist	A quoted, semicolon-delimited list of alternate directories for font resources. *(Optional; No default)*
GetExtendedErrorString	Obtain an an internal *Adobe PDF Library* error string when the *PDF2IMGCOM* method returns a non-zero value *(PDF2IMGCOM only; Optional; No default)*
GetLastError	Obtain an error string when the *PDF2IMGCOM* method returns a non-zero

	value *(PDF2IMGCOM only; Optional; No default)*
help	Print the help list of available commands. *(PDF2IMG only; Optional; No default)*
ignorewarn	Suppress warnings for non-renderable content *(Optional; No default)*
jpegquality	JPEG compression quality, from `1` to `100`, where higher values produce a better image but also a larger output file size *(JPG only; Optional; Default* `75`*)*
maxbandmem	Maximum memory to use per band of multiband conversion output, in bytes *(JPG or TIFF only; Optional; Default* `300000000`*)*
multipage	Produce one, multipage TIFF output file of the requested name, rather than the default of single-page, sequentially-named output files *(TIFF only; Optional; No default)*
noannot	Suppress viewable annotations *(Optional; No default)*
nocmm	Suppress Color Management Module *(Optional; Default* `False`*)*
noenhancethinlines	Do not enhance thin lines when rendering *(Optional; No default)*
OPP	Enable Overprint Preview (OPP) in output *(Optional; No default)*
output	Prefix for output filename(s) to be created *(Optional; Default to Input PDF Filename plus sequence number)*
pages	Page(s) (or range of pages) to process; *e.g.* `2-4,7,9,14-last` *(Optional; No default)*
password	Password string required to open the document for conversion; *e.g.* `sesame` *(Optional; No default)*
pdfregion	Region of PDF page to rasterize; note the slight differences in option names between products (*PDFIMGCOM* options use a "`box`" suffix): *PDF2IMG* options: `art`, `bleed`, `bounding`, `crop`, `media` or `trim` *(Optional; Default* `crop`*)* *PDF2IMGCOM* options: `artbox`, `bleedbox`, `boundingbox`, `cropbox`, `mediabox` or `trimbox` *(Optional; Default* `cropbox`*)*
pixelcount	Absolute picture size, expressed as horizontal `x` vertical number of pixels *(Optional; No default)*
resolution	Output resolution, from 12 to 2400, in Dots per Inch *(Optional; Default* `300`*)*
reverse	Create a negative image *(Grayscale output formats only; Optional; No default)*
smoothing	Image antialiasing; note the slight differences in option names between products (*PDFIMGCOM* options use a "`smooth_`" prefix): *PDF2IMG* options: `none`, `text` or `all` *(Optional; Default* `none`*)* *PDF2IMGCOM* options: `smooth_none`, `smooth_text` or `smooth_all` *(Optional; Default* `smooth_none`*)*

Arguments and Options

While *PDF2IMG* can be run with as few as two command-line arguments (the input PDF file name and the graphic format to which you would like to convert it), you have numerous options and additional command-line arguments available to you.

NOTE: You can always enter the command `pdf2img -help` to generate a Usage message listing the command-line syntax.

inputFile

<`inputFile`> 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. *(Required)*

NOTE: Output file(s) will be placed in the same location as your input file, unless you specify otherwise via the -`output` option (See "Redirecting Output to Another Location" on page 4.16 following). Any existing output file(s) of the same name(s) will be overwritten by the new one(s).

outputForm

<`outputForm`> is the output graphic format you request: `BMP`, `EPS`, `GIF`, `JPG`, `PDF`, `PNG`, `RAW` or `TIF`. *(Required)*

Page-Split Functionality for PDF Output

As of the *PDF2IMG* v2.2P1j release, specifying `PDF` output will invoke the new page-split functionality, which accepts a multi-page input PDF document and generates one or more output single-page PDF documents.

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 width.

Asprinted

-`asprinted` *(No default)*

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 on-screen viewing are included, but -`asprinted` allows you to override that and reverse the distinction, so that you can render printable annotations (*i.e.* those annotations which have been flagged as printable by the document author) when converting the document to an image.

Furthermore, the use of `-asprinted` will suppress annotations flagged as *non*-printing by the document author (*i.e.* those intended for on-screen viewing only). Those will no longer appear in output images if this argument is selected. *(Optional; No default)*

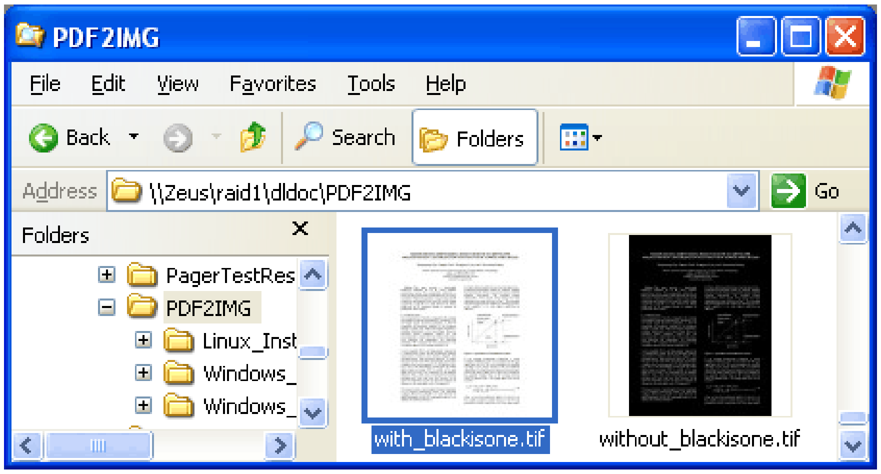
NOTE: This command can be overridden by the `-noannot` flag.

Blackisone

`-blackisone` *(TIFF only; No default)*

`-blackisone`, if present, will direct *PDF2IMG* to declare a reversed `PhotometricInterpretation` value of `black=1;white=0` in the header of an output 1-bit (B/W) TIFF image file. This corrects a problem existing in some third-party PDF documents, in which the converted TIFF output may unexpectedly display in reversed (negative) format when viewed via certain display utilities.

For example, in the *Windows Explorer* screenshot below, the white-on-black thumbnail image at right shows one such problem document which was converted without the `blackisone` flag. Its corrected equivalent is on the left:



In most circumstances you will not need this switch, and it will have no visible effect on systems or utilities which 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, not this one. See "Reverse" on page 4.21 for full details.

(TIFF only; Optional; No default)

BPC

`-bpc=[1 | 8]` *(Default* `8`*)*

```
-bpc=1
-bpc=8
```

`-bpc` (formerly `-bps` in early releases of *PDF2IMG*) is the Image Depth expressed as Bits per Color channel, the number of bits used to represent a color channel sample in the selected output format, typically `8` for color or grayscale images, `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 (*e.g.* Black and white), since the single bit can only represent `0` or `1`, the presence or absence of that color. `8` indicates that 8 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, `-colormodel=gray` and `-bpc=1` are also required and must be specified.

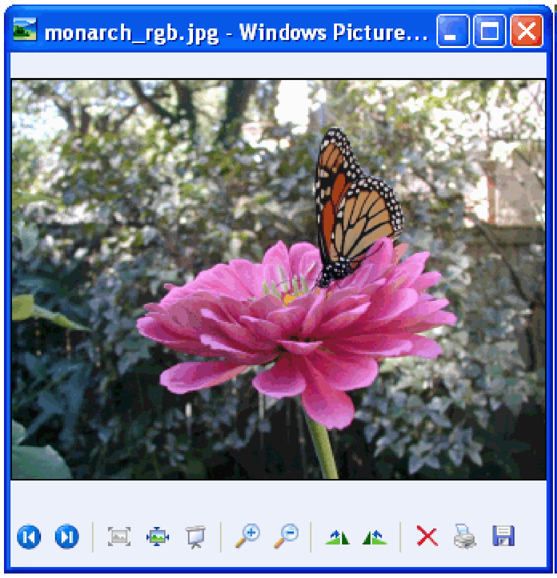
In Color images, each pixel is made up of three color channels (`RGB` — Red, Green and Blue respectively) or four color channels (`CMYK` — Cyan, Magenta, Yellow and Black respectively). Bit depth for each of these is always 8, and indicates that there may be 256 "shades" of each color. *(Optional; Default* `8`*)*

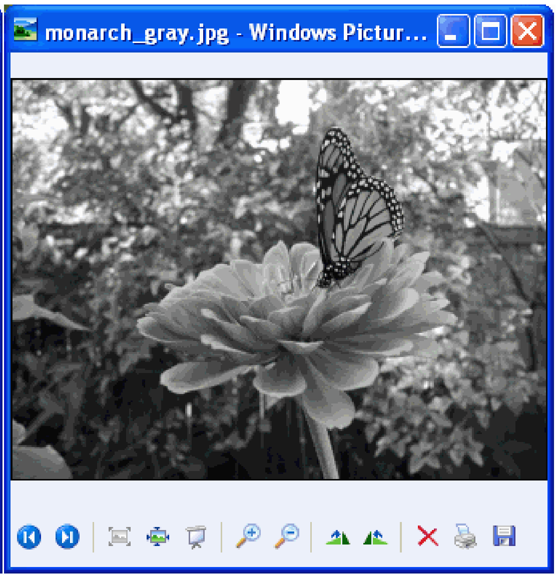
ColorModel

-colormodel=[gray | cmyk | rgb | rgba] *(Default* rgb*)*

-colormodel=gray

-colormodel is one of rgb, rgba, cmyk or gray, depending on output format selected. The valid colormodel choices are determined by the output format selected. *PDF2IMG* will return an error message if the requested colormodel is invalid for the selected format. *(Optional; Default* RGB*)*





NOTE: When producing TIFF g3 or g4 compressed output, -colormodel=gray and -bpc=1 are also required and must be specified.

OutputFormat	ColorModels Available
BMP	GRAY or RGB
GIF	GRAY or RGB
JPG	CMYK, GRAY or RGB
PNG	GRAY, RGB or RGBa
TIFF	CMYK, GRAY, RGB or RGBa

NOTE: *Adobe Photoshop* reads and writes CMYK JPEG files in a slightly non-standard format. As a result, CMYK JPEG files generated by *PDF2IMG* may appear somewhat discolored when viewed in *Photoshop*.

Compression

-compression=[no | jpg | lzw | g3 | g4] *(Default* `lzw`*)*

```
-compression=no
-compression=jpg
-compression=lzw
-compression=g3  (also requires colormodel=gray and bpc=1)
-compression=g4  (also requires colormodel=gray and bpc=1)
```

`-compression` is used to specify output compression of TIFF images as desired. Valid values depend on the type of TIFF images being processed:

- For color images, `jpg`, `lzw` or `no` may be selected, to turn compression on or off as desired
- For black and white images (1 channel, 1 bit), `g3` or `g4` compression can also be specified.

NOTE: When producing TIFF g3 or g4 compressed output, `-colormodel=gray` and `-bpc=1` are also required and must be specified.

(TIFF only; Optional; Default `lzw`*)*

Digits

`-digits=[0 to 9]` *(No default)*

The `-digits` command-line argument allows you to specify the number of digits to be used for the sequential output filename numbering suffix. Normally, leading digits are only added as required in order to maintain file sorting order (*e.g.* a 200-page input file will be processed to output as `FILE_001`, `FILE_002` and so on), but the `-digits` argument may be used to force a specific number of digits regardless of the input page count; *e.g.* specifying `-digits=4` will generate output filenames such as `FILE0001`, `FILE0002`, etc.

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

Entering a `-digits` value of `0` (zero) will revert to normal sequential numbering and filenaming logic. Entering a `-digits` value of `1` (one) will suppress output of any leading zeroes, as well as the preceding Underscore character.

CAUTION: 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. No error will occur, but later filenames in the output sequence may be one or more digits longer than earlier filenames as a result (*e.g.* `FILE98`, `FILE99`, `FILE100` instead of `FILE098`, `FILE099`, `FILE100`), possibly leading to file-ordering problems later on.

(Optional; No default)

FirstOnly

`-firstonly` *(No default)*

`-firstonly`, if present, will direct *PDF2IMG* to convert only the first page of the input file rather than all of them. This option does not accept a value. *(Optional; No default)*

Fontlist

`-fontlist="directory1;directory2;directoryN"`*(No default)*

```
-fontlist="C:\Test\ClientFonts"
-fontlist="C:\AlternateFonts\Test;C:\Application\Resources"
```

`-fontlist`, if present, will pass to *PDF2IMG* an alternate list of directories where font resources can be found. *PDF2IMG* will use the first instance of each font that it locates. The following usage rules apply:

- Values (*i.e.* directories) within the list for this argument must be separated by semicolons.
- Wildcard characters such as tildes (~) or asterisks (*) are not allowed.
- The value list itself must be enclosed within a pair of Double Quotes.
- A value list given here will replace the default search of Adobe font locations. *(see details below)*

You generally do not need the `-fontlist` option unless you are using font files not actually installed on your machine (*i.e.* where the font files are present on your machine, but are not recognized or installed as fonts by the operating system). In addition, the `-fontlist` option will replace *PDF2IMG*'s standard search for fonts installed by Adobe products. For example, if the `-fontlist` option is *not* given, *PDF2IMG* on Windows will search these resource areas that an *Adobe Acrobat* v8 installation creates:

```
C:\Program Files\Adobe\Acrobat 8.0\Resource\Font
C:\Program Files\Adobe\Acrobat 8.0\Resource\Cmap
C:\Program Files\Adobe\Acrobat 8.0\Resource\CIDFont
```

Regardless of whether the `-fontlist` option is given, both Windows and UNIX versions always search the following locations, relative to the present working directory:

- `Resource`
- `Resource/Cmap`
- `Resource/Font`

Also see “Environment Variable Definitions” on page 4.29.

As of *PDF2IMG* v1.2, you can list up to 16 locations via the `-fontlist` option.

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 platforms, or `X-Windows` and `OpenWindows` font locations on Solaris platforms are always searched. The -`fontlist` argument only overrides the Adobe location search.

(Optional; No default)

GetExtendedErrorString

`GetExtendedErrorString` *(PDF2IMGCOM only; No default)*

The `GetExtendedErrorString` call is available for *PDF2IMGCOM* applications only, and can be used to obtain an *Adobe PDF Library* internal error string when the *PDF2IMG* method in use returned a non-zero value. This returns the text string generated by the last error condition encountered by the *Adobe PDF Library*, and may provide more detail than that returned by *PDF2IMG*.

Not all *PDF2IMGCOM* errors originate at the library level, but those that do may have a more explanatory string returned via this call. (The `PDF2IMG.exe` sample application has also been modified to call this function upon error via the `pdf2img_get_extended_error()` call; see its sample source code file provided in your *PDF2IMG* release for more details.)

Any error message stored here will be cleared by a subsequent successful *PDF2IMGCOM* operation. If no error message is present, the string `"No error."` (exactly as shown) will be returned. *(PDF2IMGCOM only; Optional; No default)*

GetLastError

`GetLastError` *(PDF2IMGCOM only; No default)*

`-GetLastError` is available for *PDF2IMGCOM* applications only, and can be used to obtain an error string when the *PDF2IMG* method in use returned a non-zero value.

Any error message stored here will be cleared by a subsequent successful *PDF2IMGCOM* operation. If no error message is present, the string `"No error."` (exactly as shown) will be returned. *(PDF2IMGCOM only; Optional; No default)*

help

`-help` *(No default)*

`-help`, if present, will generate a Usage listing to the screen, giving basic information on the proper syntax and values (if applicable) for all *PDF2IMG* command-line arguments as shipped. This option does not accept a value. *(Optional; No default)*

ignorewarn

`-ignorewarn` *(No default)*

`-ignorewarn`, if present, will suppress the warning normally returned whenever nonrenderable content is encountered in the input file. In *PDF2IMGCOM* usage, this is the condition flagged by the `pdf2img_check_for_missing_appearances` call for conditions such as nonrenderable annotations or form fields. This option does not accept a value. *(Optional; No default)*

JPEGQuality

`-jpegquality=[1 to 100]` *(Default* `75`*)*

```
-jpegquality=50
-jpegquality=100
```

`-jpegquality` is a value from 1 to 100, representing the quality/size value for the JPEG compressor. A higher value will produce a better quality image, though also a larger output file as a result. Lower values will produce lower-quality images but more efficient, smaller output file sizes.

NOTE: Lowering the JPG 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.

(Optional; Default `75`*)*

Maxbandmem

`-maxbandmem=[100000000 to 4200000000]`
(JPG or TIFF only; Default `300000000`*; See Caution below)*

When generating JPG 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) in order to use smaller chunks of memory, then reassembles the bitmaps into the finished output image. This banding approach (if needed for the input) 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. `-maxbandmem` is a feature of the enhanced conversion process for converting to JPG or TIF, allowing you to fine-tune that process as needed, if you find that your application's performance is enhanced by making the size of the rendering bands either larger or smaller as desired (within the technical limits of your machine).

CAUTION: 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 sample. Higher values are allowed if passed in through the `pdf2img_set_max_band_memory` API.

(JPG or TIFF only; Optional; Default `300,000,000` *bytes)*

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` will direct *PDF2IMG* to produce one, multipage TIFF output file instead. *(TIFF only; Optional; No default)*

Noannot

`-noannot` *(No default)*

Specify `-noannot` on a *PDF2IMG* command line to prevent displayable annotations from appearing in output.

For similar action in *PDF2IMGCOM* DLL applications, see the `pdf2img_setprintannot` call (with a value of `0`).

CAUTION: This command should be used 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, and no others.

NOTE: This command will override the `-asprinted` flag.

NOCMM

`-nocmm=[true | false]` *(Default* `False`*)*

```
-nocmm=true
-nocmm=false
```

`-nocmm` controls the use or non-use of the Color Management Module (*CMM*) and embedded color profiles during conversion to selected output graphic image formats.

For *PDF2IMG* v1.6 or higher, color management is normally in effect. This means that *PDF2IMG* assumes an output profile of *Adobe Acrobat* 9 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* 9 CMYK, Adobe 1998 RGB, or Gamma 2.2 respectively. When generating TIF, JPG, PNG or BMP output, *PDF2IMG* will embed the corresponding profile in the output image.

When `-nocmm` is specified as `true`, *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` *(No default)*

PDF2IMG makes use of the "Enhance thin lines" rendering option (as seen in *Adobe Reader* or *Adobe Acrobat*) by default. If this is not the effect you want (*e.g.* you need to maintain the legacy output appearance of *PDF2IMG* prior to v2.1P1b, when this option was adopted as the default behavior), this command-line switch will turn it off.

For *PDF2IMGCOM* applications, the `pdf2img_set_enhance_thin_lines` call may also be used.

OPP

`-OPP` *(No default)*

Prior to the development of this flag, Overprint content would be lost during the rendering process. Specifying this flag will now generate a graphic that represents what the input PDF page would look like after being printed, accounting for ink overprinting. New enhancements have been added to support Overprint rendering, including in the C library (via the `pdf2img_set_OPP` API call), this `-OPP` command-line option, the OPP COM library interface parameter, and a C# example update. All settings are False by default. *(Optional; No default)*

Output

`-output=[filename]` *(Default input filename plus sequence number)*

```
-output=alternate_name
-output="C:\Converted_Files\alternate_name"
```

`-output` is the prefix for the output filename(s) to be created. The value given here will be used for the output filename, with a sequence number and an appropriate extension appended to indicate the output file type (*e.g.* `sample_1.gif`, `sample_1.jpg`, etc.). For a multi-page input PDF document, sequential, separate output files will be created for each page of input, with a sequence number appended to each. (*e.g.* `sample_1.gif`, `sample_2.gif`, etc.; see "Multi-Page Processing" on page 4.25 for more details.)

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

If the `-output` option is not given, the input filename is used (with sequence number appended), along with the file type extension determined by the output format selected. *(Optional; Default input filename plus sequence number)*

Redirecting Output to Another Location

You can use the `-output` option to redirect output file(s) to another location, but the following notes apply:

- Any existing output file(s) of the same name will be overwritten by the new one(s).
- The alternate file folder location must already exist; *PDF2IMG* will not create it.
- An output filename must be given along with the folder name; the input filename is not used by default when redirecting output.

Pages

`-pages=[range]` *(No default)*

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

`-pages` can 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. (*e.g.* 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.) Page ranges must be given in increasing order (*i.e.* starting page number must be lower than ending page number), and no spaces are allowed within the `pages` value.

Processing to End of File

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. *(Optional; No default)*

Password

-password=[string] *(No default)*

```
-password=sesame
```

-password will pass in the User or Owner password value 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 security restrictions can be processed by providing the User password here. 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.

PDFRegion

PDF2IMG:

-pdfregion=[art | bleed | bounding | crop | media | trim] *(Default* crop*)*

```
-pdfregion=art
-pdfregion=bleed
-pdfregion=bounding
-pdfregion=media
-pdfregion=trim
```

PDF2IMGCOM:

-pdfregion=[artbox | bleedbox | boundingbox | cropbox | mediabox | trimbox] *(Default* cropbox*)*

```
-pdfregion=artbox
-pdfregion=bleedbox
-pdfregion=boundingbox
-pdfregion=mediabox
-pdfregion=trimbox
```

-pdfregion enables you to select a region of the input page(s) to rasterize. Elements not within the indicated area are ignored and do not appear in output. pdfregion values correspond to page boundary definitions as given in Section 10.10.1 of the *PDF Reference*, version 1.5, "Page Boundaries," summarized below:

`PDFRegion` **(*PDF2IMG*)**	`PDFRegion` **(*PDF2IMGCOM*)**	**Coverage Area**
`art`	`artbox`	Extent of the page's meaningful content, as intended by the page's creator
`bleed`	`bleedbox`	Region to which the page contents should be clipped when output in a production environment
`bounding`	`boundingbox`	All page area within the declared BoundingBox dimensions
`crop`	`cropbox`	Region to which the page contents should be clipped when displayed or printed *(Default)*
`media`	`mediabox`	Boundaries of the physical medium on which the page is to be printed
`trim`	`trimbox`	Dimensions of the finished page after trimming

(Optional; Default `crop` *via PDF2IMG*; `cropbox` *via PDF2IMGCOM)*

PDF2IMGCOM Enumeration Values

When building an application with *PDF2IMGCOM* modules, the following enumeration values should be assumed:

```
typedef enum PDFregionEnum {
                     [helpstring("ARTBOX")] ARTBOX =1,
                     [helpstring("BLEEDBOX")] BLEEDBOX = 2,
                     [helpstring("BOUNDBOX")] BOUNDINGBOX = 3,
                     [helpstring("CROPBOX")] CROPBOX = 4,
                     [helpstring("MEDIABOX")] MEDIABOX = 5,
                     [helpstring("TRIMBOX")] TRIMBOX = 6
                     }ENUM_PDFREGION;
```

PixelCount

-pixelcount=[width x height] *(No default)*

-pixelcount=w:[width] *(No default)*

-pixelcount=h:[height] *(No default)*

```
-pixelcount=2550x3300
-pixelcount=660X300
-pixelcount=w:200
-pixelcount=h:300
```

-`pixelcount` enables you to specify the exact dimensions of your output width and/or height in pixels. As of *PDF2IMG* v1.2.8, both `width` and `height` values are optional, though if you provide both, you must give the `width` first, `height` second, separated by a case-insensitive "`x`." No spaces are allowed.

NOTE: The spaces in the boldface argument template above are for legibility only. Actual command-line arguments giving two or more values, such as the samples above, must not contain spaces between the values.

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 do not want to alter the proportions or aspect ratio of your original input page(s), 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 width.

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. *(Optional; No default)*

Resolution

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

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

```
-resolution=600
-resolution=600x300
```

`-resolution` is the output Dots per Inch value, from 12 to 2400. If one value is given, it will be applied to both dimensions; if two values are given, the first will be applied horizontally and the second will be applied vertically.

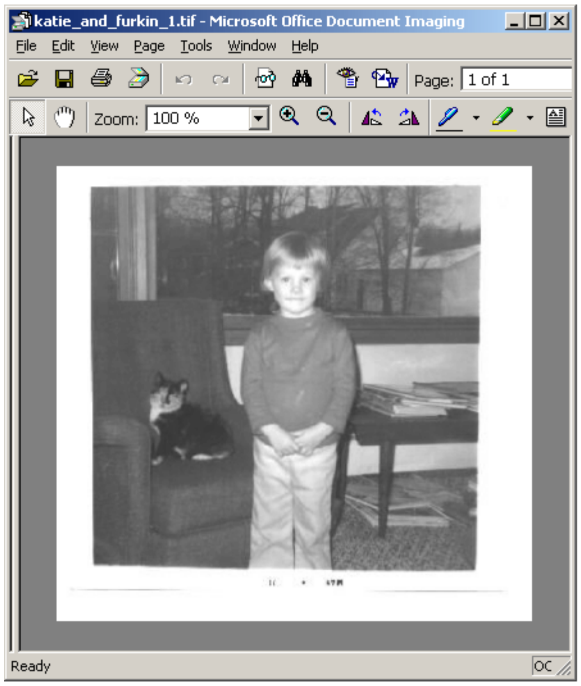
NOTE: The spaces in the boldface argument template above are for legibility only. Actual command-line arguments giving two or more values, such as the samples above, must not contain spaces between the values.

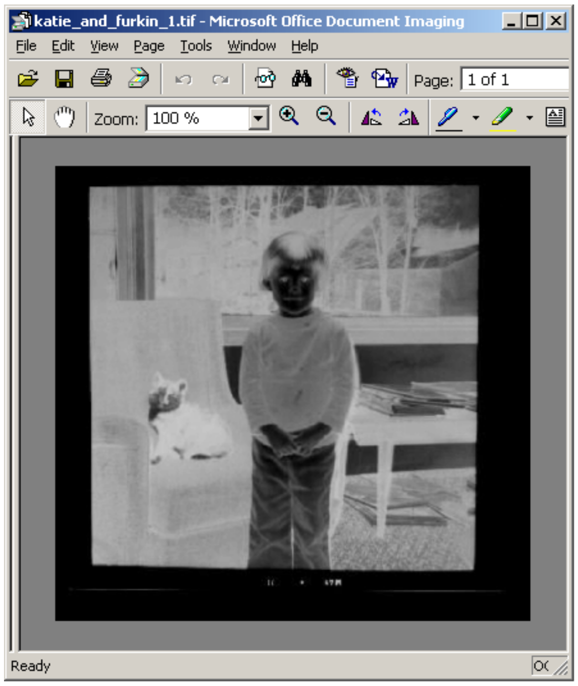
For best results this should be specified as a multiple of the DPI resolution of the intended output device, since this will minimize or eliminate the need for any interpolation of values if you match your image file resolution to the device that will display it. Also see “PixelCount” on page 4.19 for another way to control the specific output size of your image. *(Optional; Default* `300` *on both dimensions)*

Reverse

-reverse *(Grayscale only; No default)*

If performing a conversion to Grayscale output, -reverse will direct *PDF2IMG* to reverse the Grayscale values to produce a "negative" image:


katie_and_furkin_1.tif - Microsoft Office Document Imaging
File Edit View Page Tools Window Help
Page: 1 of 1
Zoom: 100 %
Ready


katie_and_furkin_1.tif - Microsoft Office Document Imaging
File Edit View Page Tools Window Help
Page: 1 of 1
Zoom: 100 %
Ready

NOTE: This switch 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 switch, not this one. See "Blackisone" on page 4.6 for full details.

(Grayscale only; Optional; No default)

Smoothing

PDF2IMG:

-smoothing=[none | text | all] *(Default* none*)*

```
-smoothing=none
-smoothing=text
-smoothing=all
```

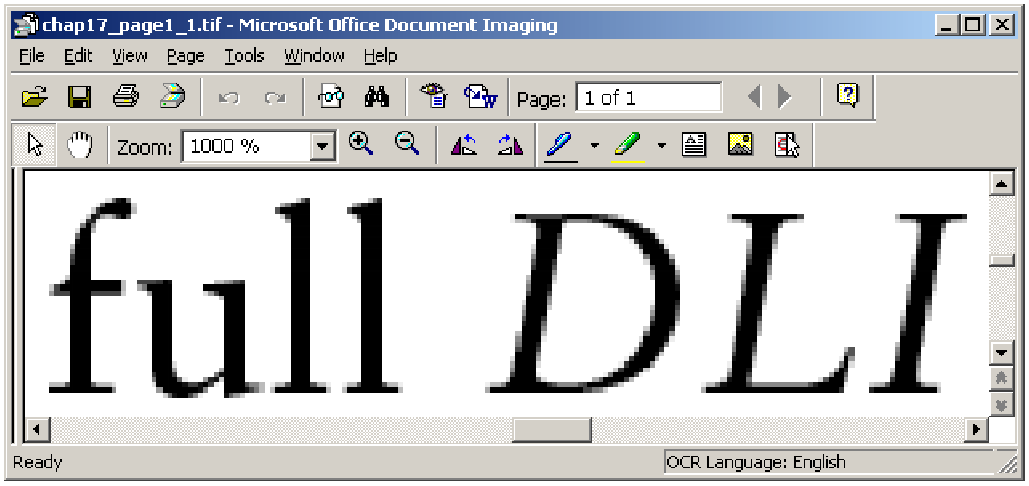
PDF2IMGCOM:

-smoothing=[smooth_none | smooth_text | smooth_all] *(Default* smooth_none*)*

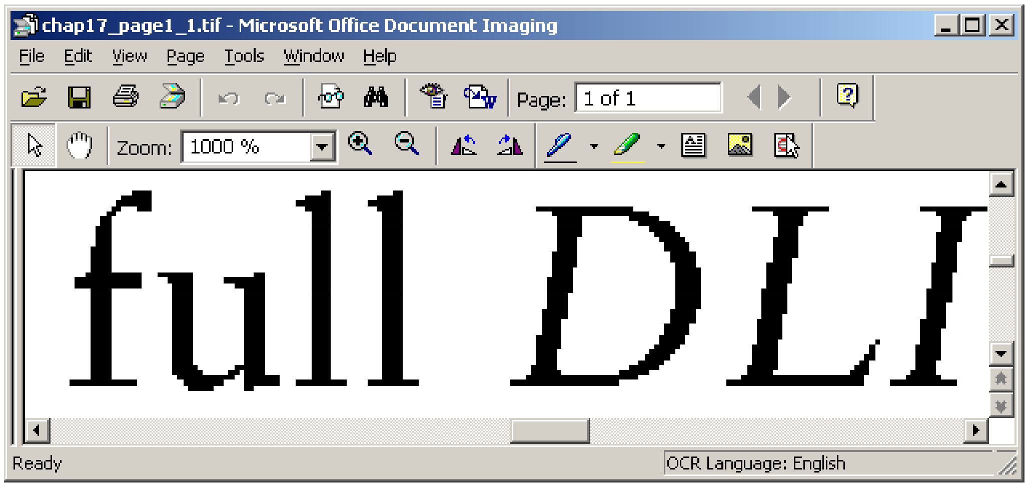
```
-smoothing=smooth_none
-smoothing=smooth_text
-smoothing=smooth_all
```

User-controlled antialiasing, or "smoothing," is now available in *PDF2IMG* v1.1P1k and later releases. The `-smoothing` flag is optional and accepts one of "`none`," "`text`" or "`all`" values via *PDF2IMG*, or "`smooth_none`," "`smooth_text`" or "`smooth_all`" via *PDF2IMGCOM*:

Text Smoothing (1000% Enlargement)


chap17_page1_1.tif - Microsoft Office Document Imaging
File Edit View Page Tools Window Help
Page: 1 of 1
Zoom: 1000 %
full DLI
Ready
OCR Language: English

No Smoothing (1000% Enlargement)


chap17_page1_1.tif - Microsoft Office Document Imaging
File Edit View Page Tools Window Help
Page: 1 of 1
Zoom: 1000 %
full DLI
Ready
OCR Language: English

Smoothing is most helpful when creating low-resolution outputs, but is not advised when creating image files for eventual paper/print output. It is also not recommended for black-and-white (1bpp) output files, in order to preserve sharpness at high magnifications

NOTE: With the release of *PDF2IMG* v1.1P1k, the default behavior was changed from smoothing to *no* smoothing. If you are upgrading from a previous release of *PDF2IMG*, you should verify that converted output from the v1.1P1k release (or newer) is still acceptable using your current settings.

(Optional; Default `none` *via PDF2IMG;* `smooth_none` *via PDF2IMGCOM)*

PDF2IMGCOM Enumeration Values

When building an application with *PDF2IMGCOM* modules, the following enumeration values should be assumed:

```
typedef enum smoothingEnum {
                     [helpstring("SMOOTH_ALL")] SMOOTH_ALL = 1,
                     [helpstring("SMOOTH_NONE")] SMOOTH_NONE = 2,
                     [helpstring("SMOOTH_TEXT")] SMOOTH_TEXT = 3
                     }ENUM_SMOOTHING;
```

Multi-Page Processing

By default, PDF files are converted page-wise into multiple, single-page image files, with an Underscore character ("_") added between the filename prefix and the sequential-counter suffix: *e.g.* a 5-page input file named "`convert.pdf`," when converted to `BMP` output, will produce consecutive, individual output files named "`convert_1.bmp`" through "`convert_5.bmp`."

For large numbers of output files (*e.g.* more than 9, more than 99, etc.), the sequence-number suffix will be automatically padded by default with leading zeroes as required, in order to simplify the process of handling the output files in bulk while still maintaining their consecutive order (*e.g.* "`convert_001.bmp`" through "`convert_237.bmp`").

Remember:

- As of the *PDF2IMG* v2.2P1j release, specifying PDF as the output format will invoke a new page-split functionality, accepting a multi-page input PDF document, and generating one or more single-page output PDF documents.
- You can use the optional `-firstonly` command-line argument to do a single, first-page test of the document conversion before launching a large-scale process. (See “FirstOnly” on page 4.10 above for more details.)
- For TIFF output conversions of multipage input PDF files, you can use the optional `-multipage` command-line argument to generate a single, multipage TIFF document. (See “Multipage” on page 4.14 above for more details.)
- You can use the optional `-digits` command-line argument to override the default sequence numbering of output files, and specify your own fixed number of digits for the filename counter suffix. This will also suppress the Underscore character ("_") normally added between the filename prefix and the sequence number. (See “Digits” on page 4.10 above for more details.)
- The COM object offers a `no_first_outputNumber` parameter that can suppress the addition of a trailing "`_1`" on the file name when generating the first output page of the file. *(Default* `False`*)*
- You can use the optional `-output` command-line argument to specify an alternate output filename prefix, in place of the default input filename prefix (see “Output” on page 4.15 above).

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.

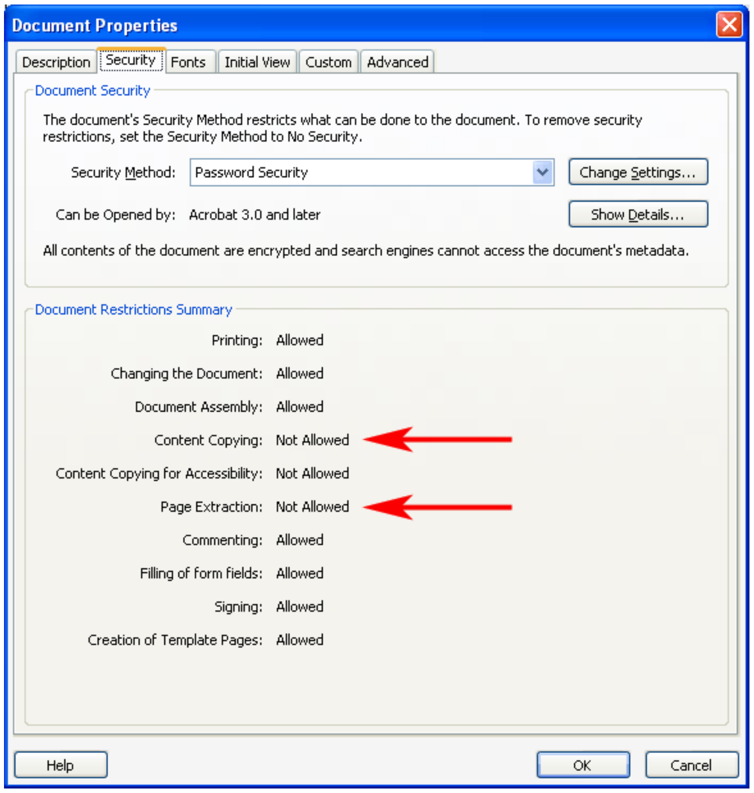
General Usage Notes

Problems with File Opening

Opening a PDF file for conversion is not the same as opening it for viewing. In some cases, file protection issues such as either or both of the following may prevent *PDF2IMG* from completing the conversion process:

- Files which are password-protected against opening will require you to provide that password via either the `-password` command-line argument (if using a console application) or the `pdf2img_new_conversion_with_password` or `pdf2img_new_memconversion_with_password` API calls (for *PDF2IMGCOM* applications).
- The file must allow Content Copying or Extraction (as shown in the *Adobe Acrobat* Document Properties Security display) before conversion can proceed.

In *Adobe Acrobat*, you can display the file's Security settings by opening the file in the viewer and pressing Ctrl+D, then selecting the Security option from the left-hand pane (in *Acrobat* v6.*x*) or upper tab (in *Acrobat* v7.*x* or v8.*x*) of the popup window to check the file's Security attributes:



Document Security

The document's Security Method restricts what can be done to the document. To remove security restrictions, set the Security Method to No Security.

Security Method: Password Security

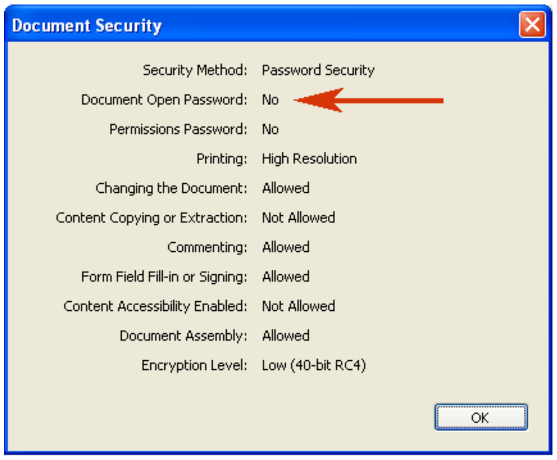
Can be Opened by: Acrobat 3.0 and later

All contents of the document are encrypted and search engines cannot access the document's metadata.

Document Restrictions Summary

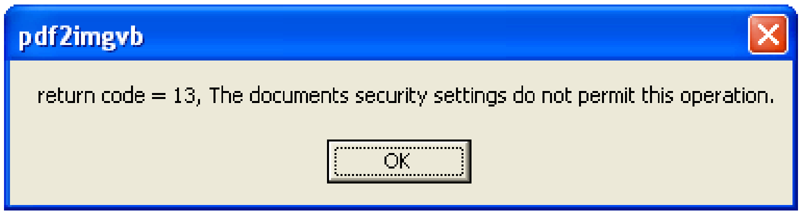
Printing:	Allowed
Changing the Document:	Allowed
Document Assembly:	Allowed
Content Copying:	Not Allowed
Content Copying for Accessibility:	Not Allowed
Page Extraction:	Not Allowed
Commenting:	Allowed
Filling of form fields:	Allowed
Signing:	Allowed
Creation of Template Pages:	Allowed

You should verify that "Content Copying or Extraction" is allowed. If "Password Security" is also indicated, press the Show Details... button to verify that "Document Open Password" is *not* set to "Yes". If it is, you will need to obtain its Open password and provide that to *PDF2IMG*:



Security Method:	Password Security
Document Open Password:	No
Permissions Password:	No
Printing:	High Resolution
Changing the Document:	Allowed
Content Copying or Extraction:	Not Allowed
Commenting:	Allowed
Form Field Fill-in or Signing:	Allowed
Content Accessibility Enabled:	Not Allowed
Document Assembly:	Allowed
Encryption Level:	Low (40-bit RC4)

PDF2IMG may be able to process a file carrying a Permissions Password (one which requires a password before its Security settings can be modified), depending on what other Security settings are in force, but if a Document Open Password is present, you will need to know what it is before proceeding, and you cannot convert a PDF file that disallows copying as explained above. A conversion that cannot proceed due to security or permission problems will return an error message:



return code = 13, The documents security settings do not permit this operation.

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

Font and `CMap` Location Searches

By default, *PDF2IMG* searches for fonts installed and used by Adobe products (*e.g. Adobe Acrobat* or *Adobe FrameMaker*), and will use those if present. However, you can replace that search with your own list of locations (up to 16 as of *PDF2IMG* v1.2) via the `-fontlist` command-line parameter, to direct *PDF2IMG* to other resource locations instead of the Adobe sites.

After searching any `-fontlist` locations you may provide, Windows and UNIX versions always search the following locations, relative to the present working directory:

- `Resource`
- `Resource/Cmap`
- `Resource/Font`

Since `-fontlist` locations are searched first, any resource you provide via the `-fontlist` argument will supercede another of the same name that may appear in one of the default locations.

Also see “Environment Variable Definitions” on page 4.29.

High-Resolution Conversions

Large PDF files processed at high resolution settings may cause *PDF2IMG* to abort with 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. Rasterization of PDF files at high resolution will require large amounts of free memory.

Available Font/CMap Archive File

A separate `.tar` file is available on request which contains an archive of fonts and `CMap` files which are distributed with *Adobe PDF Library* and which are licensed for redistribution by you for use with *PDF2IMG*. Contact your Datalogics Support representative to obtain a copy of this archive.

NOTE: Fonts provided with *PDF2IMG* are licensed for use and redistribution with *PDF2IMG* only.

Environment Variable Definitions

On UNIX platforms, set `LD_LIBRARY_PATH` to the locations of the *Adobe PDF Library*, *PDF2IMG* and GLIBC shared libraries (the latter are usually found in `/usr/local/lib`). They do not need to reside in the same folder(s) but must all be referenced by this variable.

In order to build applications using the *PDF2IMG* library in Linux, the working directory and the GCC 4.*x* libraries directory must be included in `LD_LIBRARY_PATH`. This can be achieved as follows:

```
$GCC_LIBS=/opt/gcc-4.2.1/lib #(Example only; actual path & version may vary)

export $LD_LIBRARY_PATH=$GCC_LIBS:$PWD:$LD_LIBRARY_PATH
```

NOTE: Although the *PDF2IMG* distribution includes copies of the *Adobe PDF Library* shared libraries, users with separate licenses and installations of *Adobe PDF Library* applications may use their current libraries and resources instead.

It is important to understand that *PDF2IMG* can convert a document which references fonts to be used but does not include them, but if those specific fonts cannot be found, *PDF2IMG* must substitute its own. While this is not considered an error condition, it may cause undesirable results, especially in the case of more-obscure or decorative typefaces, or symbol fonts which cannot be otherwise simulated. See “Conversions with Missing Resources” on page 4.30 (following) for more details and examples.

Conversions with ICC Color Profiles

PDF2IMG will honor calibrated colorspaces in PDF files, if ICC color profiles are found during conversion and if output color management is in effect. (As of the *PDF2IMG* v1.6 release, output color management is now set on by default.)

Unless it is suppressed (via either the Color Management Module `-nocmm` command-line call or the `pdf2img_set_colormanagement False` API call, as appropriate), it will also write target ICC profiles to TIF, JPG, PNG or BMP output.

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

Table 4-1: Default Conversion Color Profiles

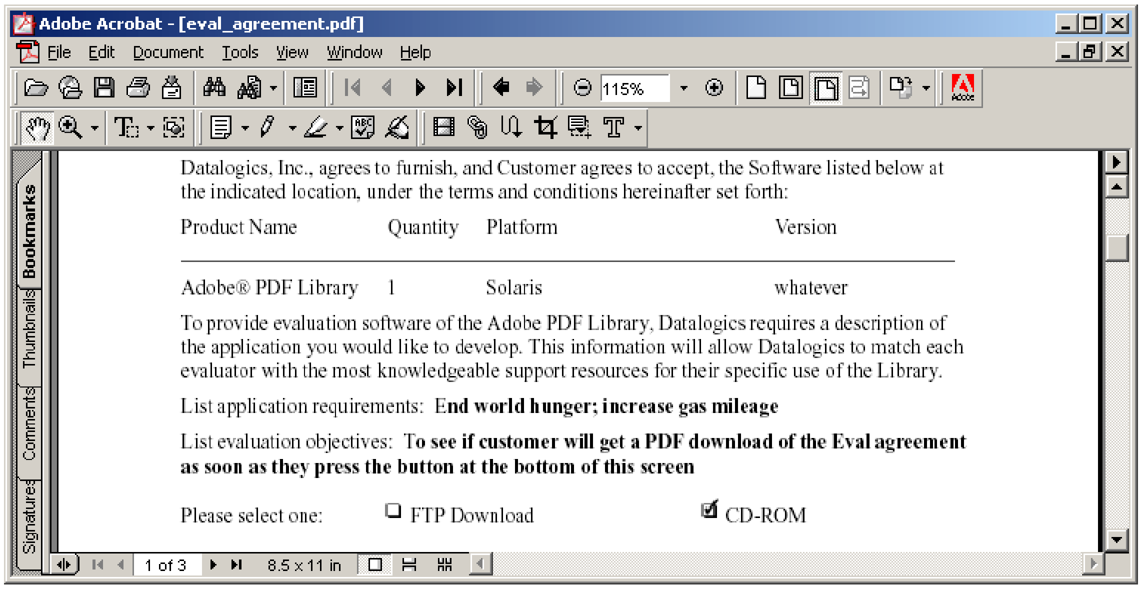
Colorspace	ICC Color Profile (Input)	ICC Color Profile (Output)
RGB	Adobe 1998 RGB	sRGB
CMYK	*Adobe Reader* 9 CMYK	*Adobe Reader* 9 CMYK (Simplified US web coated SWOP V2)
Gray	Gamma 2.2	Gamma 2.2

Conversions with Missing Resources

Ideally, a PDF document should always have its fonts embedded and subset within the document, so that it will not have to rely on external resources when viewed, printed or (in the case of *PDF2IMG*) converted on a system other than the one on which it was created. If it does not carry its fonts within the document, the system receiving the document must locate those resources elsewhere, or use substitutes (*e.g.* font fauxing) if possible.

PDF2IMG will attempt to make use of font resources on the system if the document being converted calls for them but does not embed them within its file. Normally this is done by use of environment variables to point *PDF2IMG* to the right location(s) where font resources will be found (see "Environment Variable Definitions" on page 4.29, preceding). However, if *PDF2IMG* cannot find the appropriate resources, it will generate its own font fauxing, as shown in the following illustrations, in order to complete the processing:

Original PDF Document (excerpted)


Adobe Acrobat - [eval_agreement.pdf]
File Edit Document Tools View Window Help
115%
Bookmarks
Thumbnails
Comments
Signatures
1 of 3
8.5 x 11 in

Datalogics, Inc., agrees to furnish, and Customer agrees to accept, the Software listed below at the indicated location, under the terms and conditions hereinafter set forth:

Product Name	Quantity	Platform	Version
Adobe® PDF Library	1	Solaris	whatever

To provide evaluation software of the Adobe PDF Library, Datalogics requires a description of the application you would like to develop. This information will allow Datalogics to match each evaluator with the most knowledgeable support resources for their specific use of the Library.

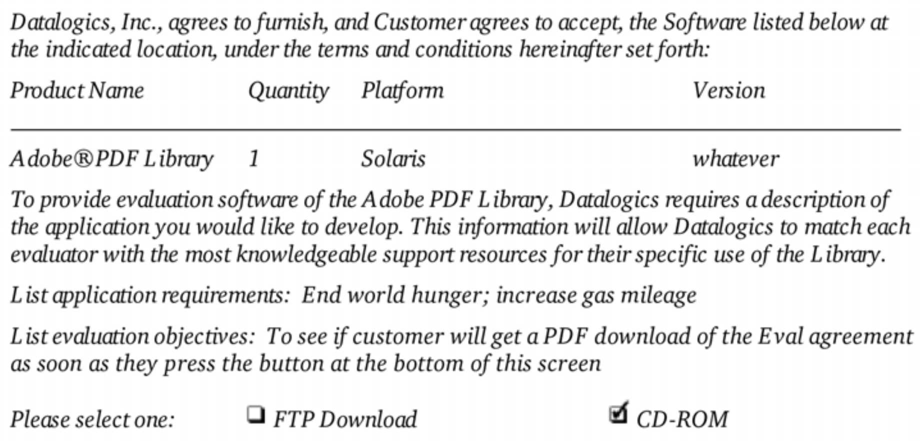
List application requirements: **End world hunger; increase gas mileage**

List evaluation objectives: **To see if customer will get a PDF download of the Eval agreement as soon as they press the button at the bottom of this screen**

Please select one: ☐ FTP Download [✓ CD-ROM]

In the original PDF sample above, certain fonts are referenced but not embedded in the document. As long as the document is viewed on a machine that contains those fonts, it will appear in its original form, as shown here.

GIF Output Conversion (without resources)



Datalogics, Inc., agrees to furnish, and Customer agrees to accept, the Software listed below at the indicated location, under the terms and conditions hereinafter set forth:

Product Name	*Quantity*	*Platform*	*Version*
Adobe® PDF Library	*1*	*Solaris*	*whatever*

To provide evaluation software of the Adobe PDF Library, Datalogics requires a description of the application you would like to develop. This information will allow Datalogics to match each evaluator with the most knowledgeable support resources for their specific use of the Library.

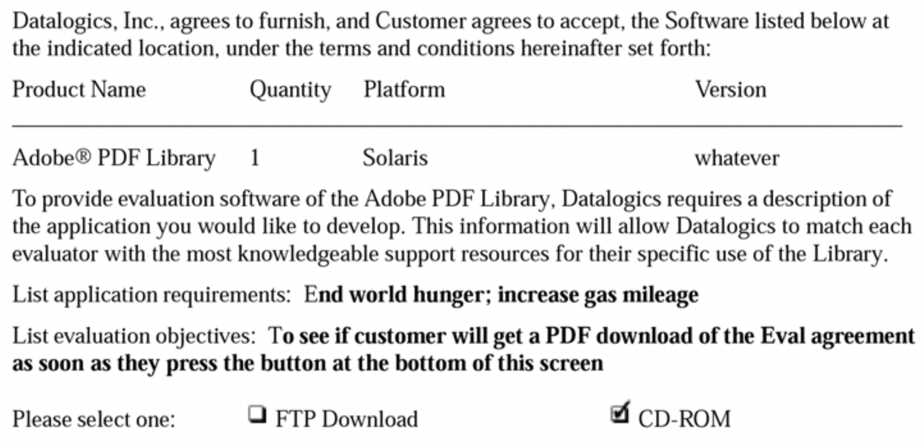
List application requirements: End world hunger; increase gas mileage

List evaluation objectives: To see if customer will get a PDF download of the Eval agreement as soon as they press the button at the bottom of this screen

Please select one: ❑ *FTP Download* [✓ *CD-ROM*]

In this output sample (above), *PDF2IMG* processed the document without access to font resources, and substituted its own in order to complete the processing.

GIF Output Conversion (with resources)



Datalogics, Inc., agrees to furnish, and Customer agrees to accept, the Software listed below at the indicated location, under the terms and conditions hereinafter set forth:

Product Name	Quantity	Platform	Version
Adobe® PDF Library	1	Solaris	whatever

To provide evaluation software of the Adobe PDF Library, Datalogics requires a description of the application you would like to develop. This information will allow Datalogics to match each evaluator with the most knowledgeable support resources for their specific use of the Library.

List application requirements: E**nd world hunger; increase gas mileage**

List evaluation objectives: T**o see if customer will get a PDF download of the Eval agreement as soon as they press the button at the bottom of this screen**

Please select one: ❑ FTP Download [✓ CD-ROM]

After providing *PDF2IMG* with font and Cmap resources (either by pointing it to existing system font resources via environment variables or by unpacking the optional font and Cmap distribution file into its present working directory), it was able to correctly render the converted document.

You may need the font and Cmap resource archives if you are running *PDF2IMG* on a machine which does not have an existing installation of *Adobe Reader*, *Adobe Acrobat* or *Adobe PDF Library*. Contact your Datalogics representative for more information.

pdf2imglib Reference Appendix

This appendix defines API calls available via `pdf2imglib.lib`, for use in adding *PDF2IMG* conversion capabilities to your own applications.

pdf2img_check_for_missing_appearances

(ImageConversion iC, int firstPage, int lastPage)

Return Value: int

Description	This call checks pages within the specified range for nonrenderable annotations or form fields. It returns a value greater than `0` if the page range contains nonrenderable content, or `0` if the page range does not contain nonrenderable 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, writing 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. *(Required)*
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: constant 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	`constant 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 receiive 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_get_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_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_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 here. 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_conversion_with_password` with either a `NULL` pointer or a pointer to zero length string in the password field. 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 call.

6 The password must grant the following permissions on the PDF document:

If using:	Permission Required
Any output format	Open, 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	
Header	`pdf2imglib.h`
Related Methods	`pdf2img_destroy_conversion`, `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 here. 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 field. 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, 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`
	`1` (or any non-zero): `black = 1`/`white = 0`
	(Optional; 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 calls set the Bits Per Channel (`bpc`) value for the output color space: either `1` (*e.g.* 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` *(Optional; Default* `8`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	
Availability	All platforms

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, 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* will assume that Device colors on input are calibrated as shown in the table below (see Technical Notes below), and will assume a corresponding output profile to match. When generating TIF, JPG, PNG or 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` *(Optional; Default* `true`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	
Availability	All platforms

Technical Notes

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

Table A-2: 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. 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: `GRAYcolor`, `RGBcolor` or `CMYKcolor` *(Optional; 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 types. See the table below:

Table A-3: Available Output Types per Colorspace

Colorspace	Output Type
GRAYcolor	TIFF, JPEG, PNG, GIF
RGBcolor	TIFF, JPEG, BMP, PNG, GIF
CMYKcolor	n/a

pdf2img_set_compression
(ImageConversion iC, CompressionCode cmCode)
Return Value: int

Description	This call sets the output compression. This function is for TIFF output only. See Technical Notes below for more details.
Parameters	• `ImageConversion iC`: the data structure containing the specifications for the active, current conversion • `CompressionCode cmCode`: Output compression value: `NOcompression`, `LZWcompression`, `G3compression`, `G4compression` or `JPGcompression` *(Optional; TIFF output only; 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 output. JPEG graphics use DCT compression, PNG graphics use Flate compression, and GIF graphics use LZW compression. BMP graphics are currently not compressed.

2 Due to internal processing problems, the `JPGcompression` value had been removed from the initial *PDF2IMG* v1.3 release, but has now been restored.

3 `G3compression` and `G4compression` values are only valid for B/W (1 channel, 1 bit) TIFF images.

pdf2img_set_enhance_thin_lines

(ImageConversion iC, int newVal)

Return Value: int

Description	This call will toggle the Enhance Thin Lines setting (as used in *Adobe Reader* or *Adobe 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 *(Optional; Default* `1`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	
Availability	All platforms

Technical Notes

1 This reflects a new default rendering setting for `enhance_thin_lines` that was first introduced in the *PDF2IMG* v2.1P1b release, intended to better 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 desired, such as to match previously-rendered output from earlier releases of *PDF2IMG* prior to v2.1P1b.

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 inch. Valid range is `12` to `2400`. *(Optional; Default* `300`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	`pdf2img_set_vert_res`
Availability	All platforms

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 JPG or TIF in one pass. If not, it rasterizes the page in bands (strips) in order to use smaller chunks of memory, then 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 `1,000,000` to `4,200,000,000` *(Optional; Default* `300,000,000`*)*
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 pixels. TIF output has been tested up to a band of 68898 x 34449 pixels in width.
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 sample. Higher values are allowed if passed in through this `pdf2img_set_max_band_memory` API.

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, with each converted PDF page corresponding 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 *(Optional; TIFF output only; 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, accounting 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 *(Optional; 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 *Adobe PDF Reference Guide*. (See Technical Notes below, and Section 10.10.1 of the *Adobe PDF Reference Guide*: "Page Boundaries") 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` or `BOUNDINGbox` *(Optional; Default* `CROPbox`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	
Availability	All platforms

Technical Notes

1 `PDFRegionCode rCode` values below represent the following regions of the PDF page. For full details on all values for `PDFRegionCode rCode` except `BOUNDINGbox`, see Section 10.10.1 of the *Adobe PDF Reference Guide*: "Page Boundaries."

Table A-4: 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, including any extended areas surrounding the finished page for bleed, printing marks, or other such purposes.	None

PDF Region	Description	Default Value
`ARTbox`	Extent of the page's meaningful content (including potential white space) as intended by the page's creator	Page's Crop Box
`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

Descriptions summarized from Adobe PDF Reference Guide *v1.7, Section 10.10.1*

pdf2img_set_output_type

(ImageConversion iC, OutputTypeCode oCode)

Return Value: int

Description	This call specifies the output graphic type(s) (one or more) into which *PDF2IMG* will convert the document. See Technical Notes below. *(Required)*
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` or `GIFoutput` *(Required; No Default)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	
Availability	All platforms

Technical Notes

1 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.

Table A-5: Available Output 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_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 desired balance 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`, where `1` is the smallest file; `100` is the highest quality image. *(Optional; Default* `75`*)*
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` *(Optional; Default* `75`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	
Availability	All platforms

Technical Notes

1 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_reverse
(ImageConversion iC, unsigned short int reverse)
Return Value: int

Description	If a non-zero, positive `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 *(Optional; 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, 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.*(Optional; 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,Liine 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 inch. Valid range is `12` to `2400`. *(Optional; 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, not on paper: non-printing annotations will be shown; 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 for, and printable annotations will appear. Non-printing annotations, those used for on-screen display only, will be suppressed. *(Optional; Default* 0*)*
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 *(Optional; Default* `0`*)*
Return Value	`int`
Exceptions	
Header	`pdf2imglib.h`
Related Methods	`pdf2img_setprintannot`
Availability	All platforms

Technical Notes

1 This command can be overridden by the `pdf2img_setprintannot` setting.

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 *(Optional; 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 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, and no others.

2 This command will override the `pdf2img_setasprinted` setting.

pdf2img_start_multipage

(ImageConversion iC, constant 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;Optional; 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 filename 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`
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

Index