FTIP2I
PDF to Image Conversion Utility

version 1.1b

July 12, 2004

FTI Consulting, INC.
333 Wacker Road, Chicago IL, 60606

Chapter 1 Overview

Chapter 1 provides an overview of the FTIP2I utility.

Chapter 2 FTIP2I Installation

Chapter 2 discusses installation of the FTIP2I on supported platforms and the tool used to build the installer.

Chapter 3 Development Tools

Chapter 3 describes the development tools used to create the FTIP2I utility: Ghostscript, Perl, and Perl2Exe.

Chapter 4 Dealing with non-conforming PDF documents

Chapter 4 describes how the utility deals with non-conforming PDF documents.

Chapter 5 Quality of the generated images

Chapter 5 discusses the issues that affect the quality of the generated images.

Chapter 6 Font Issues

Chapter 6 discusses font issues in the rendering and interpretation of PDF files.

Chapter 7 Filenaming Issues

Chapter 7 discusses the renaming of files to conform to Ghostscript's subset of valid filenames.

Chapter 8 Interface Issues

Chapter 7 discusses command line parameters: Return code status, Performance issues, and monitoring.

1. Introduction

The FTIP2I is a command line utility for converting PDF files to PNG or TIFF image files. The utility supports:

PDF 1.2 features: Security, annotations, binary files.

PDF 1.3 features: Structure, Digital Signatures, embedding,

JavaScript, RTL, color separations, PS3

PDF 1.4 features: Transparency, XML & Metadata, Tagged PDF

PDF 1.5 features: selectively view or hide text or graphics, JPEG2000

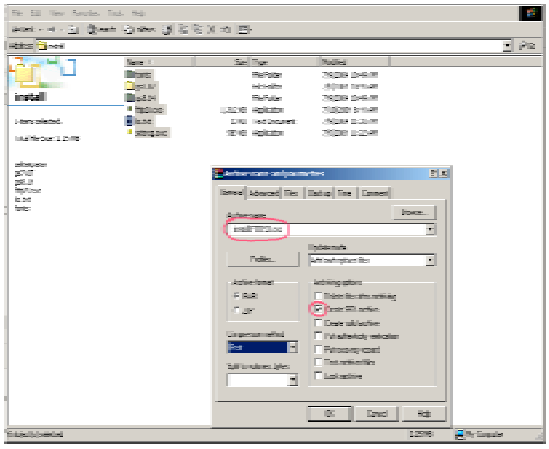
compression, Enhancements to Tagged PDF, Additional annotation types, and other changes to annotations

2. Installation

The installation of the utility is straightforward: installation files are copied to the destination directory (default \Program Files\FTIP2I) and then the HKEY_LOCAL_MACHINE \Software\FTIP2I key is update to reflect the destination path.

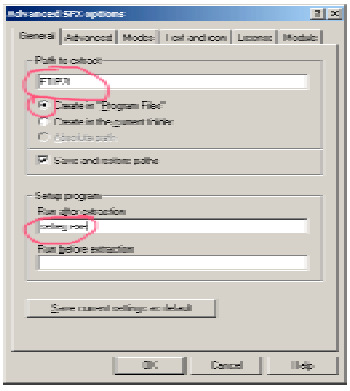
The installer is built with WinRAR (http://www.rarlab.com/). To create a distribution application :

1. select all the files in the distribution directory then press the right mouse button; select “Add to Archive”. The WinRAR window will appear.

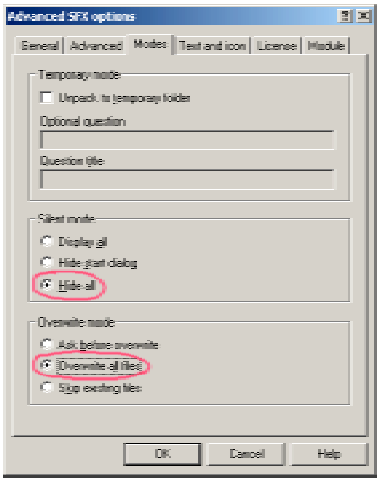


2. Click on the “Advanced” tab and then the “SFX Options” button (Self-expanding executable!) to display the advanced SFX control.

3. Select “Create in Program Files” and select “FTIP2I” as the path to extract to. Enter “setreg.exe” in “Run after execution”. This program registers the installation path into the windows registry. Trial Max can check to see if this key exists in the registry to determine if the program has been installed. If the key is not found this installer should be run.



4. Select “Hide all” and “Overwrite all files” to make the installation executable transparent and fully automatic. To create an interactive installation select “Display all”.



5. Click “OK” to create the installer.

3. Tools used.

Ghostscript:

Ghostscript is an interpreter for the PostScript language and is the main workhorse for the FTIP2I utility. Ghostscript is distributed under the AFPL free public license (http://www.ghostscript.com/doc/cvs/Public.htm). A PostScript interpreter transforms the PDF file to PS and renders text, graphics commands, and bitmapped images into a bitmapped format. The FTIP2I uses two versions of ghostscript to handle handle PDF conversions: AFPL 8.14 and GNU 7.07. The 8.14 version has higher quality rendering and better support for the PDF features but is more sensitive to improperly formed PDF files created using non-Adobe tools such as www.verypdf.com.

Perl.

The utility software code was created in ActivePerl 5.6.1 build 638.

(http://www.activestate.com/Products/Download/Download.plex?id=ActivePerl)

General System Requirements

Recommended 75MB hard disk space

Windows Installer requirements : MSI version 2.0

For Win95/98/Me

http://downloads.activestate.com/contrib/Microsoft/MSI2.0/9x/InstMsiA.exe

For WinNT/2000
http://downloads.activestate.com/contrib/Microsoft/MSI2.0/NT/InstMsiW.exe

Windows XP does not need to be updated

Windows Server 2003 does not need to be updated

Perl2Exe

Perl2Exe V8.00 for Win32 is a command line program (http://www.indigostar.com/perl2exe.htm) for converting perl scripts to executable file allowing stand alone programs to be created in perl that do not require the perl interpreter. Executable files can be shipped without the having perl source code.

Usage: perl2exe ftip2i.pl

Generates ftip2i.exe. No additional dlls are required for the installation. The executable is standalone.

4. Dealing with non conforming PDF documents

The issue of PDF files that do not conform to the PDF specifications is a major one for any PDF interpreter. The PDF 1.4 specification[1] for Acrobat 5, and the PDF 1.5 specification[2] for Acrobat 6 are poorly understood by many tool vendors. Ghostscript does not have the same flexibility in adapting to non-conforming documents as the Adobe products do. The Adobe products implement PDF functionality through the PDF library whose release always coincides with Reader technology. Unfortunately, Adobe has a very restrictive policy on the release of the PDF library, limiting its use on a case-by-case basis to strategic partners, System Integrators, and Enterprise IT developers. There are very few partners permitted to have the interpreting/rendering technology necessary for a PDF to image converter; most partners are restricted to PDF generating technology and forms processing.

Ghostscript does provide a great deal of flexibility in dealing with the real world examples of non-conforming PDF files: e.g. 1.4 features showing up in PDF files claiming to be PDF 1.3; pointers to blank lines before the xref line rather than the xref itself; The PDF specification that the “xref” be on a line by itself. Acrobat Reader, however, only requires the line to begin with “xref”. GS adapts to this as well.

The FTIP2I utility uses two version of ghostscript in combination to conversion of the broadest range of PDF documents. GS 8.14 implements the PDF 1.5 features, but at a cost of requiring stricter conformity with the PDF specification. The great majority of PDFs in the field are version 1.3 and 1.4 compatible and GS version 7.07 does a better job dealing with malformed documents in this range.

When converting PDF files, the FTIP2I utility first tries to convert using v.8.14. If unsuccessful, the conversion is attempted with v7.07. If that is unsuccessful a fatal error will be flagged.

Older PDF files created by the Delphion (www.delphion.com) have invalid xref references to the first pdf object. The file US05710835.pdf is an example of this error. The xref table starts with

```
xref
   1   117
0000000000 65535 f
```

instead of

```
xref
   0   117
0000000000 65535 f
```

This is a violation of the PDF specification. Unfortunately, there's no simple work-around in Ghostscript. Apparently, Acrobat Reader does further analysis and cleanup of the PDF files to repair broken xref tables. Ghostscript relies on the published PDF specification and doesn’t correct this sort of problem.

Due to the wide spread use of the Delphion, Intellectual Property Network - https://www.delphion.com) and the consistency of this error, the FTIP2I utility patches these faulty PDF files by patching the file with the regular expression:

$buf =~ s/(\r?\n?xref\r?\n?) *(1) *(\d*\r?\n?)/$1 0 $3/;

and writing the PDF file to the destination directory with the filename input.pdf. This file is used as the input file for the conversion.

Files created by the OS X Quartz PDF context are flag for conversion by the 7.07 version. This rerouting of the normal flow of control is to avoid an exception from being thrown- and caught by the .NET debugger (or VS 6.0). That catch blocks the program until the execution is continued after user

[1] http://partners.adobe.com/asn/acrobat/docs/File_Format_Specifications/PDFReference.pdf
[2] http://partners.adobe.com/asn/acrobat/sdk/public/docs/PDFReference15_v5.pdf

intervention. This problem does not apply to end user environments where the utility code handles the problem automatically.

TODO: Tests need be made of converter performance on files that secured to prevent alterations, printing or annotation. Also, encrypted files.

5. Quality of the generated images.

The appears to create imaged versions of the PDFs that are faithful reproductions of the original documents as viewed by the Adobe Reader software. There are several issues of importance in determining the quality of the converted image.

1. Anti-aliasing of text and graphics. Text oriented documents converted to TIFF group IV images have the "jaggies" at high magnification compared with files rendered to PNG 24 bit format. To improve the smoothness, the images can be rendered at higher resolution and then subsampled or converted first to gray scale (anti-aliasing done) and then converted to Group IV. Samples:

PNG – 300 dpi – 24 bit

TIFF Group IV – 300 dpi

2. All output images have the same format (TIFF or PNG) regardless of the actual color qualities of the individual pages. The settings are determined by the global color settings for the document. Some background on how this happens: The Perl modules for PDF information gathering (# colors, # pages, page size, orientation, etc.) are not able to handle the malformed documents in the data set. Instead, the FTIP2I utility scans through the PDF files to identify the color content of the document. Some more work needs to be done to make this identification correct. The second part of the problem is the inability to parse the PDF and split into individual pages at the start of conversion. A number of different techniques were evaluated in an attempt to do this, but none were successful over the whole FTI sample data set. The pdfsplit utility from www.pdfeverywhere.com came closest to splitting the PDFs down where there were then able be be analyzed on a page basis. The malformed PDF syntax was also cleaned up in the process. Unfortunately a large number of files were not able to be processed.

A couple of alternative approaches are possible. The documents that contain some color content pages would first be converted to PNGs and then the B&W pages would be rerendered as TIFFs as needed. This approach has the overhead of two separate image file conversions and the requirement for the ImageMagick library, or the equivalent toolkit, for raster to raster conversion. GS doesn't have flexibility in performing raster transformations. Doing this would be not much effort, but it may be desirable to use the standard TrialMax utilities for this purpose. Another approach is to create a utility that can cleanup the PDF syntax faults and perform a conversion.

Another approach is to clean up the PDF error and perform a split function using custom code. I've made some experiments to determine the feasibility of writing a series of methods to do this. The first and probably most difficult one would read through the PDF file in binary mode, writing out to a text file formatted so that one finding (along with its byte offset and byte length and other formatting information) occurs per line. The second method would read the text output of the first, collate and check and cross-check its findings, and then write out its own formatted text file which identifies (also one per line) any

findings of errors. The third method would take the two forms: a) reading the output of the second function, it produces a prose description of each error if any (including the line number and column number where it occurs when read in) and b) also reading the output of the second script, it produces a formatted text file which prescribes the steps to be taken to repair the flawed pdf file and either then performs that itself or leaves it to another, fourth method.

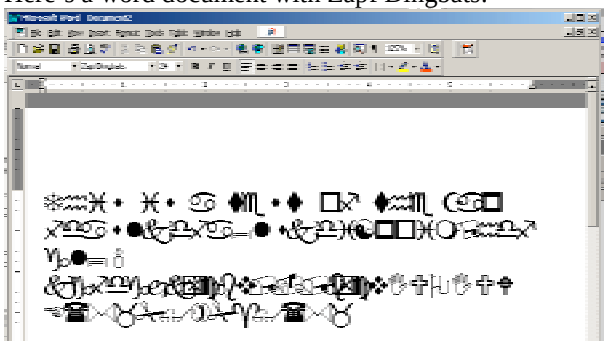
This approach should reveal a fair number of the errors in the pdf files which are presumed to exist, though certainly not all of them (such as parameter errors in graphics operations). It should be at its most thorough within that part of the pdf file occurring after the "xref" line and in tracing back the references from there to the numbered indirect objects defined earlier in the file. I'm also hoping Writing this pdfchecker as a set of related functions might enable me to spot more easily some errors that are occurring commonly in the files in question so that their fixing might be speeded by having more systematic information as a basis to go on. Finally, I should say that the pdfchecker would be expected to take into account the increased formatting requirements of linearized pdf documents. There were a large number of these documents in the exhibits folder. (perhaps a week of effort would be required for this approach).

Another technique would be to create a hybrid output images device that would selectively create a PNG or TIFF file based on image content. This code would have to be maintained with GS upgrades. More reasearch is need to determine the feasibility of this approach.

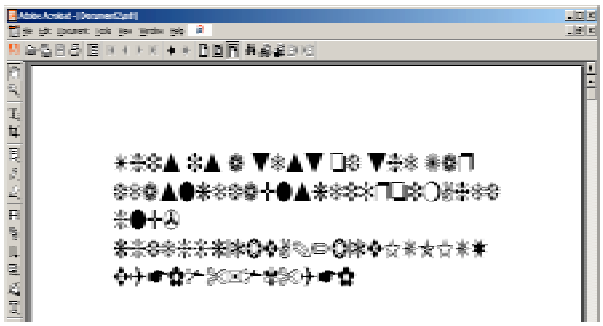
Perhaps the best approach for now is to perform the double image conversion. With more experience with the types of documents that the FTIP2I will be encountering in the field, one of the other techniques could be developed as needed at a later date.

6. Font Issues

Adobe Distiller/ Adobe acrobat / and Acrobat reader have a number of QC issues that relate to font lookup, translation, and embedding. A quick example of font handling with the default distiller settings:
Here's a word document with Zapf Dingbats:



and then converted with Distiller to PDF:



and then this PDF converted with FTIP2I as an image file:

The FTIP2I provides an extreme example of conversion error
Due to poor font embedding choices. Most font translations will net have
Such serious visual consequences; kerning differences and minor
Glyph difference will be the most likely result.

To minimize the font matching irregularities and to make for more consistant conversion, the Ghostscript is configured to search only the \Program Files\FTIP2I\fonts directory and not the XP or 2000 system font directory. In the event that critical conversions are required that reference specialized fonts such as barcodes or Simplified and Traditional Chinese (/Adobe-CNS1 or /Adobe-GB1), the GS_FONTS environment variable should be set to include those fonts in the path.

The list of the fonts used in each PDF file and the corresponding mapping is indicated in the output directory\font.log. These files are reference against a list of known bad fonts imbedded in the script. Perhaps an administrative utility would be nice at some point to add or remove files from this "bad" list when new problem fonts are detected or mapping fonts are installed on the system.

7. Filename Issues

Certain characters in the input filename are mapped or suppressed so as not to conflict with the Ghostscript filenaming conventions ("(", ")", " .pdf"). If the Directory and/or file name is not changed an error will report during the conversion and no image file will be generated. This should be temporary with the name reverting back to the original name at the completion of the conversion process.

8. Interface Issues

Command line parameters

Return code status

The return code from the exe is masked with 255 and then shifted left 8 bits. If the number of pages converted is returned by the return code, a maximum of 127 pages may be indicated. To determine the number of pages, the conv.log file should be scanned for the line in the form:

Processing pages %d through %d.

The first integer should always be 1. The second should indicate the number of image file generated if the return code status was "0" (success). This number should coincide with the number of image files in the output directory.

Will the trialMax software need a way to identify whether a given page is TIFF or PNG?

Performance issues

TODO: Process priority. The PDF conversion process will take the CPU time that is given it. It should be determined at what process priority the conversion should be run. If other system activity is required while the conversion process is underway, it would be nice to set the priority towards background process. ~ 1 hour

Monitoring issues

TODO: A monitoring process that returns the page number being worked on would make for a better user experience when very large files are being converted. There is currently little user feed back during these conversions. This can be implemented by selecting low level GS debug info which will stream to the stdout. The control process will scan for page change data or other messages of interest for the user. 1-2 hours.

GUI Interface

A gui wrapper around the command utility would probably be a nice thing to have out in the field. 3-5 hours.

Command line options

C:\Program Files\FTIP2I> **ftip2i**

Error: No input file directory specified

ftip2i.pl 1.1b: PDF to image conversion utility.

usage: ftip2i.pl [-hvd] [-f file]

-in <input file/dir> : PDF File or directory containing PDF files
-out <output dir> : Destination directory for images
[-help] : this (help) message
[-verbose] : verbose output
[-res] : resolution (dpi)
[-retry] : force image conversion to be run again

example: ftip2i.pl -v -in infile.pdf -out outputdir

Patent Issues

Here's a quick run down on the main issues:

A. Adobe Patents. The Adobe developer resources page is a useful location for their policy for the patents they hold on PDF technology (http://partners.adobe.com/asn/developer/legalnotices.jsp). The patent 5,860,074, which is excluded, deals with "optimized" PDF to provide quick early display of pages containing both text and other objects. It licensed only for generating (optimized) PDF, not for rendering it. Adobe made this decision to insure that other PDF views will be slower than theirs. Not applicable to ghostscript.

B. Accurate Screens. Ghostscript uses an alternate screening technique that achieves comparable results.

C. TrueType rendering is implemented with un-hinted TrueType rasterization infringing Apple's patents. Fonts that abuse the TrueType specification and produce incorrect fonts when the hints are ignored are put on the "bad font list". These are rare.

D. PDF 1.4 contains JBIG2 developed by the JBIG2 standards body. The standards board strove to make JBIG2 patent and royalty free, unlike the original JBIG . Ghostscript provides only a decoder for JBIG2, no encoder