PDFTron PDF2SVG™
User Manual

Version 6.0

LEGAL STATEMENT AND COPYRIGHT NOTICE

PDFTron PDF2SVG™ Command-Line Application User Manual
Part number: PDFTRON-4-PDF2SVGCMD
Last Updated: July 19, 2013

All information contained herein is the property of PDFTron Systems, Inc. (“PDFTron”). No part of this publication (whether in hardcopy or electronic form) may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of PDFTron Systems, Inc.

The information in this publication is provided for informational use only, is subject to change without notice, and should not be construed as a commitment by PDFTron. PDFTron assumes no responsibility or liability for any loss or damage that may arise from the use of any information in this publication. The software described in this user manual is furnished under License (enclosed in the software package) and may only be used or copied in accordance with the terms of that License.

PDFTron and the names of PDFTron products referenced herein are either trademarks and/or service marks and/or registered trademarks of PDFTron Systems, Inc. PDFTron, PDFNet SDK, PDF/A Manager, PDF2Image, PDF2SVG, PDF2Text, PDF2XPS, XPSConvert, PDFTron PDFSecure, PDF PageMaster, CosEdit, PDFNet SDK, PDF2Image SDK, PDF2SVG SDK, PDF2Text SDK, PDF2XPS SDK, XPSConvert SDK, PDFSecure SDK, PDF PageMaster SDK and associated Logos are either trademarks and/or service marks and/or registered trademarks of PDFTron Systems, Inc.

Any other brand or product names mentioned in this publication are the registered trademarks or trademarks of their respective holders. Mention of a product in this document does not necessarily imply endorsement of the product.

TABLE OF CONTENTS

Legal Statement and Copyright Notice 2

1. Introduction 5

1.1 An Introduction to PDFTron PDF2SVG 5

1.1.1 Key Functions 5
1.1.2 What's New in PDF2SVG? 6
1.1.3 Why SVG? 6
1.1.4 Common Use Case Scenarios 6
1.1.5 Operating Systems Supported 6
1.1.6 System Requirements 6

2. Installing and Uninstalling PDF2SVG 7

2.1 PDF2SVG Installation 7

2.2 Demo Version Installation 7

2.3 Registering PDF2SVG on Your Server(s) 8

2.4 Uninstalling PDF2SVG 8

3. Overview 9

3.1 Basic Syntax 9

3.2 Command-Line Summary 10

3.3 Basic Usage 133

3.3.1 How to save converted files in a given folder? 13
3.3.2 How can I control the output names of generated files? 13
3.3.3 How do I create compressed SVG (SVGZ)? 13
3.3.4 How do I produce stand-alone SVG? 13
3.3.5 How do I open a password protected PDF? 14
3.3.6 Why is PDF2SVG generating page thumbnails and the XML summary document? 14
3.3.7 How do I specify which pages to convert? 14
3.3.8 How do I batch-convert files? 15
3.3.9 How can I show/hide crop marks or the trim region? 15
3.3.10 What quality can I expect form the SVG output? 15

3.4 General Usage Examples 17

Example 1. The simplest command line: Convert PDF to SVG. 17
Example 2. Convert PDF to compressed SVG and without thumbnails and XML summary. 17
Example 3. Convert a password protected file to SVG. 17
Example 4. Convert all PDF document in a given folders to stand alone SVG. 17

3.5 Batch Processing and the Use of Wildcards 18

3.6 Exit Codes 19

3.7 Frequently Asked Questions 20

3.7.1 Why do conversions stop working after entering API key and secret (Error Code 7)? 20

3.7.2 What is SVG? **20**
3.7.3 Why are some text rasterized (no longer vector)? **20**
3.7.4 How do I create an SVG eBook? **20**
3.7.5 Is it possible to customize the "look-and-feel" of the SVG eBook (HTML wrappers)? **21**
3.7.6 Why are some fonts not accurate? 21
3.7.7 Why is a white space separating neighboring pictures? **21**
3.7.8 Can I integrate PDF2SVG with my client/server application? **21**
3.7.9 Does PDF2SVG have any dependencies on third party components/software? **21**

3.8 XML Summary Document 22

4. Support 26

4.1 Reporting Problems 26

4.2 Contact Information 26

1. Introduction

1.1 An Introduction to PDFTron PDF2SVG

PDFTron's PDF2SVG is an efficient, stand-alone command-line application that enables users to convert and publish PDF documents as SVG (Scalable Vector Graphics), the open-standard W3C recommendation for high-end graphics on the web.

PDF2SVG enables high-quality conversion from PDF to SVG that maintains the original layout of the document and preserves hyperlinks, colors and fonts. The resulting self-contained and compact SVG files can be distributed, viewed, edited, stored, printed, and published onto web sites. For quick and easy document navigation and viewing, PDF2SVG can be configured to create page thumbnails as well as an XML summary describing the document components such as metadata, bookmarks, annotations, etc. Using XSLT or any other XML processor, the user can quickly generate custom HTML and JavaScript 'wrappers' that enable the user to browse multi-page documents in any web browser.

Like other PDFTron products, PDF2SVG does not rely on any other third-party software.

PDF2SVG is also available as a software component for embedding into third-party applications. Please see http://www.pdftron.com/pdf2svg for more information.

1.1.1 Key Functions

- High quality conversion from PDF (Portable Document Format) to SVG (Scalable Vector Graphics) that maintains the original document layout and preserves hyperlinks, colors and fonts.
  - Font support: Type1, TrueType and Type0/CID Fonts
  - Color: ICC, DeviceN, Separation, RGB, CMYK, Indexed, etc.
  - Support for encrypted PDF documents (40 and 128 bit RC4, 128 bit AES).
  - Support for all kinds of patterns, functions, and compression schemes.
  - Support for all annotation types.
- Compression and SVG optimization: Converted documents can be saved as SVGZ compressed files for fast web downloads.
- PDF2SVG allows for generation of stand-alone SVG files (i.e. SVG files with embedded resources) as well as for SVG files with shared resources.
- All text is converted to Unicode dramatically simplifying text manipulation, editing, and searching. PDF2SVG makes every attempt to map text to a Unicode public area, resulting in better repurposing and text-searching capabilities of converted documents.
- Font embedding and font substitution option: The font embedding option allows for reliable and accurate font reproduction. The font substitution option produces smaller files but may not produce 100% accurate fonts on all systems.
- Thumbnail generation.
- Option to extract document-level information that has no counterpart in SVG (e.g. metadata, bookmarks, annotations, etc).
- Batch conversion: Using PDF2SVG you can easily convert single files or whole PDF repositories.
- Efficiency: PDF2SVG is based on PDFNet SDK, making it extremely fast and efficient. The conversion speed is suitable for interactive and dynamic applications.
- Support for all versions of the PDF format (PDF 1.0 to ISO32000).

1.1.2 What's New in PDF2SVG?

- Based on the latest PDFNet 6.0 release. See http://www.pdftron.com/pdfnet/whatsnew.html
- Flattener options to help reduce complex PDF files into simpler and faster SVG.
- OpenType font support added, and all fonts can now be shared between pages.
- On-Premises PDFTron Web Services (PWS) pay-as-you-go enabled.

1.1.3 Why SVG?

There are many benefits of converting your documents to SVG:

- SVG is a W3C (Web Standards Consortium) standard format and is backed by a large number of companies and non-profit organizations.
- Free SVG viewers are widely available on major platforms and operating systems.
- Browser support for SVG is now common.
- Because SVG is based on XML, the document can be easily edited in a text editor. SVG XML content can be linked to back-end business processes such as databases, application servers, and other rich sources of real-time information.
- There are a growing number of affordable and powerful SVG authoring and editing solutions.
- Enhanced search capabilities. All text in SVG is stored in standard XML syntax and Unicode encoding that makes searching operations within a document or across large collections of documents a breeze.

1.1.4 Common Use Case Scenarios

- Server-based, on-demand conversion of PDF documents to SVG.
- Batch processing of PDF collections. PDF2SVG is particularly useful in assembling product catalogues and brochures.
- PDF content extraction and repurposing through SVG and XML.

1.1.5 Operating Systems Supported

- Windows 8, 7, 2008, Vista, XP, 2003, 2000, NT, 98
- Mac OSX
- Linux

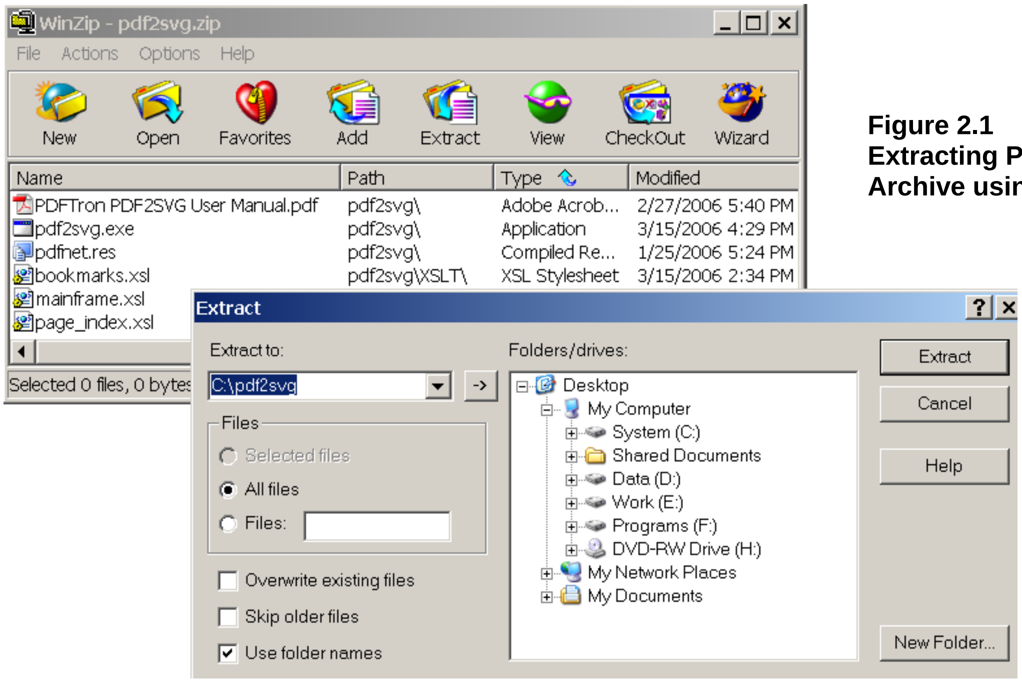
1.1.6 System Requirements

- At least 10 MB of free disk space.
- Memory requirement is heavily dependent on the nature of the document being converted into an image file.

2. Installing and Uninstalling PDF2SVG

2.1 PDF2SVG Installation

PDF2SVG Command-line Application is supplied as a download from a distributor or directly from www.pdftron.com. The release is packaged as a .zip file. (pdf2svg.zip). Unzip the archive in the desired location and make sure to preserve the directory (folder) structure when extracting the archive. In order to register the software, copy the license file provided to you into the "pdf2svg" folder.



Name	Path	Type	Modified
PDFTron PDF2SVG User Manual.pdf	pdf2svg\	Adobe Acrob...	2/27/2006 5:40 PM
pdf2svg.exe	pdf2svg\	Application	3/15/2006 4:29 PM
pdfnet.res	pdf2svg\	Compiled Re...	1/25/2006 5:24 PM
bookmarks.xsl	pdf2svg\XSLT\	XSL Stylesheet	3/15/2006 2:34 PM
mainframe.xsl			
page_index.xsl			

Selected 0 files, 0 bytes

Extract to: C:\pdf2svg

Folders/drives:

Files
○ Selected files
● All files
○ Files:

☐ Overwrite existing files
☐ Skip older files
☑ Use folder names

Figure 2.1
Extracting PDF2SVG Archive using WinZip

2.2 Demo Version Installation

If you wish to evaluate the product, you can download the demo version of the product without any serial number or license key.

To do this, go to PDFTron's **Downloads** page at www.pdftron.com/downloads.html. Click on the appropriate product version/name, which will bring you to the product and the appropriate link for the demo download. Simply Download the zip file (pdf2svg.zip) and extract the archive in the desired location, while making sure to preserve the directory (folder) structure when extracting the archive. This will provide you with a working copy of the application. The limitation of the evaluation version is that all pages in processed documents will have a demo stamp.

2.3 Registering PDF2SVG on your Server(s)

When you are ready to switch to production, and you want to get rid of the watermarking feature of the demo version, follow these steps to register PDF2SVG and sign up for an On-Premises PDFTron Web Services (PWS) pay-as-you-go account.

1. Go to https://api.pdftron.com
2. Click on the “Sign up” button and fill in the registration details.
3. Shortly after the registration you will receive a confirmation email with an activation link and your API ID and API Secret, which is required to connect to Cloud API from your app.
4. To activate subscription, log into your account and proceed to “Subscription” page to select your account plan.
5. Select the plan that meets your projected conversion volume and fill in your billing info.
6. Open ‘docpub.lic’ in a text editor.
7. Fill-in API_ID and API_Secret that you obtained via email (in step 3). You can also find the same information under the 'Account' section in your PDFTron Cloud account. For example:

```
#-------------------------------------
# License Information:
#-------------------------------------
API_ID = your_API_ID_here
API_Secret = your_API_Secret_here
#-------------------------------------
# To register the Software, save the attachment in the 'docpub' folder.
# When extracting the archive, please make sure to preserve the directory (folder)
```

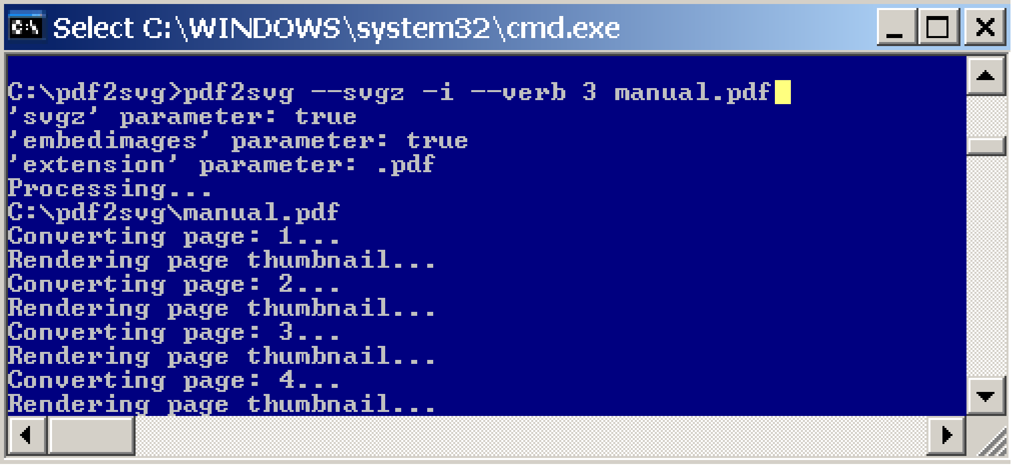
8. Once registered, any conversions you perform will be free from demo stamps and will be reported in your PDFTron Cloud management console (https://api.pdftron.com/v2/console).

2.4 Uninstalling PDF2SVG

To remove PDF2SVG from a computer, simply delete the “pdf2svg” folder.

3. Overview

PDFTron’s PDF2SVG is a command-line application designed to convert PDF files to SVG, the open-standard W3C recommendation for high-end graphics on the web. The flawless conversion process creates web-ready SVG documents. This section covers the basic use of PDF2SVG explaining all the available options.



```
C:\pdf2svg>pdf2svg --svgz -i --verb 3 manual.pdf
'svgz' parameter: true
'embedimages' parameter: true
'extension' parameter: .pdf
Processing...
C:\pdf2svg\manual.pdf
Converting page: 1...
Rendering page thumbnail...
Converting page: 2...
Rendering page thumbnail...
Converting page: 3...
Rendering page thumbnail...
Converting page: 4...
Rendering page thumbnail...
```

Figure 3.0   PDF2SVG Command-line Application.

3.1 Basic Syntax

The basic command-line syntax is:

```
pdf2svg [options] file1 file2 folder1 file3 ...
```

3.2 Command-Line Summary

The following command-line arguments are available for PDF2SVG.

Option	Parameter	Description
-o or --output	e.g. -o myfolder -o "C:\My Folder1\F2"	The output folder used to store converted files. By default, the currently selected working folder will be used to store converted SVG files.
--prefix	--prefix myprefix	The prefix for the output SVG file. The output filename will be constructed by appending the prefix string, the page number, and the appropriate extension (e.g. myprefix1.svg, myprefix2.svg, etc). The prefix option should be used only for conversion of individual documents. By default, the each input filename will be used as a prefix.
--digits	--digits 4	The number of digits used in the page counter portion of the output filename. By default, new digits are added as needed; however this parameter could be used to format the page counter field to a uniform width (e.g. myfile0001.svg, myfile0002.svg, etc).
--subfolders		Process all sub-directory for every directory specified in the argument list. By default, sub-directories are not processed.
-a or --pages	Convert page 1,3, and 10: -a 1,3,10 Convert all even pages: -a even Convert pages in the range from 3-11 and page 50: --pages 3-11,50 Convert all odd pages and all pages in the range from 100 to the last page: -a odd,100-	Specifies the list of pages to convert. By default, all pages are converted.
--svgz		Compress output SVG files using GZIP/SVGZ compression. The default extension for compressed SVG is 'svgz'. By default, generated SVG output is not compressed.
-i or --embedimages		Embeds all images. Using this option it is possible to create self contained SVG files (i.e. files without any references to external resources). Although it is sometimes desirable to create self contained files, this option can result in slower rendering in some viewers. The files with embedded images may also be slower to download over the Net, and because images can't be shared among different pages the total file size for the entire document may increase.

		By default, all images are saved as external files.
--nofonts		Disables conversion of font data to SVG. This option will usually result in the smaller SVG file size, but due to font substitution the text may not render accurately. By default, all available fonts are converted to SVG.
--svgfonts		UseSVG fonts instead of Opentype fonts.
--embedfonts		Embeds all fonts. Using this option it is possible to create self contained SVG files (i.e. files without any references to external resources).
--preserve _fontnames		Use the font/font-family naming scheme as obtained from the source file. This works best with --nofonts enabled.
--nounicode		Disables mapping of text to public Unicode region. Instead text will be converted using a custom encoding. By default, all text is mapped to Unicode.
-b or --box	-b media	Specifies the page box/region to use for clipping. Possible values are: ▪ **media** ▪ **crop** ▪ **trim** ▪ **bleed** ▪ **art** The default is page crop region.
-c or --crop	-c 216,522,330,600	User definable crop box to be used as a top level clip region in the output SVG. By default, the clip region is identical to currently selected page 'box'.
--noclip		Disables page clipping. Any content outside of page boundaries will be visible. By default, all pages are clipped using the crop region for the page.
--noannots		Disables conversion of form fields and annotations.
--noxmldoc		Disables generation of the XML wrapper document.
--thumbsize	--thumbsize 150	The dimension of thumbnail image in pixels. By default, PDF2SVG will generate 150x150 thumbnails.
--nothumbs		Disables generation of thumbnail images.
--flatten	--flatten off	Used to reduce some PDF content to a simple background image. While flattening tries to preserve vector text, some text might be flattened, especially in simple mode. Options are: OFF, disable flattening. FAST, will convert content deemed complex to a background image, while trying to preserve vector text, and keeping file size down. SIMPLE, reduces the PDF to two layers; a RGB background image layer and an overlapping vector text layer. Default is FAST.
--flatten_ threshold	--flatten_threshold keep_most	Used to control how precise or relaxed text flattening is. Some text can be preserved (not flattened to image) at the expense that the output might not be

		exactly the same as the input. VERY_STRICT, render (flatten) any text that is clipped or occluded. STRICT, render text that are marginally clipped or occluded. DEFAULT, render text that are somewhat clipped or occluded. KEEP_MOST, only render text that are seriously clipped or occluded. KEEP_ALL, only render text that are completely occluded, or used as a clipping path.
--individual_char_placement		Some viewers do not support the default text positioning correctly. This option works around this issue to place text correctly, but produces verbose output. This option will override --remove_char_placement.
--remove_char_placement		Disable the output of character positions. This will produce slightly smaller output files than the default setting, but many viewers do not support the output correctly
--noglyphhex		Removes hex escape strings for the unicode attribute of glyph. This is useful when using the --nounicode option and the resulting SVG will be displayed in a web browser as it prevents mismapping of charcodes to glyphs. This option is only applied when fonts are embeded.
--noprompt		Disables any user input. By default, the application will ask for a valid password if the password is incorrect.
-p or --pass	-p "my pass"	The password for the input file. Not required if the input document is not secured.
--extension	--extension “.pdf”	The default file extension used to process PDF documents. The default is “.pdf”.
-h or --help		Print a listing of available options.
-v or --version		Print the version information.
--verb	--verb 2	Set the verbosity level. Valid parameter values are 0, 1, and 2. The higher number results in more feedback. The default is 1.

3.3 Basic Usage

3.3.1 How to save converted files in a given folder?

By default, PDF2SVG saves converted files in the current working folder. To specify another output location, use the ‘-o’ (or --output) parameter. For example:

```
pdf2svg –o “c:\My Output” 1.pdf 2.pdf 3.pdf
```

Note: If the specified path does not exist, PDF2SVG will attempt to create the necessary folders.

3.3.2 How can I control the output names of generated files?

By default, PDF2SVG creates a separate SVG file for every page in the document. The output filename is constructed using the name of the input PDF file, the page number, and appropriate file extension (i.e. svg or svgz). For example, the following command-line generates a sequence of SVG files starting with mydoc_1.svg, mydoc_2.svg, etc.:

```
pdf2svg mydoc.pdf
```

PDF2SVG allows output filename customizations using the ‘--prefix’ and ‘--digits’ options. For example, the following command-line generates a sequence of SVG files starting with newname_0001.svg, newname_0002.svg, etc.:

```
pdf2svg --prefix newname --digits 4 mydoc.pdf
```

The ‘--digits’ parameter specifies the number of digits used in the page counter portion of the output filename. By default, new digits are added as needed, but the ‘prefix’ parameter could be used to format the page counter field to a uniform width (e.g. myfile0001.svg, myfile0010.svg, instead of myfile_1.svg, myfile_10.svg, etc).

To avoid any ambiguities in file naming, the prefix option should be used only for conversion of individual documents.

3.3.3 How do I create compressed SVG (SVGZ)?

To create compressed SVG (SVGZ), use ‘--svgz’ as one of the command-line options. This option will instruct PDF2SVG for compress SVG using GZIP compression and to generate output files with the ‘svgz’ extension. For example,

```
pdf2svg --svgz in.pdf
```

3.3.4 How do I produce stand-alone SVG?

Some PDF documents use many small bitmaps to represent text or patterns. In this case, the converted SVG document will reference hundreds of external images. You may choose to embed these images within the SVG document using the ‘--embedimages’ or (‘-i’) option.

You can also embed the fonts in each page that uses the font, by using '--embedfonts' option.

By embedding images, and fonts, it is possible to create self contained SVG files (i.e. files without any references to external resources). Although it is sometimes desirable to create self contained files, this option can result in files that are slower to render in some viewers. The files with embedded images and/or fonts may also be slower to download over the Net, and because images can't be shared among different pages the total file size for the entire document may increase.

3.3.5 How do I open a password protected PDF?

PDF2SVG will, without user intervention, decrypt and convert documents secured with a master/owner password. If the document is secured using a user (or file open) password, PDF2SVG will prompt you to enter the password.

For unattended conversion, the password can also be specified directly on the command-line using the '-p' (or --password) option. For example:

```
PDF2SVG -p secret secured.pdf
```

The above command line will convert PDF to SVG and will use the provided password ('secret') to open the secured document (i.e. 'secured.pdf').

Note: PDF2SVG supports all standard security options available in PDF, including 40 and 128 bit RC4 encryption, Crypt filters, and 128 AES (Advanced Encryption Standard) encryption.

3.3.6 Why is PDF2SVG generating page thumbnails and the XML summary document?

By default, PDF2SVG generates a bitmap thumbnail for each converted SVG and one XML summary document for the entire document. Image thumbnails can be used for quick preview of SVG documents, whereas XML summary document could be used to create HTML files that wrap SVG files in a web ready eBook. XML summary document can also be used for content repurposing, navigation, and indexing.

PDF2SVG lets you control the dimensions of thumbnail images using '--thumbsize' parameter. The following command-line will generate 512x512 pixel image thumbnails for every page in the document:

```
pdf2svg --thumbsize 512 in.pdf
```

To disable generation of thumbnail images, use the '--nothumbs' option. Similarly, to disable generation of wrapper XML, use the '--noxmldoc' switch.

3.3.7 How do I specify which pages to convert?

By default, PDF2SVG will convert all pages. You can specify a subset of pages to convert using the '-a' or '--pages' options. For example:

```
pdf2svg -a 1,3,10 in.pdf
```

will convert only pages 1, 3, and 10. Please note that PDF2SVG assumes that all pages are numbered sequentially starting from page 1.

To specify a range of pages, use dash character between numbers. For example:

```
pdf2svg -a 1,10-20,50- in.pdf
```

will convert the first page, pages in the range from 10 to 20 and all pages starting with page 50 to the last page in the document.

All even pages can be selected using the ‘e’ (or ‘even’) string. For example, the following line converts all even pages:

```
pdf2svg --pages even in.pdf
```

Similarly, odd pages can be selected using the ‘o’ (or ‘odd’) string. The following line converts all odd pages in the document and every page in the range from 100 to the last page:

```
pdf2svg --pages odd,100- in.pdf
```

3.3.8 How do I batch-convert files?

PDF2SVG supports batch conversion of many PDF files in a single pass. To convert all PDF files in a given folder(s) you can use the following syntax:

```
pdf2svg myfolder1
```

The ‘--subfolders’ option can be used to recursively process all subfolders. For example, the following line will convert all documents in ‘myfolder1’ and ‘myfolder2’ as well as all subfolders:

```
pdf2svg --subfolders myfolder1 myfolder2
```

By default, PDF2SVG will convert all files with the extension ‘.pdf’. To select different files based on the extension use the ‘--extension’ parameter. For example, to convert all PDF documents with a custom extension ‘.blob’, you could use the following line:

```
pdf2svg --extension .blob --subfolders myfolder1
```

3.3.9 How can I show/hide crop marks or the trim region?

A PDF page can define as many as five separate boundaries to control various aspects of the imaging process:

- The media box defines the boundaries of the physical medium on which the page is to be printed. It may include any extended area surrounding the finished page for bleed, printing marks, or other such purposes. It may also include areas close to the edges of the medium that cannot be marked because of physical limitations of the output device. Content falling outside this boundary can safely be discarded without affecting the meaning of the PDF file.
- The crop box defines the region to which the contents of the page are to be clipped (cropped) when displayed or printed. Unlike the other boxes, the crop box has no defined meaning in terms of physical page geometry or intended use; it merely imposes clipping on the page contents. The default value is the page’s media box.
- The bleed box defines the region to which the contents of the page should be clipped when output in a production environment. This may include any extra bleed area needed to accommodate the physical limitations of cutting, folding, and trimming equipment. The default value is the page’s crop box.
- The trim box defines the intended dimensions of the finished page after trimming. It may be smaller than the media box to allow for production related content, such as printing instructions, cut marks, or color bars. The default value is the page’s crop box.

- The art box defines the extent of the page’s meaningful content (including potential white space) as intended by the page’s creator. The default value is the page’s crop box.

By default, PDF2SVG uses the page crop box as a default clip region. Different page regions can be selected as the default clip region using the -b (or --box) parameter. For example, the following line will instruct PDF2SVG to use the media box for rasterization:

```
pdf2svg --box media in.pdf
```

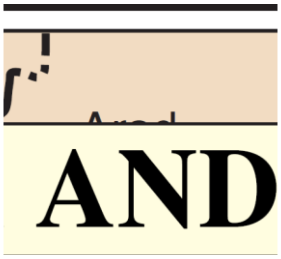
3.3.10 What quality can I expect from the SVG output?

Since PDF2SVG always attempts to maintain the original document appearance, the vast majority of output files will successfully preserve the appearance and quality of the original documents. Occasionally, there will be elements that can’t be accurately converted to SVG.

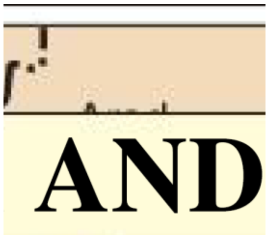
There is the option to 'flatten' content. The default when converting to SVG is 'fast' flatten mode, which will try to generate a PDF that renders faster on limited speed/memory devices, and will render content that is deemed too complex for SVG viewers to handle. This can include changing color spaces, to converting complex paths to an image. You can also turn this off to help ensure that as much content is preserved as is.

```
pdf2svg --flatten off doc.pdf
```

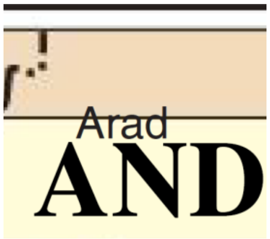
If you want to flatten, there is a way to control how much is flattened, or not, by adjusting the flatten threshold. The images below demonstrate how --flatten_threshold can affect flattening. Notice that the large text is never flattened, only the text occluded by the rectangle.



Original PDF



--flatten_threshold default



--flatten_threshold keep_all

3.4 General Usage Examples

Example 1. The simplest command line: Convert PDF to SVG.

Notes:

- The ‘-o’ (or --output) parameter is used to specify the output folder. If this option is not specified, all converted SVG-s will be stored in the current working folder.

```
pdf2svg -o outfolder in.pdf
```

Example 2. Convert PDF to compressed SVG and without thumbnails and XML summary.

Notes:

- The ‘--noxmldo’ option disables generation of thumbnails.
- The ‘--nothumbs’ option disables generation of thumbnails.
- The ‘--svgz’ option instructs PDF2SVG to compress SVG using GZIP compression.
- The ‘--verb’ option instructs PDF2SVG to output more feedback in the console window.

```
pdf2svg --output test_out/ex2 --svgz --nothumbs --noxmldoc --verb 3 in.pdf
```

Example 3. Convert a password protected file to SVG.

Notes:

- The ‘-p’ (or --pass) parameter is used to specify the password (i.e. ‘secret’) required to open the encrypted document.
- The ‘--pages’ option instructs PDF2SVG to convert only the first page.

```
pdf2svg -p secret -o ex3 --nothumbs --noxmldoc --pages 1 secret.pdf
```

Example 4. Convert all PDF document in a given folders to stand alone SVG.

Notes:

- The ‘--bbox’ parameter instructs PDF2SVG to use media box for clipping instead of crop box, which is the default.
- The ‘--embedimages’ option (or –i in the short form) instructs PDF2SVG to embed all images as inline resources. This option produces stand-alone SVG files (i.e. SVG files without external references).

```
pdf2svg -o OUT --embedimages --box media "My Folder1" "My Folder2"
```

3.5 Batch Processing and the Use of Wildcards

PDF2SVG supports processing of multiple input documents in the same run. For example, it is possible to specify multiple PDF folders and PDF2SVG will automatically process all PDF documents matching a given file extension. For example, the following command-line will process all PDF documents in folders ‘test1’ and ‘test2’

```
c:\>pdf2svg -o c:/output_folder c:/test1 c:/test2
```

Wildcard characters can also be used to process multiple input files.

For example, if a directory contains the following PDF documents:

```
C:\test1 >dir
 Directory of C:\test1
01/04/2007  03:35 PM    <DIR>          .
01/04/2007  03:35 PM    <DIR>          ..
05/21/2004  02:27 PM           A1.pdf
05/03/2005  09:38 AM           A2.pdf
05/20/2003  08:46 AM           B1.pdf
05/15/2003  12:50 PM           B2.pdf
```

To process all PDF documents in this folder, you could specify:

```
c:\>pdf2svg -o c:/output_folder c:/test1/*.pdf
```

To process all PDF documents staring with ‘A’, you could specify:

```
pdf2svg-o c:/output_folder c:/test1/A*.pdf
```

Or to process all PDF documents ending with ‘1’, you could specify:

```
pdf2svg -o c:/output_folder c:/test1/*1.pdf
```

You can use either of the two standard wildcards — the question mark (?) and the asterisk (*) — to specify filename and path arguments on the command line.

The wildcards are expanded in the same manner as operating system commands. (See your operating system user's guide if you are unfamiliar with wildcards). Enclosing an argument in double quotation marks (**" "**) suppresses the wildcard expansion. Within quoted arguments, you can represent quotation marks literally by preceding the double-quotation-mark character with a backslash (****). If no matches are found for the wildcard argument, the argument is passed literally.

3.6 Exit Codes

To provide additional feedback, PDF2SVG returns exit codes after completing processing. The exit codes can be used to provide user feedback, for logging etc. This is particularly important for applications running in an unattended environment.

The following table lists possible exit codes and their description:

Exit Code	Description
0	All files converted successfully.
1	Unspecified error.
2	Bad license key.
3	Failed to create output directory.
4	Failed to read the input document.
5	The PDF password is incorrect.
6	Conversion error.
7	Failed to connect to server.

All codes other then '0' indicate that there was an error during the conversion process.

To get detailed information on an error, set the *--verb* parameter to 2.

The following illustrates a sample Windows batch script that processes exit codes:

```
@echo off rem convert all PDF files in 'data' folder

pdf2svg data
if errorlevel 1 goto inputerr
if errorlevel 2 goto passwd
if errorlevel 3 goto converr
if errorlevel 4 goto othererror
if errorlevel 0 goto exit

:passwd
echo Document is protected. Need a valid password to open the document.
goto exit

:inputerr
echo No input files specified.
goto exit

:converr
echo A file conversion error was encountered.
goto exit

:othererror
echo An error encountered during processing.
goto exit

:exit
```

3.7 Frequently Asked Questions

3.7.1 Why do conversions stop working after entering API key and secret (Error Code 7)?

PDF2SVG returns with error code 7 if connection with PDFTron servers wasn't established. To help identifying what's the issue you can run PDF2SVG with a "--verb 2" option, which would print additional information to the command line. These are the error messages you will see if something goes wrong:

1) Can’t establish a connection due networking error; Check your connectivity to the internet and firewall settings.
2) Credentials provided for authentication are incorrect. Make sure you are subscribed to pay as you go plan. You can check your subscription plan and credentials at api.pdftron.com.
3) Server is not responding. If the error persists contact support@pdftron.com.
4) Server is not recognizing a conversion. Please contact support@pdftron.com.

3.7.2 What is SVG?

SVG (Scalable Vector Graphics) , developed by a working group of the World Wide Web Consortium (W3C), is an open-standard vector graphics format for describing two-dimensional graphics in XML (Extensible Markup Language).

SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects. SVG drawings can be interactive and dynamic. Animations can be defined and triggered either declaratively (i.e., by embedding SVG animation elements in SVG content) or via scripting.

3.7.3 Why are some text rasterized (no longer vector)?

This can happen for a few different reasons.

First is if other content around the text may have been flattened, and this flattening might have forced the text to be flattened also. Try --flatten off to verify if it is related to the flattener. If true, then modifying the --flatten_threshold option might allow you to produce a better SVG output.

Another reason, is with the flattener on, Type3 fonts will be rasterized. Turning --flatten off will preserve the Type3 font as an SVG font.

3.7.4 How do I create an SVG eBook?

In PDF2SVG versions prior to version 3.0, HTML eBook wrapper generation was part of the conversion process.

With PDF2SVG V3.0 and higher, there is added flexibility because the conversion process generates an XML summary document that can be transformed using XSLT (or another XML processor) to a customized HTML, or to another file format. For a simple example of how to generate an HTML eBook wrapper, see the XSLT folder in your PDF2SVG directory.

3.7.5 Is it possible to customize the "look-and-feel" of the SVG eBook (HTML wrappers)?

Because PDF2SVG generates the XML document outline without any formatting elements, you have full control over the "look-and-feel" of the HTML wrapper. The transformation and formatting from XML to HTML can be specified using XSLT or another XML processor.

3.7.6 Why are some fonts not accurate?

In order to preserve fonts, you need to enable font embedding. Conversions with font embedding will produce slightly larger SVG documents, but the text should display correctly on different systems.

In some cases the original font may neither be available on the system nor in the PDF. In this case, PDF and SVG viewers need to perform font substitution. Because SVG viewers use different font substitution procedures, the results may differ from one viewer to another. To avoid font substitution errors, make sure to create PDF documents with embedded fonts.

3.7.7 Why is a white space separating neighboring pictures?

In some cases, SVG viewers that support anti-aliased rendering produce line/space artifacts at neighboring picture elements (e.g. for image tiles or polygons sharing common edges).
These artifacts are not a byproduct of PDF2SVG conversion, but are produced due to anti-aliased rendering in the SVG viewer. To eliminate anti-aliasing artifacts you can try to disable 'high-quality' rendering option in your SVG viewer.

3.7.8 Can I integrate PDF2SVG with my client/server application?

PDF2SVG has a simple-to-use API that can be easily integrated into third-party client and server-based applications. PDF2SVG is available as a .NET component or as a cross-platform C++ library. For more information on licensing the PDF2SVG SDK, please contact a PDFTron representative at info@pdftron.com.

3.7.9 Does PDF2SVG have any dependencies on third party components/software?

PDF2SVG is a completely stand-alone application and does not include any dependencies on third-party components or software.

3.8 XML Summary Document

This section describes the XML Summary Document that can be generated using PDF2SVG and its potential use in various applications.

By default PDF2SVG generates an XML Summary Document for every PDF document. The XML Summary Document contains document-level information that is not part of SVG files that describe individual pages. The information includes general information about the document (such as author, subject, title, keywords), as well as a listing of document parts and relationships such as pages, thumbnails, annotations, and bookmarks.

The following is a sample XML snippet generated by converting this user manual to SVG:

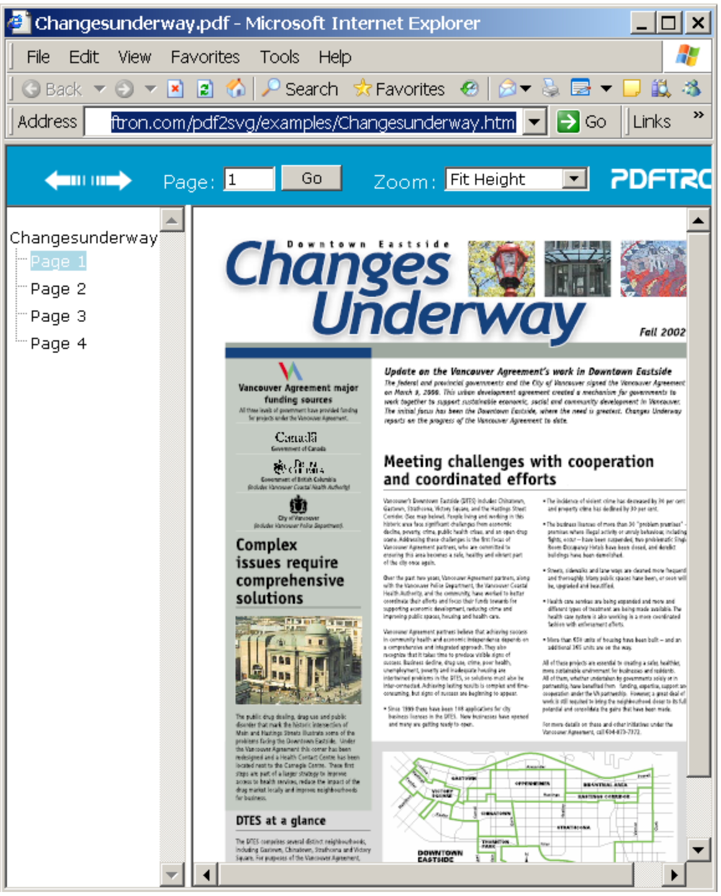
```
<?xml version="1.0"?>
<!-- Generator: PDFTron PDF2SVG Converter -->
<doc name="1" ext="svg">
 <info>
  <title>PDFTron PDF2SVG User Manual</title>
  <author>PDFTron Systems</author>
  <subject>PDFTron PDF2SVG User Manual</subject>
  <keywords></keywords>
  <creator>Acrobat PDFMaker 7.0.7 for Word</creator>
  <producer>Acrobat Distiller 7.0.5 (Windows)</producer>
 </info>
 <pages>
  <page id="1" href="1_1.svg" width="612.0000" height="792.0000">
   <thumb href="1_1_thumb.jpg"/>
  </page>
  <page id="2" href="1_2.svg" width="612.0000" height="792.0000">
   <thumb href="1_2_thumb.jpg"/>
  </page>
  ...
 </pages>
 <bookmarks>
   <bookmark title=" 2. Installing and Uninstalling PDF2SVG" open="true"
goto="7" href="1_7.svg">
    <bookmark title="2.1 PDF2SVG Installation" open="false" goto="7"
href="1_7.svg"/>
    <bookmark title="2.2 Demo Version Installation" open="false" goto="7"
href="1_7.svg"/>
    <bookmark title="2.3 Uninstalling PDF2SVG" open="false" goto="7"
href="1_7.svg"/>
   </bookmark>
   ...
 </bookmarks>
</doc>
```

Most of the elements and attributes are self explanatory. The ‘info’ element lists document information properties, the ‘pages’ element lists all ‘page’ elements that are part of the high level ‘document’, and the ‘bookmarks’ element specifies the outline tree that can be used for quick navigation between pages.

The summary document can be used as a map of the abstract document that contains many SVG files representing document pages, as well as outline tree and annotations describing how different document parts are related.

In most cases, the summary document is further consumed by an XML consumer/processor (e.g. XML DOM/SAX Library or XSLT). For example, an application may read XML summary to create database records for archiving purposes. Another application may implement interactive navigation through SVG pages using the document outline.

Yet another example of the XML wrapper consumer is an eBook generator that converts the XML Summary Document to HTML. The generated HTML would wrap converted SVG files and would provide web-based eBook interface for navigation between different pages, including bookmark tree, thumbnail index, etc. The end result would look like what is illustrated in the following figure:


Changesunderway.pdf - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Search
Favorites
Address
Go
Links
Page:
Go
Zoom:
PDFTRC
Changesunderway
Page 1
Page 2
Page 3
Page 4
Downtown Eastside
Changes Underway
Fall 2002
Update on the Vancouver Agreement's work in Downtown Eastside
Vancouver Agreement major funding sources
Meeting challenges with cooperation and coordinated efforts
Complex issues require comprehensive solutions
DTES at a glance

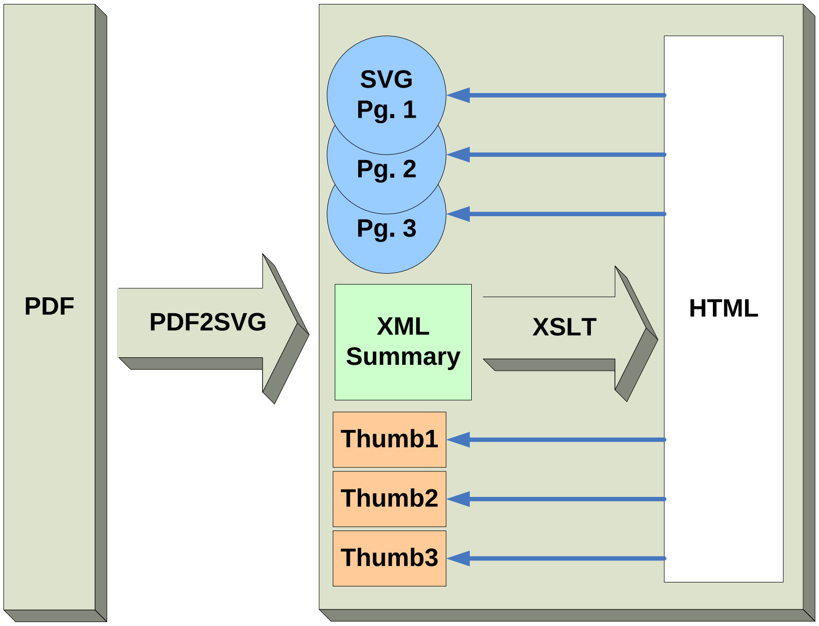
Address: ftron.com/pdf2svg/examples/Changesunderway.htm

Page: 1

Zoom: Fit Height

Figure: SVG wrapped in an HTML web-browser eBook.

The process used to create HTML eBook wrapping converted SVG-s is illustrated in the following figure:



Using PDF2SVG, a PDF document is converted to a set of SVG images and their thumbnails, as well as the XML Summary Document. The fastest way to create HTML wrappers around SVG is using XSLT. XSLT is a very simple language for transforming XML documents. A simple XSLT transform may look as follows:

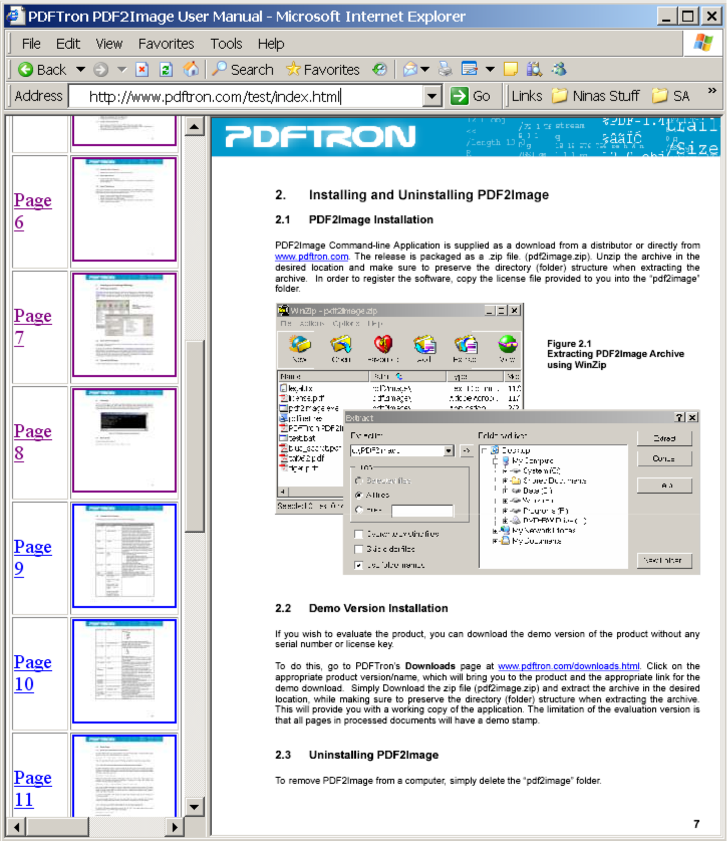
```
<?xml version='1.0'?>
<xsl:stylesheet version='1.0'
xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>
  <xsl:output method='html' indent='yes' doctype-public='-//W3C//DTD HTML
3.2 Final//EN'/>
  <xsl:template match='/'>
    <HTML>
      <HEAD>
        <TITLE>HTML SVG Wrapper</TITLE>
      </HEAD>
      <BODY>
          <xsl:apply-templates select='doc/info'/>
          <HR/>
        <xsl:apply-templates select='doc/pages'/>
      </BODY>
    </HTML>
  </xsl:template>

  <xsl:template match='info'>
   <table border="0" cellspacing="0" cellpadding="4">
     <tr><td>Title:</td><td><xsl:value-of select='title'/></td></tr>
     <tr><td>Author:</td><td><xsl:value-of select='author'/></td></tr>
     <tr><td>Subject:</td><td><xsl:value-of select='subject'/></td></tr>
     <tr><td>Keywords:</td><td><xsl:value-of select='keywords'/></td></tr>
     <tr><td>Creator:</td><td><xsl:value-of select='creator'/></td></tr>
     <tr><td>Producer:</td><td><xsl:value-of select='producer'/></td></tr>
  </table>
  </xsl:template>

  <xsl:template match='pages'>
      <TABLE BORDER="1">
    <xsl:apply-templates/>
    </TABLE>
  </xsl:template>

  <xsl:template match='page'>
    <TR>
       <TD><A TARGET="view" HREF="{@href}">Page <xsl:value-of
select='@id'/></A></TD>
       <TD><A TARGET="view" HREF="{@href}"><IMG
SRC="{thumb/@href}"/></A></TD>
    </TR>
  </xsl:template>
</xsl:stylesheet>
```

The above XSLT template will create an HTML page containing general information about the documents such as it title, subject, keywords, etc. The HTML will also contain a thumbnail index of all pages in the document. Clicking on page labels or on thumbnails will open SVG graphics in the right pane of the browser window. The final result would look as follows:



http://www.pdftron.com/test/index.html

2. Installing and Uninstalling PDF2Image

2.1 PDF2Image Installation

PDF2Image Command-line Application is supplied as a download from a distributor or directly from www.pdftron.com. The release is packaged as a .zip file. (pdf2image.zip). Unzip the archive in the desired location and make sure to preserve the directory (folder) structure when extracting the archive. In order to register the software, copy the license file provided to you into the "pdf2image" folder.

Figure 2.1
Extracting PDF2Image Archive using WinZip

2.2 Demo Version Installation

If you wish to evaluate the product, you can download the demo version of the product without any serial number or license key.

To do this, go to PDFTron's **Downloads** page at www.pdftron.com/downloads.html. Click on the appropriate product version/name, which will bring you to the product and the appropriate link for the demo download. Simply Download the zip file (pdf2image.zip) and extract the archive in the desired location, while making sure to preserve the directory (folder) structure when extracting the archive. This will provide you with a working copy of the application. The limitation of the evaluation version is that all pages in processed documents will have a demo stamp.

2.3 Uninstalling PDF2Image

To remove PDF2Image from a computer, simply delete the "pdf2image" folder.

7

To run XSLT transforms you can use your favorite XSLT processor. As a starting point, PDF2SVG distribution comes with a sample project illustrating how to run XSLT transform using Microsoft .NET Framework.

4. Support

4.1 Reporting Problems

If you encounter a problem or question regarding PDFTron PDF2SVG which is not addressed in this manual or on PDFTron's website, please submit a problem report to PDFTron's Support Group at http://www.pdftron.com/reportproblem.html.

When submitting a problem you will be asked to provide the following information:

- Contact details
- Product and Version of the product
- Detailed description of problem
- Problem file(s)
- Whether you have an AMS (Annual Maintenance Subscription)
- Any other information that may be related

4.2 Contact Information

To contact PDFTron directly, please use the contact information below:

Tel: 1-604-730-8989
Fax: 1-604-676-2477

Web site: www.pdftron.com
PDFNet SDK Forum: http://groups.google.com/forum/#!forum/pdfnet-sdk
WebViewer Forum: http://groups.google.com/group/pdfnet-webviewer

Email Contacts:

General Business Inquiries: info@pdftron.com
Sales & Licensing: sales@pdftron.com
Product Support: support@pdftron.com
Professional Services: services@pdftron.com
Website related questions: webmaster@pdftron.com
Press & News: press@pdftron.com