



PDF Forms
Documentation

FyTek, Inc.

Web site: <http://www.fytek.com>

FyTek's PDF Forms

Trademarks

FyTek, FyTek PDF Forms and the FyTek logo are registered trademarks or trademarks of FyTek Incorporated in the United States and/or other countries. Acrobat, Adobe, Adobe PDF and Adobe Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product names, logos, designs, titles, words or phrases mentioned within this publication may be trademarks, servicemarks, or tradenames of FyTek, Inc. or other entities and may be registered in certain jurisdictions including internationally.

FyTek Disclaimer

FYTEK, INC. MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE. Copyright © 2000–2014 FyTek, Inc. All rights reserved. This manual may not be copied, photocopied, reproduced, translated, or converted to any electronic or machine-readable form in whole or in part without prior written approval of FyTek, Inc.

This guide may contain links to third-party websites that are not under the control of FyTek, and FyTek is not responsible for the content on any linked site. If you access a third-party website mentioned in this guide, then you do so at your own risk. FyTek provides these links only as a convenience, and the inclusion of the link does not imply that FyTek endorses or accepts any responsibility for the content on those third-party sites.

Acknowledgments

Software Development: Mike Bernardo

Writing: Mike Bernardo

FyTek, Inc.

P.O. Box 71093

Madison Heights, MI 48071

Introduction

PDF Forms is a program to merge text with a background image or PDF, similar to printing text on top of a pre-printed form. Only the output is a PDF or XPS rather than a paper form. Some benefits of this are:

- Display order, invoice, statements, etc. on the web
- Email documents
- Archive information electronically in PDF or XPS format

You may also use SQL commands to extract the data for your form. Note that only CSV is currently supported on all platforms and Windows versions also support Oracle, MySQL, and DB2. Using SQL commands you can loop through your data and place items on the page where you need them without having a form-ready input file.

Use a scanned copy of your form as the background along with a text file from your system or SQL query to dynamically create a PDF or XPS. The image must be a jpeg using either 256 color grayscale or 24-bit color. Some types of tiffs are supported as well - must be black & white, packbits or CCITT encoded with a single image strip.

You may also use an existing PDF or XPS as the background. Note that only other PDFs can be used as backgrounds for PDF output and other XPS documents can be used as backgrounds for XPS output. You'll need to supply the PDF's user password, if any, when using an encrypted PDF for the background. The PDF should be relatively small since the background PDF will be embedded in the resulting file. A background PDF created from line drawing commands (using FyTek's PDF Report Writer for instance) will look better at a variety of resolutions than a jpeg background.

Some features of this software product are:

- Use existing dot-matrix-ready text without modification
- Input UTF-8 formatted files for Chinese, Japanese, or Korean text
- Extract contents via SQL without creating a file up-front
- Specify text and image position, size and scale
- Apply different font or color settings to any area on the page
- Optionally encrypt and password protect the document
- Use several text sources for one PDF
- Create the background as a layer that can be turned on or off

PDF Forms

Introduction

The text is assumed to be a fixed width (courier) font with spacing in the file so the text can be lined up with any background PDF or image. There is an option to convert variable width fonts to fixed width if you prefer to use a built-in or custom font that happens to be variable width. A form-feed (ASCII 12) character can be used to separate pages or an indicator in the first column of the data.

Using the Executable

The program `pdfforms.exe` is the Windows executable program. It takes as input a text file and optionally a background PDF or image along with parameters for positioning. The syntax is:

```
pdfforms.exe filein.txt fileout.pdf [options]
```

You may also use a parameter file to work with more than file. See the [Parameter File](#) section for more information.

Use "stdout" for the output file name to send the PDF contents to standard output (STDOUT, the terminal or pipe to a file). You may then capture the output or use it to stream to the user's browser.

You may also send input to the program from standard input (STDIN, the keyboard by default). Use a dash as the file name in this case. This option must be the first one passed to the program. For example:

```
pdfforms - myfile.pdf -pdf mybkg.pdf
```

Then type in text and end with a single dot or Ctrl-D (Ctrl-Z on DOS). Using this approach you can also pipe a file into the program from Linux/Unix like this:

```
cat sample.txt | pdfforms - myfile.pdf -pdf mybkg.pdf
```

For some situations (such as passing data from an environment variable) you may want to bypass entering an input file. Use `-none` as the input file name in that case. This option must be the first one passed to the program. The `FORMTEXT` environment variable will be checked when the input file name is set to `-none`. Here's a PHP example where the text is stored in the variable `FORMTEXT`:

```
<?php
function pdfformcall($options = "")
{
    header("Content-Type: application/pdf");
    flush();
    passthru("/path-to-program/pdfforms -none stdout $options");
}
$var = "Page 1 text...\n";
$var .= "Page 2 text...\n";
$var .= "Page 3 text...\n";
putenv("FORMTEXT=$var");
pdfformcall("-pdf mybackground.pdf");
?>
```

Using the Executable

You may use input files encoded in UTF-8 format for multi-byte languages. This screen shot shows the included sample `sample_utf8.txt` with some sample text in various languages.

```
This file is in UTF-8 format.

Use the option -utf8 or the method UTF8 to include the needed font information.

This example requires the Japanese font pack available from Adobe.

You can download it from http://www.adobe.com/products/acrobat/cjkfontpack.html if Reader
does not download it automatically.

Japanese: すべての人間は、生まれながらにして自由であり、
Chinese: 人人生而自由(在)在尊嚴和權利上一律平等。他們賦有理性和良心(心)並應以兄弟關係的精神互相對待。
Korean: 모든 인간은 태어날 때부터 자유로우며 그 존엄과 권리에 있어 동등하다
```

The following lists the various options you may include on the command line.

- `-addendum file.dat|text` Used to provide a list of images (one per page) or pages from the background PDF to show at the end of the output PDF. This can be used for further instructions, disclaimer information, etc. Pass either a file name containing the tagged information or the tagged data itself. See the [Addendum](#) section for the details on the layout.
- `-author text` Sets the document author.
- `-bkglayer text` Used to denote the background PDF or image as a separate layer. Supply a name for the background that the user will see in the viewer. Users can selectively turn the background on or off. Requires Acrobat or Acrobat Reader 6.0 or higher to use. Users of earlier versions of Acrobat or Acrobat Reader will not be able to turn off the background.
- `-bkgpass password` The owner password for the background PDF. This is only needed when the background PDF is encrypted. The owner (permissions) password is needed to decrypt the PDF in order to use it in the background.

Options

-c1	Used to specify the first column contains a vertical print instruction. In this case, the first character is checked to see how to print the line based on the following: 0 = Advance 2 lines then print - = Advance 3 lines then print 1 = Start a new page + = Don't advance the line before printing blank = Advance a line and print
-ckjwidth <i>number</i>	Enter the default width for Chinese, Korean, or Japanese characters. The default is 1000. By comparison, the width for courier text is 600 per character so two courier letters take up more space horizontally than a single CKJ glyph. This can be used to align a mix of CKJ and ASCII text on a line. For example, if need each CKJ glyph to take up exactly the same amount of horizontal space as two ASCII letters, set this to 1200. Larger values will result in more spacing between each CKJ glyph.
-color <i>RGB</i>	The color for the text.
-comment <i>text</i>	Assigns a text string to search for at the beginning of lines in the text file. Any lines which match are ignored as comments.
-comp <i>number</i>	Text compression factor (default is 100).
-comp15	Uses a compression algorithm compatible with PDF 1.5 (Acrobat 6.0). PDFs with this form of compression can be viewed only with Acrobat or Reader version 6 or higher. The reduction in size is based on the number and type of objects in the PDF but in general is around 10-20%. Not all PDFs will be reduced by the same percentage factor.
-copies <i>number</i>	Number of copies to print when using the -print or -printer commands. Default is 1. May not work on all systems.
-creator <i>text</i>	Sets the document creator.
-debug <i>debugfile</i>	Used to produce a file of processed queries with timings and number of rows returned.

Options

<code>-decode <i>text</i></code>	This works the same as the <code>-utf8</code> option but here you specify the encoding your file is using. For example, if your file is encoded in GB2312 (Simplified Chinese), you would pass <code>-decode "GB2312"</code> . This is the same as if you used <code>-utf8 "zh-cn,Y"</code> but this option makes it a little clearer as to what the input encoding is. Use GB2312 for Simplified Chinese, big5 for Traditional Chinese, euc-ja for Japanese, or euc-kr for Korean.
<code>-defwidth <i>number</i></code>	Enter the default width for base fonts or an added font. The default for courier is 600. Only necessary if you are trying to match some widths between courier and CKJ fonts or if you want to force a variable width font to fixed width (either a base font or one you are embedding). Larger values will result in more spacing between each character.
<code>-e128</code>	Sets 128-bit encryption method. Files encrypted with 128-bit encryption can only be opened with Acrobat or Acrobat Reader 5.0 or above. The default encryption is 40-bit which works with Acrobat and Acrobat Reader 4.0 and above.
<code>-endchars <i>number</i></code>	Number of characters to remove from the end of each line.

PDF Forms

Options

<code>-font <i>number</i></code>	<p>Font to use (default is 1).</p> <ul style="list-style-type: none">1 - Courier2 - Courier Bold3 - Courier Italics4 - Courier Bold-Italics5 - Helvetica6 - Helvetica Bold7 - Helvetica Italics8 - Helvetica Bold-Italics9 - Times Roman10 - Times Roman Bold11 - Times Roman Italics12 - Times Roman Bold-Italics13 - Symbol14 - Zapf Dingbats15 - 3 of 9 Barcode16 - UPC Size 117 - UPC Size 218 - UPC Size 3 <p>Note that fonts 1-4 are fixed width. The rest are variable width. Spacing will generally not work out properly using font 5 or higher unless you are plotting each section or word separately. You may use the <code>-defwidth</code> option to force a variable width font to fixed width.</p>
<code>-fontfile <i>path-file</i></code>	<p>A TrueType or Type1 font file to embed and use for the text. This should be a fixed width font to keep data lined up on the form. Or use the <code>-defwidth</code> option to force the font to be fixed width. Do not use the <code>-font</code> option with this option. You may place the codepage to use after the font separated by a colon. Must be a codepage that is included in the TrueType font. For example: <code>-fontfile arial.ttf:1250</code>. Currently, the other codepages supported by PDF Forms are:</p> <ul style="list-style-type: none">1250 (Central European)1251 (Cyrillic)1253 (Greek)1254 (Turkish)1255 (Hebrew)1256 (Arabic)
<code>-force</code>	<p>Turns off the prompt to overwrite the output file if it already exists.</p>
<code>-guioff</code>	<p>Suppresses the dialog window that shows the current build progress.</p>

Options

-img <i>image.jpg</i>	Background image file. Use quotes around the name if you have spaces in the path or file name.
-imgleft <i>number</i>	Left position from page edge for the image in inches or units (default is 0).
-imglrmargin <i>number</i>	Left and right margin for the image in inches or units (default is 0).
-imgrect <i>x1,y1,x2,y2</i>	The bounding rectangle for the image in inches or units. For example, use "1,1,7.5,10" to fit an image on an 8.5 x 11 page with a 1 inch margin all around. The image will be scaled to fit. You do not need to pass -imgleft, -imgtop, -imglrmargin or -imgtbmargin when using this option.
-imgtbmargin <i>number</i>	Top and bottom margin for the image in inches or units (default is 0).
-imgtop <i>number</i>	Top position from page edge for the image in inches or units (default is 0).
-infile <i>path-file</i>	Used to set the input file when using a parameter file . This allows a different input file to be used than the one specified in the parameter file without modifying the parameter file.
-ini <i>path-file</i>	Configuration file containing executable options to pass in. The commands may be all on one line or on multiple lines. Any of the options in this section may be used. The file format is nothing more than the options as you would type them on the command line. For example: <i>-units in -font courier</i> .
-keeppb	Keeps the page break line. Use this option if you're missing an extra line at the top of pages once they break. Some files may include a page break character and immediately follow with the first line of the new page without a line feed.
-keywords <i>text</i>	Sets the document keywords.
-kr	Keep image aspect ratio when scaling. Large images will be scaled to fit the page. This option prevents the image from being scaled more in one direction than the other.
-linspace <i>number</i>	Line spacing in 1/72 of an inch (default is the text point size + 1).

Options

-mail	Opens the user's email program to a composition window with the newly created PDF attached. May not work with all email programs. The default settings for to, cc and bcc addresses as well as the subject and body are taken from the options described below. Only for Windows based systems.
-mailauth <i>text</i>	The authentication protocol to use. The default is LOGIN but you may use NTLM. Only needed if the mail server requires authentication. (SMTP only)
-mailauthid <i>text</i>	The user name used to log into the server. Only needed if the mail server requires authentication. (SMTP only)
-mailauthpwd <i>text</i>	The password for the user name used to log into the server. Only needed if the mail server requires authentication. (SMTP only)
-mailbcc <i>text</i>	The address(es) to BCC (blind carbon-copy) the email to. Must be an address in the form of name@somecompany.com. Separate multiple addresses with a comma.
-mailbody <i>text</i>	The body text of the email. Enclose in quotes. This may also be a file name. If so, the contents of the file will be used as the body. Use a \n for a new line when the body is entered using this option. You may also send HTML formatted body text. Put the <HTML> tag as the first line of the body text and it will be sent as HTML rather than plain text. Avoid using references to other local files in the HTML body, such as images, as they will not be sent with the message. You may use images with a web location as the source however.
-mailcc <i>text</i>	The address(es) to CC (carbon-copy) the email to. Must be an address in the form of name@somecompany.com. Separate multiple addresses with a comma.
-mailfakecc <i>text</i>	The CC address to show for the email. The default is the CC address(es). (SMTP only)
-mailfakefrom <i>text</i>	The from address to show for the email. The default is the FROM address. (SMTP only)
-mailfaketo <i>text</i>	The to address to show for the email. The default is the TO address(es). (SMTP only)

PDF Forms

Options

<i>-mailfiles text</i>	A comma separated list of file names to include with the mailing. The path must be fully qualified for each file.
<i>-mailfrom text</i>	The from address for the email. Must be an address in the form of somename@mycompany.com. (SMTP only)
<i>-maillog text</i>	The name of a log file to use for date/time emails were sent as well as any errors. This is optional. (SMTP only)
<i>-mailnodialog</i>	Sends the email via MAPI without a composition window. The user may still receive a dialog box asking if it's OK to send the message on their behalf. The message is send via MAPI if they decide they want to send it. Use the -mailsmtp option instead to send the email via SMTP without user intervention.
<i>-mailpri text</i>	The message priority. Set to either HIGH or LOW. Leave this option off for normal priority. (SMTP only)
<i>-mailreply text</i>	The reply to address for the email. Must be an address in the form of somename@mycompany.com. The default is the FROM address. (SMTP only)
<i>-mailscr file</i>	For Unix/Linux systems where -mail is not available. Specify a script that will receive as a parameter the output PDF file name. Create a script for your operating system that will be used to bring up an email window with an attached PDF.
<i>-mailscrp text</i>	Optional - any extra parameters you want to pass to -mailscr. These are placed on the command line after the output PDF file name.
<i>-mailsmtp text</i>	The SMTP server to use for sending the mail. For example, mail.yourdomain.com. You may also pass the port if necessary after the server name or IP address by adding a colon followed by the port number. For example, 192.168.0.30:25.
<i>-mailsub text</i>	The subject of the email. Enclose in quotes.
<i>-mailto text</i>	The address(es) to send the email to. Must be an address in the form of name@somecompany.com. Separate multiple addresses with a comma.

PDF Forms

Options

-noannotate	Disables add/change of form fields or annotations.
-nochange	Disables changes to the document.
-nocopy	Disables copying of text and/or graphics from the document.
-noprint	Disables printing of the document. To create a PDF with both printing and copying disabled for the user you would run something similar to: <pre>pdfforms filein.frw fileout.pdf -o abc123 -u xyz -noprint -nocopy</pre> The file could only be opened by someone who knows one of the two passwords (abc123 or xyz). Using a password of abc123 gives full access while using the password of xyz does not allow printing or copying of text.
-o <i>password</i>	Sets the owner password for the PDF. If not specified but the user password is, this is set to the user password. Also, when not specified, the owner has only the rights granted when the document was created. So for example, if -noprint was specified, then it is impossible for the owner to print the document.
-open	Automatically opens the newly created PDF. The application used depends on what program is associated with opening PDFs on the machine. Only for Windows based systems.
-openscr <i>file</i>	For Unix/Linux systems where -open is not available. Specify a script that will receive as a parameter the output PDF file name. Create a script for your operating system that will be used to open PDFs.
-openscrp <i>text</i>	Optional - any extra parameters you want to pass to -openscr. These are placed on the command line after the output PDF file name.

PDF Forms

Options

<code>-pagebreakhex <i>text</i></code>	The hex value of the page break character(s) to use. The default is "0C" which is ASCII 12. Or, pass a blank string to have the software look for the character. In this case, PDF Forms will look for a line that might have one or more spaces followed by an ASCII character from 0-19 (except for 10 and 13 which are line feed and carriage return).
<code>-pagefont <i>font,pointsize</i></code>	Sets the font and point size for the page numbering string. Pass the font number or a font file along with the point size. For example, <code>-pagefont 2,9</code> will use font #2 at a point size of 9. Another example is <code>-pagefont c:\fonts\myfont.ttf,9</code> to use a custom font instead.
<code>-pageh <i>height</i></code>	Page height in inches or units (default is 11").
<code>-pagepos <i>x,y</i></code>	The x and y position for the page numbering string. The x position is in inches or units from the left edge of the page. The y position is in inches or units from the top edge of the page. Text is left justified with the baseline of the text at the y position specified. The page number text will render just above the top of the page if <code>-pagestr</code> is used without this option.
<code>-pagerotate <i>number</i></code>	Degrees to rotate each page clockwise. Allowed values are 0 (default), 90, 180 and 270.
<code>-pagestr <i>text</i></code>	Text for page numbering string. You can use <code>%p</code> for the current page and <code>%t</code> for the total number of pages. For example, "Page %p of %t" will print "Page 4 of 59" on page 4 of a PDF with 59 pages. Use two %'s in a row if you're running the program from within a batch file (i.e. "Page: %%p").
<code>-pagew <i>width</i></code>	Page width in inches or units (default is 8.5").
<code>-pdf <i>file.pdf</i></code>	Background PDF. Use quotes around the name if you have spaces in the path or file name. Remember to supply the user password, if there is one, using the <code>-bkypass</code> option when using an encrypted background PDF.
<code>-pdfannots</code>	Keep annotations (like web links) from the background PDF.

PDF Forms

Options

<code>-pdfnewpage</code>	Use with <code>-pdfpagefmt</code> . This option sets the page break to occur when the value set by <code>-pdfpagefmt</code> is found. You should not be using any page break (form-feed) commands in the input file in this case. Each page must start with text that matches the value from <code>-pdfpagefmt</code> .
<code>-pdfpage <i>number</i></code>	Page number from included PDF to use. Default is 1.
<code>-pdfpagefmt <i>text</i></code>	A case-sensitive string to search for in the input file to switch to another page in the background PDF. Use a <code>%d</code> for the page number variable. For example, assume you set <code>-pdfpagefmt "<PAGE=%d>"</code> . A line from the input file containing <code><PAGE=2></code> will use page 2 from the background PDF. Any page which doesn't contain the specified string will use the background page from the <code>-pdfpage</code> option. Each entry must appear on a separate line in the input file by itself as any other data occurring on the same line will be discarded. Use two <code>%</code> 's in a row if you're running the program from within a batch file (i.e. <code>-pdfpagefmt "<PAGE=%d>"</code>).
<code>-pdfpageinc</code>	Used to auto-increment the background PDF page number for each page of the input. The counter will wrap around back to 1 if there are more input pages than background pages. For example, suppose you have a background PDF with 3 pages and an input text file containing 6 pages. In the output PDF, pages 1 and 4 will use background page 1, pages 2 and 5 will use background page 2 and pages 3 and 6 will use background page 3.
<code>-pdfpagethru <i>number</i></code>	The through page number from included PDF to use. Use this when you have 2 or more pages you want to cycle through the background pages. For example, set <code>-pdfpage 2</code> and <code>-pdfpagethru 4</code> to use the three background pages 2, 3, and 4. On page 4 of the output, the cycle will start over with background pages 2, 3, and 4.
<code>-point <i>number</i></code>	Text point size to use (default is 10).
<code>-print</code>	Automatically prints the newly created PDF to the default printer. Must have Acrobat or Acrobat Reader installed. Only for Windows based systems.

PDF Forms

Options

-printdlg	Brings up the Acrobat print dialog box and allows printer selection. This only works when the user has Acrobat or Acrobat Reader associated with PDFs on their machine.
-printer <i>printer device port</i>	Used to print the PDF to the specified printer. There is no print dialog box in this case. This option takes three parameters: printer, device and port. You may pass in just the printer and leave device and port blank to use the default settings for the printer. For example: -printer "Accounting Printer" "HP LaserJet 5" "lpt1:" or -printer "Shipping Printer" You may also use the printer port as the first parameter and leave the last two off if you are using a network printer or don't know the printer name. For example: -printer "\\server\printer"
-printerlist <i>file</i>	Used to generate a list of printers available on the system. This can be used to verify what printers the program finds and what they are called. The list generated is tab separated and includes the printer name, device name and port. Use any of the printer names in the file with the -printer option. This option is only available under Windows systems. Use this option by itself as the program will exit after generating the list
-printscr <i>file</i>	For Unix/Linux systems where -print is not available. Specify a script that will receive as a parameter the output PDF file name. Create a script for your operating system that will be used to print PDFs.
-printscrep <i>text</i>	Optional - any extra parameters you want to pass to -printscr. These are placed on the command line after the output PDF file name.
-producer <i>text</i>	Sets the document producer.

Options

- `-querystr text` Default values to pass in for dynamic processing. This overrides any settings from the environment variable `QUERY_STRING`. Pass name/value pairs with an ampersand in-between. For example, `"$reg='A','B'&$drink=ice%20water"` will set the variable `$reg` to `"A','B"` and `$drink` to `"ice water"`. You can then use `$reg` and `$drink` as part of dynamic SQL queries or other logic. See the [SQL](#) section for details on queries and loop processing.
- `-removecodes ["text"]` Removes printer codes from the input. Any string starting with an escape (ASCII 27) and ending with a space (by default) is deleted from the input. You can optionally pass a text string that can be a regular expression. For example, suppose you want to remove the following code (where `x` is ASCII 27):
`xEx&l1Ox(s0p4102T12/31/2005`
- You want to keep the date at the end since that isn't part of the sequence but there is no space character in between. In this case you can use `-removecodes "T"`. Everything in the escape sequence up to (and including) the letter `T` will be removed. If you need more control, you can pass a Perl regular expression instead. For example, use `"\d+T"` to remove up to a `T` that is preceded by one or more digits.
- `-rows number` The number of rows (lines) per page when no form feed or column one indicator is used in the input file.
- `-scale number` Page scaling factor (default is 100).
- `-signbgcolor text` Optional. The background color for the signature field. See the [Digital Signature](#) section for details.
- `-signderfile text` The path and name of the der-encoded signing certificate. For example, `"c:\keys\mykey.der"`. See the [Digital Signature](#) section for details.
- `-signhide` Optional. Keeps the signature field from showing on the first page of the output PDF. See the [Digital Signature](#) section for details.

Options

<code>-signimg <i>text</i></code>	Optional. The path and name of an image to use for the signature. Set this option to "none" to not place any image in the signature field. See the Digital Signature section for details.
<code>-signkeepratio</code>	Optional. Keep the image x/y scaling ratio when using an image with a signature field. See the Digital Signature section for details.
<code>-signname <i>text</i></code>	Optional. The name of the signature field to use in the PDF. For example, "sig1". See the Digital Signature section for details.
<code>-signpkfile <i>text</i></code>	The path and name of the private key file. For example, "c:\keys\mykey_pk.pem". See the Digital Signature section for details.
<code>-signpwd <i>text</i></code>	Optional. The password for the signing certificate private key. See the Digital Signature section for details.
<code>-signrsn <i>text</i></code>	Optional. The reason for signing the document. Default is "Attestation to the accuracy and integrity of this document". See the Digital Signature section for details.
<code>-signsize <i>text</i></code>	Optional. The fontsize for the text of the signature (0 for no text). Default is 12. See the Digital Signature section for details.
<code>-signssl <i>text</i></code>	The path and file name of the OpenSSL program. For example, "c:\openssl\bin\openssl.exe". See the Digital Signature section for details.
<code>-skippages <i>text</i></code>	Comma separated list of page numbers to skip. Use this option to skip over alignment pages, for example.
<code>-sqldb <i>text</i></code>	The database schema or driver information. See the Database Connection section for details.
<code>-sqldriver <i>text</i></code>	The data source. This is a case-sensitive string. Entries with a * are only available for Windows operating systems. Valid values are: CSV Oracle* mysql* mysqlPP ODBC* XML mysql may give slightly better performance over mysqlPP on Windows systems.

Options

-sqlpwd <i>text</i>	The password for the user when making a database connection. This may also be specified on the QUERY tag.
-sqlquery <i>text</i>	The text for your query commands or a file containing the commands. See the SQL section for details on queries and loop processing.
-sqluser <i>text</i>	User name to use when making a database connection. This may also be specified on the QUERY tag.
-strin <i>file</i>	Used to load in text from the specified tag based file. See the Text Files section for the layout.
-subject <i>text</i>	Sets the document subject.
-textrotate <i>number</i>	Degrees to rotate the text clockwise. Allowed values are 0 (default), 90, 180 and 270.
-title <i>text</i>	Sets the document title.
-txtleft <i>number</i>	Left position from page edge for text in inches or units (default is .5").
-txttop <i>number</i>	Top position from page top for text in inches or units (default is .5").
-u <i>password</i>	Sets the user password for the PDF. The following four options can be used to turn off various features for the user.
-units <i>in cm mm pt</i>	Unit of measure for options using a measurement. Default is "in" for inches. Use cm for centimeters, mm for millimeters or pt for point. One point is 1/72 of an inch. For example, setting this option to "cm" means other options such as -pagew will expect a value in centimeters rather than inches.

Options

- `-untaint number` Untaints file names. Use this on Unix systems if you get errors about "insecure dependency while running setgid" or want to restrict what file names may be used. The parameter takes a number from 1 to 3. A value of 1 allows the least characters and 3 the most. For example, if you don't want to allow files from other directories, use 1. Use 2 if you do allow file names that contain slashes (so directories can be used) or 3 for any character in a file name (such as ! or \$).
- 1 - Only letters and numbers as well as -, _ and @ will be allowed
 - 2 - Same as 1 except also allow \, / and :
 - 3 - Allow all characters
- Note this applies to all files - both input and output.
- `-utf8 language[,convert]` Use this option to treat the input as a UTF-8 encoded file. This is used for multi-byte text such as Chinese, Japanese, or Korean. Pass in the base language code. Enter 'ja' for Japanese, 'zh-tw' for Traditional Chinese, 'zh-cn' for Simplified Chinese, or 'ko' for Korean. You may pass a blank value and the program will attempt to determine. Pass a Y for the convert option if your file requires conversion to UTF-8. For example, to convert ASCII Chinese (GB2312 encoding) to UTF-8 you would pass `-utf8 "zh-cn,Y"`. Also see the `-ckjwidth` and `-defwidth` options.
- `-xps path-file` Specify the output XPS file to create in addition to the PDF. See the [XPS Document](#) section for more information on XPS.
- `-xpsback path-file` The background XPS file to use for the output XPS. Use along with the `-xps` option. See the [XPS Document](#) section for more information on XPS.
- `-xpsrect x1,y1,x2,y2` Use with the `-xpsback` option. The bounding rectangle for the XPS background in inches or units. For example, use "1,1,7.5,10" to fit the XPS background on an 8.5 x 11 page with a 1 inch margin all around. The background will be scaled to fit.

Examples

Here are some examples.

```
pdfforms report.txt report.pdf -img invoice.jpg -font 2  
-txtleft .25 -txttop .25 -imgtop .5
```

```
pdfforms report.txt report.pdf -img invoice.jpg -point 9  
-imgtop .5 -o pwd123 -u pwd999
```

```
pdfforms report.txt report.pdf -pdf backgrnd.pdf -point 9 -o  
pwd123 -u pwd999
```

```
pdfforms report.txt report.pdf -pdf backgrnd.pdf -point 10  
-mailsmtp "mail.xyzcorp.com"  
-mailto "bob@somesite.com,jane@abc.net"  
-mailfrom "reports@xyzcorp.com"  
-mailsub "Current Invoice"  
-mailbody "<HTML><FONT COLOR=#0000FF>See attachment."
```

Using the DLL (Dynamic Link Library)

The file `pdfforms.dll` is the dynamic link library. This file should reside in your Windows or Winnt directory under the `system32` sub-directory. You first must register the DLL on your system (note if you ran the setup program this happens automatically). Do this by running

```
regsvr32 pdfforms.dll
```

You should see a message box that reads:

DllRegisterServer in `pdfforms.dll` succeeded.

Click OK to continue. You are now ready to use the DLL.

The .NET version contains the same methods as the standard DLL. The .NET DLL is named `pdfformsdn.dll`.

You may use input files encoded in UTF-8 format for multi-byte languages. This screen shot shows the included sample `sample_utf8.txt` with some sample text in various languages.

```
This file is in UTF-8 format.

Use the option -utf8 or the method UTF8 to include the needed font information.

This example requires the Japanese font pack available from Adobe.

You can download it from http://www.adobe.com/products/acrobat/cjkfontpack.html if Reader does not download it automatically.

Japanese: 全ての人間は、生れながらして自由で、
Chinese: 人人生而自由(在)在尊嚴和權利上一律平等。他們賦有理性和良心(並)並應以兄弟關係的精神互相對待。
Korean: 모든 인간은 태어날 때부터 자유로우며 그 존엄과 권리에 있어 동등하다
```

The methods of PDF.Forms are:

`buildForm`

The method to call once all of the settings have been made. This method performs the merging of the text with the background and creates the PDF. The return value is 0 if successful, otherwise a negative number is returned. Check the `errmsg` property to get the text of the message.

Methods

SetAddendum (file.dat text)	Used to provide a list of images (one per page) or pages from the background PDF to show at the end of the output PDF. This can be used for further instructions, disclaimer information, etc. Pass either a file name containing the tagged information or the tagged data itself. See the Addendum section for the details on the layout.
SetAuthor (text)	Sets the document author.
SetBkgLayer (text)	Used to denote the background PDF or image as a separate layer. Supply a name for the background that the user will see in the viewer. Users can selectively turn the background on or off. Requires Acrobat or Acrobat Reader 6.0 or higher to use. Users of earlier versions of Acrobat or Acrobat Reader will not be able to turn off the background.
SetBkgPass (password)	The owner password for the background PDF. This is only needed when the background PDF is encrypted. The owner (permissions) password is needed to decrypt the PDF in order to use it in the background.
SetCKJWidth (number)	Enter the default width for Chinese, Korean, or Japanese characters. The default is 1000. By comparison, the width for courier text is 600 per character so two courier letters take up more space horizontally than a single CKJ glyph. This can be used to align a mix of CKJ and ASCII text on a line. For example, if need each CKJ glyph to take up exactly the same amount of horizontal space as two ASCII letters, set this to 1200. Larger values will result in more spacing between each CKJ glyph.

Methods

SetCmds (text)	Used to pass executable command line options into the program. You may use this option instead of or in addition to the normal methods for setting options. For example, you may pass "-units 'cm'" to this method instead of calling SetUnits("cm"). Any of the options listed in the executable command line option set may be used. These will override any options you set with the corresponding method._000011_
SetColOne	Used to specify the first column contains a vertical print instruction. In this case, the first character is checked to see how to print the line based on the following: 0 = Advance 2 lines then print - = Advance 3 lines then print 1 = Start a new page + = Don't advance the line before printing blank = Advance a line and print
SetColor (RGB)	The color for the text.
SetComment (text)	Assigns a text string to search for at the beginning of lines in the text file. Any lines which match are ignored as comments.
SetComp15	Uses a compression algorithm compatible with PDF 1.5 (Acrobat 6.0). PDFs with this form of compression can be viewed only with Acrobat or Reader version 6 or higher. The reduction in size is based on the number and type of objects in the PDF but in general is around 10-20%. Not all PDFs will be reduced by the same percentage factor.
SetCompress (number)	Text compression factor (default is 100).
SetCopies (number)	Number of copies to print when using the SetPrint or SetPrinter methods. Default is 1. May not work on all systems.

Methods

SetCreator (text)	Sets the document creator.
SetDebug (debugfile)	Used to produce a file of processed queries with timings and number of rows returned.
SetDecode (text)	This works the same as the SetUTF8 option but here you specify the encoding your file is using. For example, if your file is encoded in GB2312 (Simplified Chinese), you would pass SetDecode ("GB2312"). This is the same as if you used SetUTF8 "zh-cn", "Y" but this option makes it a little clearer as to what the input encoding is. Use GB2312 for Simplified Chinese, big5 for Traditional Chinese, euc-ja for Japanese, or euc-kr for Korean.
SetDefWidth (number)	Enter the default width for base fonts or an added font. The default for courier is 600. Only necessary if you are trying to match some widths between courier and CKJ fonts or if you want to force a variable width font to fixed width (either a base font or one you are embedding). Larger values will result in more spacing between each character.
SetEncrypt128	Sets 128-bit encryption method. Files encrypted with 128-bit encryption can only be opened with Acrobat or Acrobat Reader 5.0 or above. The default encryption is 40-bit which works with Acrobat and Acrobat Reader 4.0 and above.
SetEndChars (number)	Number of characters to remove from the end of each line.

Methods

SetFontFile (filename)	<p>A TrueType or Type1 font file to embed and use for the text. This should be a fixed width font to keep data lined up on the form. Or use the SetDefWidth option to force the font to be fixed width. Do not use the SetFontNum method with this one. You may place the codepage to use after the font separated by a colon. Must be a codepage that is included in the TrueType font. For example: <i>SetFontFile "arial.ttf:1250"</i>. Currently, the other codepages supported by PDF Forms are:</p> <ul style="list-style-type: none">1250 (Central European)1251 (Cyrillic)1253 (Greek)1254 (Turkish)1255 (Hebrew)1256 (Arabic)
SetFontNum (number)	<p>Font to use (default is 1).</p> <ul style="list-style-type: none">1 - Courier2 - Courier Bold3 - Courier Italics4 - Courier Bold-Italics5 - Helvetica6 - Helvetica Bold7 - Helvetica Italics8 - Helvetica Bold-Italics9 - Times Roman10 - Times Roman Bold11 - Times Roman Italics12 - Times Roman Bold-Italics13 - Symbol14 - Zapf Dingbats15 - 3 of 9 Barcode16 - UPC Size 117 - UPC Size 218 - UPC Size 3 <p>Note that fonts 1-4 are fixed width. The rest are variable width. Spacing will generally not work out properly using font 5 or higher unless you are plotting each section or word separately. You may use the SetDefWidth option to force a variable width font to fixed width.</p>

Methods

SetFormText (text)	Use to pass in the input text rather than create a file and use the SetInFile method. You can call this method multiple times and the text will be appended or pass in the text all at once.
SetImage (filename)	Background image file. Use quotes around the name if you have spaces in the path or file name.
SetImageLeft (number)	Left position from page edge for the image in inches or units (default is 0).
SetImageLRMargin (number)	Left and right margin for the image in inches or units (default is 0).
SetImageTBMargin (number)	Top and bottom margin for the image in inches or units (default is 0).
SetImageTop (number)	Top position from page edge for the image in inches or units (default is 0).
SetImgRect x1, y1, x2, y2	The bounding rectangle for the image in inches or units. For example, use 1,1,7.5,10 to fit an image on an 8.5 x 11 page with a 1 inch margin all around. The image will be scaled to fit. You do not need to set SetImgLeft, SetImgTop, SetImgLRMargin or SetImgTBMargin when using this method.
SetInFile (filename)	The text input file. You may also use a parameter file to work with more than file. See the Parameter File section for more information.
SetInFile2 (filename)	Used to set the input file when using a parameter file . This allows a different input file to be used than the one specified in the parameter file without modifying the parameter file.

Methods

SetKeepPBLine	Keeps the page break line. Use this method if you're missing an extra line at the top of pages once they break. Some files may include a page break character and immediately follow with the first line of the new page without a line feed.
SetKeepRatio	Keep image aspect ratio when scaling. Large images will be scaled to fit the page. This method prevents the image from being scaled more in one direction than the other.
SetKeywords (text)	Sets the document keywords.
SetLineSpacing (number)	Line spacing in 1/72 of an inch (default is the text point size + 1).
SetMail	Opens the user's email program to a composition window with the newly created PDF attached. May not work with all email programs. The default settings for to, cc and bcc addresses as well as the subject and body are taken from the methods described below.
SetMailAuth (text)	The authentication protocol to use. The default is LOGIN but you may use NTLM. Only needed if the mail server requires authentication. (SMTP only)
SetMailAuthID (text)	The user name used to log into the server. Only needed if the mail server requires authentication. (SMTP only)
SetMailAuthPwd (text)	The password for the user name used to log into the server. Only needed if the mail server requires authentication. (SMTP only)
SetMailBCC (text)	The address(es) to BCC (blind carbon-copy) the email to. Must an address in the form of name@somecompany.com. Separate multiple addresses with a comma.

Methods

SetMailBody (text)	The body text of the email. This may also be a file name. If so, the contents of the file will be used as the body. Use a \n for a new line when the body is entered using this method. You may also send HTML formatted body text. Put the <HTML> tag as the first line of the body text and it will be sent as HTML rather than plain text. Avoid using references to other local files in the HTML body, such as images, as they will not be sent with the message. You may use images with a web location as the source however.
SetMailCC (text)	The address(es) to CC (carbon-copy) the email to. Must be an address in the form of name@somecompany.com. Separate multiple addresses with a comma.
SetMailFakeCC (text)	The CC address to show for the email. The default is the CC address(es). (SMTP only)
SetMailFakeFrom (text)	The from address to show for the email. The default is the FROM address. (SMTP only)
SetMailFakeTo (text)	The to address to show for the email. The default is the TO address(es). (SMTP only)
SetMailFiles (text)	A comma separated list of file names to include with the mailing. The path must be fully qualified for each file.
SetMailFrom (text)	The from address for the email. Must be an address in the form of somename@mycompany.com. (SMTP only)
SetMailLog (text)	The name of a log file to use for date/time emails were sent as well as any errors. This is optional. (SMTP only)

Methods

SetMailNoDialog	Sends the email via MAPI without a composition window. The user may still receive a dialog box asking if it's OK to send the message on their behalf. The message is send via MAPI if they decide they want to send it. Use the SetMailSMTP method instead to send the email via SMTP without user intervention.
SetMailPriority (text)	The message priority. Set to either HIGH or LOW. Leave this method off for normal priority. (SMTP only)
SetMailReply (text)	The reply to address for the email. Must be an address in the form of somename@mycompany.com. The default is the FROM address. (SMTP only)
SetMailSMTP (text)	The SMTP server to use for sending the mail. For example, mail.yourdomain.com. You may also pass the port if necessary after the server name or IP address by adding a colon followed by the port number. For example, 192.168.0.30:25.
SetMailSubject (text)	The subject of the email. Enclose in quotes.
SetMailTo (text)	The address(es) to send the email to. Must be an address in the form of name@somecompany.com. Separate multiple addresses with a comma.
SetNoAnnote	Disables add/change of form fields or annotations.
SetNoChange	Disables changes to the document.
SetNoCopy	Disables copying of text and/or graphics from the document.
SetNoPrint	Disables printing of the document.
SetOpen	Automatically opens the newly created PDF. The application used depends on what program is associated with opening PDFs on the machine.

PDF Forms

Methods

SetOutFile (filename)	The PDF output file. Leave this method off to have the PDF contents returned to the buildForm method instead. This allows you to stream the PDF contents to a browser.
SetOwner (text)	Sets the owner password for the PDF. If not specified but the user password is, this is set to the user password. Also, when not specified, the owner has only the rights granted when the document was created. So for example, if SetNoPrint was specified, then it is impossible for the owner to print the document.
SetPageBreakHex (text)	The hex value of the page break character(s) to use. The default is "0C" which is ASCII 12. Or, pass a blank string to have the software look for the character. In this case, PDF Forms will look for a line that might have one or more spaces followed by an ASCII character from 0-19 (except for 10 and 13 which are line feed and carriage return).
SetPageFont number file, pointsize	Sets the font and point size for the page numbering string. Pass the font number or a font file along with the point size. For example, SetPageFont 2,9 will use font #2 at a point size of 9. Another example is SetPageFont "c:\fonts\myfont.ttf", 9 to use a custom font instead.
SetPageHeight (number)	Page height in inches or units (default is 11").
SetPagePos x, y	The x and y position for the page numbering string. The x position is in inches or units from the left edge of the page. The y position is in inches or units from the top edge of the page. Text is left justified with the baseline of the text at the y position specified. The page number text will render just above the top of the page if SetPageStr is used without using this method.

PDF Forms

Methods

SetPageRotate (number)	Degrees to rotate each page clockwise. Allowed values are 0 (default), 90, 180 and 270.
SetPageScale (number)	Page scaling factor (default is 100).
SetPageStr (text)	Text for page numbering string. You can use %p for the current page and %t for the total number of pages. For example, "Page %p of %t" will print "Page 4 of 59" on page 4 of a PDF with 59 pages.
SetPageWidth (number)	Page width in inches or units (default is 8.5").
SetPDF (filename)	Background PDF. Use quotes around the name if you have spaces in the path or file name.
SetPDFAnnots	Keep annotations (like web links) from the background PDF.
SetPDFNewPage	Use with SetPDFPageFmt method. This method sets the page break to occur when the value set by SetPDFPageFmt is found. You should not be using any page break (form-feed) commands in the input file in this case. Each page must start with text that matches the value from SetPDFPageFmt.
SetPDFPage (number)	Page number from included PDF to use. Default is 1.
SetPDFPageFmt (text)	A case-sensitive string to search for in the input file to switch to another page in the background PDF. Use a %d for the page number variable. For example, assume you set setPDFPageFmt("<PAGE=%d>"). A line from the input file containing <PAGE=2> will use page 2 from the background PDF. Any page which doesn't contain the specified string will use the background page from the SetPDFPage method. Each entry must appear on a separate line in the input file by itself as any other data occurring on the same line will be discarded.

Methods

SetPDFPageInc	Used to auto-increment the background PDF page number for each page of the input. The counter will wrap around back to 1 if there are more input pages than background pages. For example, suppose you have a background PDF with 3 pages and an input text file containing 6 pages. In the output PDF, pages 1 and 4 will use background page 1, pages 2 and 5 will use background page 2 and pages 3 and 6 will use background page 3.
SetPDFPageThru (number)	The through page number from included PDF to use. Use this when you have 2 or more pages you want to cycle through the background pages. For example, use SetPDFPage 2 and SetPDFPageThru 4 to use the three background pages 2, 3, and 4. On page 4 of the output, the cycle will start over with background pages 2, 3, and 4.
SetPointSize (number)	Text point size to use (default is 10).
SetPrint	Automatically prints the newly created PDF to the default printer. Must have Acrobat or Acrobat Reader installed.
SetPrintDlg	Brings up the Acrobat print dialog box and allows printer selection. This only works when the user has Acrobat or Acrobat Reader associated with PDFs on their machine.

Methods

SetPrinter printer [, device, port]	Used to print the PDF to the specified printer. There is no print dialog box in this case. This method takes three parameters: printer, device and port. You may pass in just the printer and leave off device and port to use the default settings for the printer. For example: SetPrinter "Accounting Printer", "HP LaserJet 5", "lpt1:" or SetPrinter "Shipping Printer" You may also use the printer port as the first parameter and leave the last two off if you are using a network printer or don't know the printer name. For example: SetPrinter "\\server\printer"
SetProducer (text)	Sets the document producer.
SetQUERY_STRING (text)	Default values to pass in for dynamic processing. This overrides any settings from the environment variable QUERY_STRING. Pass name/value pairs with an ampersand in-between. For example, "\$reg='A','B'&\$drink=ice%20water" will set the variable \$reg to "'A','B'" and \$drink to "ice water". You can then use \$reg and \$drink as part of dynamic SQL queries or other logic. See the SQL section for details on queries and loop processing.

Methods

<code>SetRemoveCodes</code> <code>-- or --</code> <code>SetRemoveCodes ("text")</code>	<p>Removes printer codes from the input. Any string starting with an escape (ASCII 27) and ending with a space (by default) is deleted from the input. You can optionally pass a text string that can be a regular expression. For example, suppose you want to remove the following code (where x is ASCII 27):</p> <pre>xEx&l1Ox(s0p4102T12/31/2005</pre> <p>You want to keep the date at the end since that isn't part of the sequence but there is no space character in between. In this case you can use <code>SetRemoveCodes ("T")</code>. Everything in the escape sequence up to (and including) the letter T will be removed. If you need more control, you can pass a Perl regular expression instead. For example, use <code>"\d+T"</code> to remove up to a T that is preceded by one or more digits.</p>
<code>SetRows (number)</code>	<p>The number of rows (lines) per page when no form feed or column one indicator is used in the input file.</p>

Methods

SetSign openssl-path-name, pk-file, der-file [, password [, reason [, name [, image [, kepratio [, size [, bgcolor [, hide]]]]]]]]]	<p>This method is used when signing a PDF that already contains a signature field.</p> <p>"openssl-path-name" is the path to the program OpenSSL.</p> <p>"pk-file" is the path-name of the private key file.</p> <p>"der-file" is the path-name of the der-encoded certificate.</p> <p>"password" is the password for the private key file.</p> <p>"reason" is the reason for signing the document.</p> <p>"name" is the name of the field in the existing PDF to sign or set to ":new" to create a new one.</p> <p>"image" is the path-name of an image to use as the background for the signature.</p> <p>"kepratio" is set to "Y" when using an image for the signature when you want to keep the x/y image scaling ratio.</p> <p>"size" is the point size of the font for the signature (or 0 for no text in the signature).</p> <p>"bgcolor" is the background color for the signature.</p> <p>"hide" set to "Y" to not show the signature field on the first page of the output PDF.</p> <p>See the Digital Signature section for details.</p>
SetSkipPages (text)	<p>Comma separated list of page numbers to skip. Use this method to skip over alignment pages, for example.</p>

Methods

SetSQLDB (text)	The database schema or driver information. See the Database Connection section for details.
SetSQLDriver (text)	The data source. This is a case-sensitive string. Entries with a * are only available for Windows operating systems. Valid values are: CSV Oracle* mysql* mysqlPP ODBC* XML mysql may give slightly better performance over mysqlPP on Windows systems.
SetSQLPwd (text)	The password for the user when making a database connection. This may also be specified on the QUERY tag.
SetSQLQuery (text)	The text for your query commands or a file containing the commands. See the SQL section for details on queries and loop processing.
SetSQLUser (text)	User name to use when making a database connection. This may also be specified on the QUERY tag.
SetStrFileIn (path-file)	Used to load in text from the specified tag based file. See the Text Files section for the layout.
SetSubject (text)	Sets the document subject.
SetTextLeft (number)	Left position from page edge for text in inches or units (default is .5").
SetTextRotate (number)	Degrees to rotate the text clockwise. Allowed values are 0 (default), 90, 180 and 270.
SetTextTop (number)	Top position from page top for text in inches or units (default is .5").
SetTitle (text)	Sets the document title.

Methods

SetUnits ("in" "cm" "mm" "pt")	Unit of measure for methods using a measurement. Default is "in" for inches. Use cm for centimeters, mm for millimeters or pt for point. One point is 1/72 of an inch. For example, setting this method to "cm" means other methods such as SetPageWidth will expect a value in centimeters rather than inches.
SetUser (text)	Sets the user password for the PDF. The following four methods can be used to turn off various features for the user.
SetUTF8 (language[,convert])	Use this method to treat the input as a UTF-8 encoded file. This is used for multi-byte text such as Chinese, Japanese, or Korean. Pass in the base language code. Enter 'ja' for Japanese, 'zh-tw' for Traditional Chinese, 'zh-cn' for Simplified Chinese, or 'ko' for Korean. You may pass a blank value and the program will attempt to determine. Pass a Y for the convert option if your file requires conversion to UTF-8. For example, to convert ASCII Chinese (GB2312 encoding) to UTF-8 you would pass SetUTF8("zh-cn,Y"). Also see the SetCKJWidth and SetDefWidth methods.
SetXPSBackground (filename)	The background XPS file to use for the output XPS. Use along with the SetXPSFile method. See the XPS Document section for more information on XPS.
SetXPSRect x1, y1, x2, y2	Use with the SetXPSBackground method. The bounding rectangle for the XPS background in inches or units. For example, use 1,1,7.5,10 to fit the XPS background on an 8.5 x 11 page with a 1 inch margin all around. The background will be scaled to fit.

Examples

Example

Here is an example of calling the DLL using Visual Basic.

```
Set PDF = CreateObject("PDF.Forms")
PDF.SetInFile ("d:\my documents\report.txt")
PDF.SetOutFile ("output\report.pdf")
PDF.SetImage ("d:\my documents\report.jpg")
PDF.SetFontNum (2)
PDF.SetPointSize (9)
PDF.SetImageLRMargin (.5)
PDF.SetImageTBMargin (.5)
PDF.SetTextLeft (.25)
PDF.SetTextTop (.25)
PDF.SetMailSMTP ("mail.xyzcorp.com")
PDF.SetMailFrom ("reports@xyzcorp.com")
PDF.SetMailTo ("bob@somesite.com,jane@abc.net")
PDF.SetMailSubject ("Current Invoice")
PDF.SetMailBody ("d:\my documents\mailbody.txt")
PDF.buildForm
Set PDF = Nothing
```

An example calling the DLL from PowerBuilder

```
OLEObject PDF
PDF = CREATE OLEObject
li_rc = PDF.ConnectToNewObject("PDF.Forms")
PDF.SetInFile ("d:\my documents\report.txt")
PDF.SetOutFile ("output\report.pdf")
PDF.SetImage ("d:\my documents\report.jpg")
PDF.SetFontNum (2)
PDF.SetPointSize (9)
PDF.SetImageLRMargin (.5)
PDF.SetImageTBMargin (.5)
PDF.SetTextLeft (.25)
PDF.SetTextTop (.25)
PDF.SetMailSMTP ("mail.xyzcorp.com")
PDF.SetMailFrom ("reports@xyzcorp.com")
PDF.SetMailTo ("bob@somesite.com,jane@abc.net")
PDF.SetMailSubject ("Current Invoice")
PDF.SetMailBody ("d:\my documents\mailbody.txt")
PDF.buildForm
```

Examples

The following two examples show how to use with ASP. You may need to set permissions on the DLL for IUSR_<machine_name> and/or IWAM_<machine_name>. These are the user ID's that typically run when using ASP. Also make sure these users have write permission to their temp directories. The DLL will need to unpack some internal files and store them in the temp area. If you don't set the proper permissions you'll get errors back such as "unspecified error" or "access denied" on the Server.CreateObject line.

Here is an example of calling the DLL using ASP streaming the output to the browser. Note the use of the fytek.unicode object. The VBScript will treat the returned PDF stream as Unicode (2-byte characters) which will not work with Response.binaryWrite. The method StrToByte will convert the Unicode string into a single byte string which can be streamed to the browser. This method is in the file fytek.dll included with the installation.

```
<%  
Dim PDF  
Dim binaryData  
Set obj = Server.CreateObject("fytek.unicode")  
Set PDF = Server.CreateObject("PDF.Forms")  
PDF.SetInFile ("d:\my documents\report.txt")  
PDF.SetImage ("d:\my documents\report.jpg")  
PDF.SetFontNum (2)  
PDF.SetPointSize (9)  
PDF.SetImageLRMargin (.5)  
PDF.SetImageTBMargin (.5)  
PDF.SetTextLeft (.25)  
PDF.SetTextTop (.25)  
pdfOut = PDF.buildForm  
binaryData = obj.StrToByte(pdfOut)  
Response.ContentType = "application/pdf"  
Response.binaryWrite binaryData  
set PDF = nothing  
set pdfOut = nothing  
set obj = nothing  
set binaryData = nothing  
>
```


Examples

Here is an ASP example creating the output file then redirecting the browser. This method works well with both Netscape and Internet Explorer. Make sure you have the web user set up with permission to write to whatever output directory you're using.

```
<%  
Dim PDF, RndFile  
Set PDF = Server.CreateObject("PDF.Forms")  
Randomize  
RndFile = "output\" & Int(10000000 * Rnd + 1) & ".pdf"  
PDF.SetInFile ("d:\my documents\report.txt")  
PDF.SetOutFile (RndFile)  
PDF.SetImage ("d:\my documents\report.jpg")  
PDF.SetFontNum (2)  
PDF.SetPointSize (9)  
PDF.SetImageLRMargin (.5)  
PDF.SetImageTBMargin (.5)  
PDF.SetTextLeft (.25)  
PDF.SetTextTop (.25)  
pdfOut = PDF.buildForm  
Response.redirect(RndFile)  
set pdfOut = nothing  
%>
```

Examples

Here is an example of calling the DLL using Progress.

```
DEFINE VARIABLE FormsHandle AS COM-HANDLE.  
DEFINE VARIABLE FormsRes AS CHARACTER.  
/* Creates the COM-HANDLE link */  
CREATE "PDF.Forms" FormsHandle.  
FormsHandle:SetInFile ("c:\temp\text.txt").  
FormsHandle:SetOutFile ("c:\temp\text.pdf").  
FormsHandle:SetImage ("d:\my documents\report.jpg")  
FormsHandle:SetFontNum (2)  
FormsHandle:SetPointSize (9)  
FormsHandle:SetImageLRMargin (.5)  
FormsHandle:SetImageTBMargin (.5)  
FormsHandle:SetTextLeft (.25)  
FormsHandle:SetTextTop (.25)  
/* Open the PDF once built */  
FormsHandle:SetOpen ().  
/* Tells the DLL to build the PDF */  
assign FormsRes = FormsHandle:buildForm.  
RELEASE OBJECT FormsHandle.
```

Here is an example of calling the DLL using ColdFusion.

```
<cfobject type="com" ACTION="create" name="PDF" CLASS="PDF.Forms">  
<CFSET PDF_InFile = "c:\input_path\filename.frw">  
<CFSET PDF_OutFile = "c:\output_path\filename.pdf">  
<CFSET PDF_Image = "d:\my documents\report.jpg">  
<CFSET PDF_Font = "2">  
<CFSET PDF_Point = "9">  
<CFSET PDF_LRMargin = ".5">  
<CFSET PDF_TBMargin = ".5">  
<CFSET PDF_TextLeft = ".25">  
<CFSET PDF_TextTop = ".25">  
<cfscript>  
PDF.SetInFile = PDF_InFile;  
PDF.SetOutFile = PDF_OutFile;  
PDF.SetImage = PDF_Image;  
PDF.SetFontNum = PDF_Font;  
PDF.SetPointSize = PDF_Point;  
PDF.SetImageLRMargin = PDF_LRMargin;  
PDF.SetImageTBMargin = PDF_TBMargin;  
PDF.SetTextLeft = PDF_TextLeft;  
PDF.SetTextTop = PDF_TextTop;  
PDF.buildForm;  
PDF = "Nothing";  
</cfscript>
```

Examples

Here is an example using C.

```
#include <iostream.h>
// The import directive reads the typelib information from the DLL
// and creates pdfforms.tlh and pdfforms.tli, which are included.
// These define wrappers for each of the pdfforms object methods.

#import <pdfforms.dll>
// Using VC++ 5.0 Smart Pointers makes this much easier.
// The parameter string for a method is converted to Unicode, allocated
// and passed as a variant. The wrappers call IDispatch::Invoke
// This is all compatible with MFC (use AfxOleInit instead of CoInitialize, etc.).
int main(int argc, char* argv[])
{
    HRESULT      hr;

    using namespace PDFForms_TypeLib;

    hr = CoInitialize (NULL);    // Initialize COM
    if (SUCCEEDED(hr))
    {
        try    // Each of the following lines can throw exceptions
        {
            // Create the instance and get a pointer to the interface
            IbuildFormPtr pPDF(__uuidof(PDF_Forms));
            pPDF->SetInFile (_bstr_t(L"d:\\my documents\\report.txt"));
            pPDF->SetOutFile (_bstr_t(L"d:\\output\\report.pdf"));
            pPDF->SetImage (_bstr_t(L"d:\\my documents\\report.jpg"));
            pPDF->SetFontNum (_bstr_t(L"2"));
            pPDF->SetPointSize (_bstr_t(L"9"));
            pPDF->SetImageLRMargin (_bstr_t(L".5"));
            pPDF->SetImageTBMargin (_bstr_t(L".5"));
            pPDF->SetTextLeft (_bstr_t(L".25"));
            pPDF->SetTextTop (_bstr_t(L".25"));
            pPDF->SetMailSMTP (_bstr_t(L"mail.xyzcorp.com"));
            pPDF->SetMailFrom (_bstr_t(L"reports@xyzcorp.com"));
            pPDF->SetMailTo (_bstr_t(L"bob@somesite.com,jane@abc.net"));
            pPDF->SetMailSubject (_bstr_t(L"Current Invoice"));
            pPDF->SetMailBody (_bstr_t(L"d:\\my documents\\mailbody.txt"));
            _variant_t outval = pPDF; // Build the PDF file
        }
        catch (_com_error e)
        {
            cout << e.ErrorMessage() << endl;
        }
    }
    else
        cout << "CoInitialize Failed" << endl;

    CoUninitialize();    // Uninitialize COM
    return 0;
}
```

Specifying Colors

The -color option and SetColor methods are used to supply a color for the text. In addition, certain tags such as <FILE> that are used in parameter files have options that take a color as the value. Colors in all these cases may be entered in any of the following ways:

- You may specify the red, green and blue components as values from 0 to 255, separated by a comma. In this case 0,0,0 is black and 255,255,255 is white.
- You may specify the red, green and blue components as a hex string preceded by a # sign. In this case #000000 is black and #FFFFFF is white. If all three red, green and blue components are pairs of the same character, such as #ee33dd, you may shorten to #e3d. When three characters are found after the # sign they are expanded by duplicating each character to make the longer six character code.
- You may specify one of the colors from the table below.

Color	Name	Color	Name
	Black		Green
	Silver		Lime
	Gray		Olive
	White		Yellow
	Maroon		Navy
	Red		Blue
	Purple		Teal
	Fuchsia		Aqua

You may use CMYK (Cyan, Magenta, Yellow and Black) for string colors as well. In this case, you use four numbers for the color setting instead of three. Colors in these cases may be entered as follows:

- You may specify the cyan, magenta, yellow and black components as values from 0 to 255, separated by a comma. In this case 0,0,0,255 is black and 0,0,0,0 is white. Other examples are 0,0,255,0 for yellow and 0,255,255,0 for red.
- You may specify the cyan, magenta, yellow and black components as a hex string preceded by a # sign. In this case #000000FF is black and #00000000 is white. Other examples are #0000FF00 for yellow and #00FFFF00 for red.

Parameter File

A parameter file allows you to specify multiple text files to place on the background or multiple styles for the text based on the PDF background page. It's also a way to store the values needed without passing them on the command line or through the DLL methods. Each text file maintains its own set of parameters for placement, font, color, size, etc. In addition, you may also specify bookmarks, text lines to print and filled rectangles for drawing on the background. Use an @ in front of the input file name to use a parameter file (ex. pdfforms.exe @myparms.txt mypdf.pdf). The program then knows the input contains parameter settings rather than the actual text for the form.

A sample parameter file called params.ini is included with the distribution. The file is tag based and must start with <PDFFORMS> as the first line in the file. After that, you may specify any tags described on the following pages.

Parameter File

<VAR option=value>

The "option" is named after the command line options. For example, <VAR Units=mm> will set the units to millimeters. See the [executable options](#) for a description of each variable. Command line options or DLL methods setting the same value will override the value stored in the parameter file. Here's an example showing some sample settings:

```
<VAR Units=mm>  
<VAR TxtTop=20>  
<VAR TxtLeft=35>  
<VAR PageW=210>  
<VAR PageH=297>  
<VAR Removecodes>
```

Only use one option per VAR tag. You may include as many VAR tags as you need.

<u>Parameter</u>	<u>Description</u>
option=value	The "option" is one of the command line options. Set the "value" to an appropriate value for the option. You may leave out the =value portion for parameters that do not take a value.

Parameter File

**<PRINT
PRINTER=text
DRIVER=text
PORT=text
COPIES=number>**

This tag is used on Windows based systems to specify the parameters for printing. All of the name/values are optional. Using just <PRINT> by itself will simply print a single copy to the default printer.

This registry key is checked for what program is associated with printing PDFs:

HKEY_LOCAL_MACHINE/Software/Classes/AcroExch.Document/shell/print/command/

You may need to set up this key if it does not exist. The key should contain something similar to this (depending on your version of Reader):
"C:\Program Files\Adobe\Reader 8.0\Reader\AcroRd32.exe" /p /h "%1"

The -printerlist option can be used to extract a list of printers available on the system.

Not all combinations will work on all systems so you'll want to test your settings.

<u>Parameter</u>	<u>Description</u>
PRINTER=text	Optional. The printer to use, for example "\\server\printer". The default system printer is used if PRINTER is not set.
DRIVER=text	Optional. The driver to use, for example "HP LaserJet 5".
PORT=text	Optional. The port to use, for example "lpt1:".
COPIES=number	The number of copies to print. Default is 1.

Parameter File

```
<FILE
  SRC="text"
  X=number
  Y=number
  FACE=text
  SIZE=number
  COMP=number
  LINESPACE=number
  COLOR=text
  COLONE>
```

This tag is used to specify an input file. You may have more than one and they are merged in the order found. For example:

```
<FILE SRC="d:\files\file1.txt"
  X=.25 Y=.25 COMP=100 LINESPACE=11 COLOR=0000FF FACE=1 SIZE=8>
<FILE SRC="d:\files\file2.txt"
  X=.5 Y=.25 COMP=100 LINESPACE=9 COLOR=000000 FACE=2 SIZE=7>
<FILE SRC="c:\myfiles\formtxt.txt"
  X=.25 Y=.25 COMP=100 LINESPACE=10 COLOR=009900
  FACE="c:\winnt\fonts\myfont.ttf" SIZE=9>
```

In the above example, text from file1.txt is placed on the background first. Next, file2.txt and finally formtxt.txt. Note the text parameters are all stored here so it's not necessary to pass them to the program using the input parameters (like -txtleft) or the DLL methods (like SetTextLeft).

You may use the -infile command line option or the SetInFile2 DLL method to specify a different input file. The value passed in will override the file in the first FILE tag found. For example, using the data shown above, if -infile "d:\files\filea.txt" was passed on the command line, then d:\files\filea.txt is used in place of d:\files\file1.txt.

The first file determines the number of pages for the PDF. If file1.txt has 12 pages and file2.txt has 15, the resulting PDF will only contain 12 pages and the contents of pages 13 through 15 in file2.txt will not be included. In addition, if any input text file has fewer pages than the first text file, it will wrap and repeat. For instance, you may have a section of text to repeat on all pages. You could write the text in a file once (with no page breaks in the file) and the text will print at the same location on all pages.

<u>Parameter</u>	<u>Description</u>
SRC=text	The text file to use.
X=number	Left position for the text.
Y=number	Top position to start the text.

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
FACE=text	The font number or a font file on disk. 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size.
COMP=number	The text compression percentage as a whole number (for example, enter 80 for 80%).
LINESPACE=number	The amount of space to drop down for a new line (in points = 1/72 of an inch). Usually will be about the same as the SIZE value.
COLOR=text	The color for the text.
COLONE	Include this option if the page break indicator is in column one.

Parameter File

<PDFPAGE
MATCH=text
PAGE=number
X=number
Y=number
FACE=text
SIZE=number
COMP=number
LINESPACE=number
COLOR=text
COLONE>

You would normally have just one input file listed in the [FILE](#) section when using this tag. The text settings for the page are based on what PDF background page is being used. Here's an example showing three settings based on the background page being used:

```
<PDFPAGE PAGE=1 X=.25 Y=.25 COMP=100 LINESPACE=11
  COLOR="0000FF" FACE=1 SIZE=8>
<PDFPAGE PAGE=2 X=.5 Y=.25 COMP=100 LINESPACE=9
  COLOR="00000" FACE=2 SIZE=7>
<PDFPAGE PAGE=3 X=.25 Y=.25 COMP=100 LINESPACE=12
  COLOR="00990" FACE="c:\winnt\fonts\myfont.ttf" SIZE=11>
```

You'll need to use the `-pdfpagefmt` option on the executable or the `SetPDFPageFmt` method of the DLL when not using the `MATCH` parameter. You don't need this tag if you're using an image as the background or just a single page from the background PDF.

An alternate method is to specify the search string from the input text file for the PDF page number using the `MATCH` parameter. In this case, you don't need to use `-pdfpagefmt` or `SetPDFPageFmt`. Instead, this string is searched for in the input file and the PDF page is set appropriately. An example of this would be:

```
<PDFPAGE MATCH="PDFPAGE=1" PAGE=1 X=.25 Y=.25 COMP=100
  LINESPACE=11 COLOR="0000FF" FACE=1 SIZE=8>
<PDFPAGE MATCH="PDF=BODY" PAGE=2 X=.5 Y=.25 COMP=100
  LINESPACE=9 COLOR="00000" FACE=2 SIZE=7>
<PDFPAGE MATCH="SUMMARY-SECTIONY" PAGE=3 X=.25 Y=.25 COMP=100
  LINESPACE=12 COLOR="00990"
  FACE="c:\winnt\fonts\myfont.ttf" SIZE=11>
```

Here, when the text `PDFPAGE=1` is found, the background page is set to 1. Similarly, `PDF=BODY` sets the background PDF page to 2 and `SUMMARY-SECTION` sets it to 3. The lines in the input file containing this text are ignored when the text is placed in the PDF.

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
MATCH="text"	The text to look for in the input text file. You may leave this out and use the -pdfpagefmt option on the executable or the SetPDFPageFmt method of the DLL instead, depending on your input file is structured.
PAGE=number	The background PDF page number to use.
X=number	Left position for the text.
Y=number	Top position to start the text.
FACE=text	The font number or a font file on disk. 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size.
COMP=number	The text compression percentage as a whole number (for example, enter 80 for 80%).
LINESPACE=number	The amount of space to drop down for a new line (in points = 1/72 of an inch). Usually will be about the same as the SIZE value.
COLOR=text	The color for the text.
COLONE	Include this option if the page break indicator is in column one.

Parameter File

**<RECTANGLE
X1=number
Y1=number
X2=number
Y2=number
COLOR=text>**

This tag is used to place a filled rectangle on the page. The coordinate system is in inches (or units) with the top left hand corner being position 0,0. To place a green rectangle in the middle of an 8.5 by 11 inch page you could use the following example:

```
<RECTANGLE X1=3.25 Y1=4.5 X2=5.25 Y2=6.5 COLOR=00CC00>
```

You may place as many rectangles as you want. Each is placed on the form in the order it is found in the file.

<u>Parameter</u>	<u>Description</u>
X1=number	The X position of the first corner.
Y1=number	The Y position of the first corner.
X2=number	The X position of the second corner.
Y2=number	The Y position of the second corner.
COLOR=text	The color for the rectangle.

Parameter File

```
<BOOKMARK  
  LEVEL=number  
  LINE=number  
  FROMCOL=number  
  THRUCOL=number  
  CLOSED>
```

This tag is used to define an area on the page for a bookmark. Bookmarks can be used to quickly jump from one section of a document to another. The bookmarks should represent some logical ordering of the input file. For example, employee number if the text input is in employee number order. The line number is based on the text input with line 1 as the top most line. The from and through column numbers represent the position in the text line the data you want to use as the bookmark is located. The first column is counted as column 1.

You may also place multiple bookmark entries at the same level in this section. The program will attempt to set the text for each level based on the first entry for that level. If no text is found at the specified position, the program will check the next line number and position specified for that level. Here's an example of a bookmark structure with two levels:

```
<BOOKMARK LEVEL=1 LINE=5 FROMCOL=3 THRUCOL=9 CLOSED>  
<BOOKMARK LEVEL=2 LINE=2 FROMCOL=16 THRUCOL=27>  
<BOOKMARK LEVEL=2 LINE=1 FROMCOL=20 THRUCOL=35>
```

In this example, the top most bookmark is found on line 5 from position 3 to 9 and is initially closed (usually displayed with a + sign next to it for expansion by the user). The second level is the text found on line 2 from position 16 to 27. The text found on line 1 from position 20 to 35 will be used instead for the second level if there is no text found on line 2, position 16 to 27. The assumption here is the text for the second bookmark level will vary from page to page while the text for the first bookmark level will not vary as often. The first level might represent a department code while the second an employee number. Note that you may use one or more levels, however it's not typical to go beyond five or six.

<u>Parameter</u>	<u>Description</u>
LEVEL=number	The level number of the bookmark starting with 1 as the highest level.
LINE=number	The line number in the input file the bookmark text is found on.
FROMCOL=number	The beginning column number for the bookmark text on the line.
THRUCOL=number	The ending column number for the bookmark text on the line.

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
CLOSED	Used to specify the bookmark should be closed initially, or the lower levels collapsed.

Parameter File

The ROWPAGE and ROWLINE tags are used for input files with no form feed characters or column 1 indicator. The ROWPAGE tag is used to specify the number of lines from the input file make up a page. The ROWLINE is used to specify the placement of the various lines from the input file.

The ROWPAGE tag is required in these cases in order to know how many lines of the input file make up a page. Use of the ROWLINE tag is optional. You'll only need that tag when you want to layout the lines in some other order or do something different that just display the ROWPAGE number of lines on each page.

Parameter File

<ROWPAGE LINES=number>

The ROWPAGE tag is used for input files with no form feed characters or column 1 indicator. In here, you specify the number of lines from the first input file that make up a page. For example, <ROWPAGE LINES=50>. In this case, for every 50 lines of input, start a new page. You may stop here if the data is simply to be placed on the form in 50 line increments. Or you may specify the structure and placement of those lines. The ROWLINE tag is used to describe the layout of the lines from the input file.

<u>Parameter</u>	<u>Description</u>
LINES=number	The number of lines from the input file that will be read per page.

Parameter File

<ROWLINE
FROMLINE=number
THRULINE=number
REPEATCNT=number
FROMCOL=number
NUMCHARS=number
X=number
Y=number
FACE=text
SIZE=number
COMP=number
LINESPACE=number
COLOR=text>

Only the FROMLINE and THRULINE options are required when using this tag. You may also use 0 as the beginning and ending line number for a blank line. The REPEATCNT value is used to specify the number of times you want the block of lines to repeat. The FROMCOL and NUMCHARS are used to pull a sub-string out of the current line. If you only want 10 characters from position 5, you'd set FROMCOL=5 and NUMCHARS=10. Leave out FROMCOL and NUMCHARS to use the entire line. Here's an example:

```
<ROWPAGE LINES=50>  
<ROWLINE FROMLINE=1 THRULINE=5 REPEATCNT=2>  
<ROWLINE FROMLINE=3 THRULINE=6 X=.5 Y=1 COMP=80  
  LINESPACE=13 COLOR=990000>  
<ROWLINE FROMLINE=6 THRULINE=40 FROMCOL=5 X=2 Y=5 COMP=100  
  LINESPACE=11 COLOR=000000>
```

In this example, the input file is expected to contain 50 text lines per page. First, lines 1 through 5 will shown followed by lines 1 through 5 again. The text position will then move to .5 inches from the left and 1 inch down. The text compression will be set to 80 and line spacing to 13 and the color to red. Lines 3 through 6 from the input file will then be printed once. Finally, the text position will move to 2 inches from the left and 2 inches from the top. The compression will be set back to 100 with a line spacing of 11 and text color of black. Column 5 and on from lines 6 through 40 will be printed once and the page will end. The cycle will then repeat on the next page with lines 51 to 100 from the input file.

<u>Parameter</u>	<u>Description</u>
FROMLINE=number	The beginning line number from the input file this block refers to.
THRULINE=number	The ending line number from the input file this block refers to.

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
REPEATCNT=number	The number of times to repeat these lines (default is 1).
FROMCOL=number	The starting column number for this block of text (default is 1).
NUMCHARS=number	The number of characters per line for this block of text (default is to end of line).
X=number	Left position for the text.
Y=number	Top position to start the text.
FACE=text	The font number or a font file on disk. 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size.
COMP=number	The text compression percentage as a whole number (for example, enter 80 for 80%).
LINESPACE=number	The amount of space to drop down for a new line (in points = 1/72 of an inch). Usually will be about the same as the SIZE value.
COLOR=text	The color for the text.

Parameter File

```
<COLONEFLAG  
  CODE=text  
  FACE=text  
  SIZE=number  
  COMP=number  
  LINESPACE=number  
  COLOR=text>
```

This tag lists special font settings based on a flag in column one of the input data. Here's an example:

```
<COLONEFLAG CODE=1 FACE=1 SIZE=10 LINESPACE=11  
  COLOR=00CC00>  
<COLONEFLAG CODE=2 FACE="c:\myfont.ttf" SIZE=12  
  LINESPACE=13 COLOR=000000>
```

The font is set to 1 (courier) with a point size of 10, line spacing 11 and green text when the character in column one is a "1". The font is set to the custom myfont.ttf with a point size of 12, line spacing 13 and black text when the character in column one is a "2".

<u>Parameter</u>	<u>Description</u>
CODE=text	A single character to look for in column one of the input text file.
FACE=text	The font number or a font file on disk. 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size.
COMP=number	The text compression percentage as a whole number (for example, enter 80 for 80%).

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
LINESPACE=number	The amount of space to drop down for a new line (in points = 1/72 of an inch). Usually will be about the same as the SIZE value.
COLOR=text	The color for the text.

Parameter File

```
<TEXTPROP
  X1=number
  Y1=number
  X2=number
  Y2=number
  CODE=text
  FACE=text
  SIZE=number
  COMP=number
  COLOR=text
  OFF=text
  ON=text>
```

This tag allows you to set the text properties for a text line or lines based on a rectangle on the page. The rectangle is in text coordinates so the X values are the character positions and the Y values are the line numbers. Or, you can use the CODE option to look for a string in the input text rather than X/Y coordinates. Here's an example:

```
<TEXTPROP X1=5 Y1=18 X2=30 Y2=40 FACE=1 SIZE=10 COLOR=00CC00>
<TEXTPROP X1=60 Y1=3 X2=70 Y2=3 SIZE=12 FACE="c:\myfont.ttf"
  COLOR=000000>
```

The font is set to 1 (courier) with a point size of 10 and green color for text on lines 18 to 40 in columns 5 to 30. The font is set to the custom myfont.ttf with a point size of 12 and color black for text on line 3 in columns 60 to 70.

Here's an example using the CODE option:

```
<TEXTPROP CODE="&lt;green&gt;" FACE=1 SIZE=10 COLOR=00CC00>
<TEXTPROP CODE="&lt;black&gt;" SIZE=12 FACE="c:\myfont.ttf"
  COLOR=000000>
```

This time, the string <green> and <black> will be searched for instead. Note the use of < and > for the < and > characters. These are needed when you have angle brackets to search for in the text so the software can parse the TEXTPROP tag properly.

Be sure to not overlap the areas when using multiple TEXTPROP tags. That is, the rectangles must not overlap one another in two or more TEXTPROP areas. The only time you can overlap is when the OFF/ON values are set such that no two overlapping TEXTPROP areas are in effect at the same time.

If you use the CODE option then all your TEXTPROP tags must use the option. You cannot have a mix of TEXTPROP tags with some using X/Y coordinates and others using CODE.

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
X1=number	The beginning character position or column number for the block.
Y1=number	The beginning line number for the block.
X2=number	The ending character position or column number for the block.
Y2=number	The ending line number for the block.
CODE=text	A user-defined string to look for in the line of text rather than X/Y coordinates. For example, you could use a tag such as to turn on bolding or {boldon} or anything else. You then set up a TEXTPROP tag with CODE="{boldon}" to have the TEXTPROP settings take effect. If your input text line contains "Here is a {boldon}bold section{boldoff} of text" then {boldon} will start the properties associated with that code and {boldoff} will set a different property. Note that both {boldon} and {boldoff} must be explicitly set with a TEXTPROP tag. That is, there is no closing tag as in HTML since these are user-defined strings and not tags. The text {boldon} and {boldoff} will be removed from the text before writing to the PDF. The X1, Y1, X2, Y2, OFF and ON options are ignored when CODE is used.
FACE=text	The font number or a font file on disk. 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size.

PDF Forms

Parameter File

<u>Parameter</u>	<u>Description</u>
COMP=number	The text compression percentage as a whole number (for example, enter 80 for 80%). You may also set to "auto" and the program will determine the appropriate value based on the current line's font size and compression. This allows you to set the font size larger or smaller than the current line and still have the physical length of the text line be the same.
COLOR=text	The color for the text.
OFF=text	Do not perform the text modifications on any lines containing the text string. May use a pipe delimiter (the character) for multiple strings. The text is case sensitive.
ON=text	Only perform the text modifications on lines containing the text string. May use a pipe delimiter (the character) for multiple strings. The text is case sensitive.

Text Files

The `-strin` option or `SetStrFileIn` method allows you to add text or images on a page outside of the standard text input file. Use this to add watermarks or other information. You specify the X/Y coordinates and, optionally, other aspects of the text. The text can be a single line or multi-line. The text breaks or wraps based on where you have line breaks in your input file. You may also use this feature to place invisible text on pages to allow for searching on the text.

You may use the following variables in your text. To include the text without converting to the value, place a backslash (the `\` character) in front of the `&` symbol.

- `&page;` The current page number.
- `&totpages;` The total number of pages in the output PDF.

Each tag, or command, starts with an angle bracket `<` and closes with `>`. The tag name comes directly after the opening `<`. Options are then listed, space separated, with an `=` sign between it and its value. Note this tag has an opening and closing tag. For example:

```
<UNITS VALUE="in">  
<TEXT X=2 Y=3 COLOR=red ALIGN="center">  
Sample text line 1  
Next line (page=&page;)  
</TEXT>
```

will print "Sample text line 1" with "Next line" underneath.

Text Files

**<UNITS
VALUE="text">**

The UNITS tag is used to set the unit of measure for the TEXT and/or IMG tag(s).

<u>Option</u>	<u>Description</u>
VALUE="text"	Used to set the unit of measure. Set this option first so latter TEXT tags use the proper unit setting. Valid settings are: pt - points (1/72 of an inch) in - inches cm - centimeters mm - millimeters

Text Files

```
<TEXT
  X=number
  Y=number
  FROMRIGHT
  FROMTOP
  FACE="text"
  SIZE=number
  ALIGN="text"
  ANGLE=number
  ANCHOR="text"
  NOROTATE
  IFROTATE=number
  COLOR="text"
  BGCOLOR="text"
  BORDER=number
  BORDERCOLOR="text"
  PADDING=number
  LINESPACE=number
  COMP=number
  REND=number
  SHADOW=number
  SHADOWX=number
  SHADOWY=number
  SHADOWCOLOR="text"
  SHADOWREND=number
  TRANSPARENCY=number
  TRANSPMODE="text"
  PAGES="text"
  FIRST
  NOTFIRST
  BASELINE=Yes|No
  GLYPHPOS>
</TEXT>
```

The TEXT tag is used to place text on a page or multiple pages.

<u>Option</u>	<u>Description</u>
X=number	Required. The position (in points where 1 point = 1/72 of an inch or based on current UNITS setting) from the left page edge to place the text.
Y=number	Required. The position (in points where 1 point = 1/72 of an inch or based on current UNITS setting) from the bottom page edge to place the baseline of the first line of text.

PDF Forms

Text Files

<u>Option</u>	<u>Description</u>
FROMRIGHT	Treats the X value as a distance from the right edge of the page instead of the left.
FROMTOP	Treats the Y value as a distance from the top edge of the page instead of the bottom.
FACE="text"	Set to the number of one of the following: 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size. The default is 10.
ALIGN="text"	The alignment. Use "Center" or "Right". The default is "Left".
ANGLE=number	The counter-clockwise angle of rotation around the center of the text block. Enter a value from 0 to 360. Use a negative number from -360 to 0 for clockwise rotation. The default is 0.
ANCHOR=text	Used to specify a different anchor point for the rotation. The default is the center of the text block. Specify one of the following: LL for Lower-Left LR for Lower-Right UL for Upper-Left UR for Upper-Right For most cases you'll want to use LL as the value when overriding the default center point rotation. For example, if you have a string rotated 90 degrees with the default center point rotation, its X position will vary depending on the length of the string. To keep the X position fixed, use ANCHOR="LL".

Text Files

<u>Option</u>	<u>Description</u>
NOROTATE	Prevents the text from rotating when there is a page rotation. Sometimes pages that appear to be oriented correctly when viewed actually have a rotation set on the page. When you place text on the page it then becomes rotated even though you didn't specify an ANGLE for it. This option will compensate for the page rotation and adjust the angle of the text. Note that ANCHOR will be set to "UL" and BASEALIGN will be set to "No" when this option is used regardless of what value is passed in.
IFROTATE=number	Set to a value of 0, 90, 180 or 270. This is used to specify what pages the text should be printed on based on the current page's rotation value. You might use this with the NOROTATE option to provide a different position for the text based on how the page is rotated.
COLOR="text"	The color for the text.
BGCOLOR="text"	The background color for the text.
BORDER=number	The width of the border in points. Can be a decimal number (such as .5) for a thin border.
BORDERCOLOR="text"	The color color for the border.
PADDING=number	The amount of padding to leave between the text and the border (also for background color) in points.
LINESPACE=number	The amount of space in points (1/72 of an inch) to drop down between text lines. The default is 2 + the font point size.
COMP=number	The percentage amount to horizontally compress the text by. Enter as a whole number (for example, use 80 for 80%). The default is 100.
REND=number	The render mode. The default is 0 (fill the character). Other options are: 1 - stroke (or outline) the character only 2 - stroke and fill the character 3 - no stroke or fill (invisible)
SHADOW=number	Draws a shadow effect for the text. Set to the transparency value you want for the shadow. Enter a value from 1 to 100. The lower the number, the lighter the shadow.

Text Files

<u>Option</u>	<u>Description</u>
SHADOWX=number	The X-axis offset for the shadow. Enter a value in points (1 point = 1/72 of an inch). Can be a positive or negative value (or decimal values). General range is -5 to 5.
SHADOWY=number	The Y-axis offset for the shadow. Enter a value in points (1 point = 1/72 of an inch). Can be a positive or negative value (or decimal values). General range is -5 to 5.
SHADOWCOLOR="text"	The color for the shadow. Default is black.
SHADOWREND=number	The rendering mode for the shadow text. The default is 2. Use 0 for a sharper shadow or 1 for a hollow shadow.
TRANSPARENCY=number	Optional transparency for the text. Requires Acrobat or Reader 5.0 or higher to view the transparency. Enter a value from 1 to 100. The default without this option is 100 - the text is placed on top of the background PDF with none of the background showing through.
TRANSPMODE="text"	Optional transparency mode. The valid values are: Normal (Default) Multiply Screen Overlay Darken Lighten ColorDodge ColorBurn HardLight SoftLight Difference Exclusion Hue Saturation Color Luminosity

Text Files

<u>Option</u>	<u>Description</u>
PAGES="text"	The list of pages to show the text on. Enter a comma separated list and/or use a - (minus sign) for a page range. May also use "odd" or "even". Use "oddrange" or "evenrange" along with a page range to only pull odd or even pages within that range. For example, PAGES="10-300,oddrange" will only include odd pages in the range of 10-300 (11, 13, ... 299). In comparison, PAGES="10-300,odd" will pull all odd pages and all pages within the range of 10-300. Leave PAGES out to include on all pages.
FIRST	Include on first page only (same as setting PAGES=1).
NOTFIRST	Include on any page except the first page. For example, if you set PAGES="odd" and don't want this text to show on the first page, set this option.
BASELINE=Yes No	If set to "No", the Y value (for text position) represents the top of the first line of text. If set to "Yes" (the default), the Y value represents the baseline of the first line of text. Setting to "No" has the effect of lowering the text by the point size. This is fixed as "No" when NOROTATE is used.
GLYPHPOS	Used when the text contains positioning (kerning) information. Your text must be entered in parenthesis with a number between each set of parenthesis. A positive number will move the current text position to the left while a negative number will move it to the right. That is, the value is subtracted from the current position to obtain the new position. The value is specified in 1000ths of a unit. See the example at the end of this section.

Here's a sample:

```
<UNITS VALUE="in">
<TEXT X=1 Y=3 FACE="5" PAGES="1-3,6" COLOR="green"
ALIGN="center">
This is the first line of text.
This is another line.
</TEXT>
<TEXT X=2.5 Y=4 SIZE=15 FACE="helvetica" ALIGN="center"
BORDER=3 PADDING=2>
Here is some text to center.
Line 2
```

Text Files

Last line of centered text.

```
</TEXT>
```

```
<TEXT X=1 Y=1 REND=3>
```

This text is invisible.

```
</TEXT>
```

```
<TEXT X=1 Y=6 SIZE=18 GLYPHPOS>
```

(T)-20(his)-215(text)-220(uses glyph)-230(positioning.)

(A)20(11 text must go within parenthesis.)

(Use a slash in front of any parenthesis in the text.)

(For example: \ (555\) 555-1212)

```
</TEXT>
```

Keep the following in mind when creating your tagged file:

- No spaces between the option and = sign
- No spaces between = and the value

Text Files

```
<IMG  
  SRC="text"  
  X1=number  
  Y1=number  
  X2=number  
  Y2=number  
  PAGES="test"  
  FIRST  
  NOTFIRST>
```

This tag is used to place an image on a page. The X2 and Y2 values are optional. The width and height of the image will be used if not provided. The X1 and Y1 values will represent the lower left corner of the image in this case.

<u>Parameter</u>	<u>Description</u>
SRC=text	The path and file name of the image to use.
X1=number	The X coordinate in inches or units for the lower left image corner.
Y1=number	The Y coordinate in inches or units for the lower left image corner.
X2=number	Optional. The X coordinate in inches or units for the upper right image corner.
Y2=number	Optional. The Y coordinate in inches or units for the upper right image corner.
PAGES="text"	The list of pages to show the image on. Enter a comma separated list and/or use a - (minus sign) for a page range. May also use "odd" or "even". Use "oddrange" or "evenrange" along with a page range to only pull odd or even pages within that range. For example, PAGES="10-300,oddrange" will only include odd pages in the range of 10-300 (11, 13, ... 299). In comparison, PAGES="10-300,odd" will pull all odd pages and all pages within the range of 10-300. Leave PAGES out to include on all pages.
FIRST	Include on first page only (same as setting PAGES=1).
NOTFIRST	Include on any page except the first page. For example, if you set PAGES="odd" and don't want this text to show on the first page, set this option.

Addendum

The `-addendum` option or `SetAddendum` method allows you to pass a file name or the contents describing what images or PDF pages to show at the end of the PDF output. You must use a background PDF in order to pull pages for the ending section.

There are two tags you may use — `` and `<PDFPAGE>`. You can use both but the `PDFPAGE` tag only works when you're using a background PDF for the output. The page number specified is from that background PDF.

For example, to have an image and two pages from the background PDF the file would look similar to this:

```
<IMG SRC="c:\images\myimg.jpg" X1=.5 Y1=.5 X2=8 Y2=10.5
    PAGEW=8.5 PAGEH=11>
<PDFPAGE PAGE=5>
<PDFPAGE PAGE=6>
```

The image will be placed first, each image placed on its own page, followed by pages 5 and 6 from the background PDF. You can mix in any order and provided as many images or pages as you like. The order given in the file is the order used to place in the output.

Addendum

```
<IMG  
  SRC="text"  
  X1=number  
  Y1=number  
  X2=number  
  Y2=number  
  PAGEW=number  
  PAGEH=number>
```

This tag is used to place an image on a page. The X2 and Y2 values are optional. The width and height of the image will be used if not provided.

<u>Parameter</u>	<u>Description</u>
SRC=text	The path and file name of the image to use.
X1=number	The X coordinate in inches or units for the lower left image corner.
Y1=number	The Y coordinate in inches or units for the lower left image corner.
X2=number	Optional. The X coordinate in inches or units for the upper right image corner.
Y2=number	Optional. The Y coordinate in inches or units for the upper right image corner.
PAGEW=number	The width of the page in inches or units.
PAGEH=number	The height of the page in inches or units.

Addendum

<PDFPAGE PAGE=number>

This tag is used to insert the page specified at the end of the PDF output. You must use a background PDF with the output in order for this tag to work.

<u>Parameter</u>	<u>Description</u>
PAGE=number	The background PDF page number to use.

Addendum





<UNITS VALUE=text>

Used to define the units setting for the addendum information. The IMG tag is the only tag using the units setting. The default value is from the -units option or setUnits method. The default when not using -units or setUnits is inches.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Unit of measure. Use in for inches, cm for centimeters, mm for millimeters or pt for point. One point is 1/72 of an inch. For example, setting this option to "cm" means the IMG tag will expect values in centimeters for image placement and page size.

Barcode Fonts

PDF Forms includes several barcode fonts you can use without having to load your own. They include a 3 of 9 font and several UPCA fonts. Simply specify the font number using tags such as TEXTPROP. You do not have to specify the check digit on UPCA codes as PDF Forms will compute the value automatically if not provided. For example, <TEXTPROP X1=60 Y1=3 X2=71 Y1=3 SIZE=12 FACE=16>. Here are samples of the various barcode fonts:

<u>Font</u>	<u>Name</u>	<u>Sample</u>
15	3 of 9	 0 1 2 3 4 5 A B C A B C
16	UPCA	 0 1 2 3 4 5 0 1 2 3 4 1
17	UPCA	 0 1 2 3 4 5 0 1 2 3 4 1
18	UPCA	 0 1 2 3 4 5 0 1 2 3 4 1

Digital Signatures

Reasons for Using

Digital signatures provide a way to sign a PDF electronically in order to authenticate its contents. Signing a PDF electronically is a process that creates a unique encoding out of the entire PDF. No two PDFs will have the same encoding unless they are exactly the same. This encoding is then signed (further encoded) by running a program that takes as input the encoding, a public key signature file, and a private key password to create the PDF signature. The end user can verify the PDF is authentic by checking that the applied signature is valid. Adobe Reader or Acrobat will recompute the encoding over the entire PDF and if the signature in the PDF does not match then the signature is no longer valid.

Most casual users of PDF familiar with Adobe Reader mistakenly believe that PDFs cannot be modified. In reality, it's fairly easy to change text in a PDF using Adobe Acrobat or other third-party programs like FyTek's PDF Meld. A digital signature is therefore used not to prevent changes but to validate that a PDF you are viewing has not been altered or, if altered and the signature is still valid, what changes were made. Any changes to the document after the signature is applied will be noted in the signature pane in Reader or Acrobat so you know what was modified.

Requirements for Signing

PDF Forms uses an open source program called OpenSSL that is available for most Unix and Windows installations. If you do not have OpenSSL installed, you'll need to install it first before you can digitally sign documents. Most Unix systems will likely have it - if you're not sure, try typing OpenSSL at a shell prompt and if it comes back with a prompt that looks like OpenSSL> then it is installed. Windows binaries are available here: <http://www.slproweb.com/products/Win32OpenSSL.html> or, if the link is not available, search for "openssl windows binary" in your favorite search engine.

The following section deals with using OpenSSL to create your signature files. While you don't need to be an expert at digital certificates, you should be comfortable running commands from the DOS prompt. This is a process you will probably only run once to set your certificates. Once you have them you simply supply them to the PDF Forms for signing so this section is not something you will need to do each time you want to sign a PDF.

First you'll need a certificate to sign PDFs with. You may purchase them from security companies on-line or use OpenSSL to create your own. There are 2 files you'll need to sign with:

Digital Signatures

mykey_pk.pem - your private signing key
mykey.der - your binary certificate and public key

There are two types of file formats for certificate files. One is PEM which is a text file and DER which is binary. The names of your files may be different but the point is you'll need a private key in pem format and a certificate in binary form.

Covering all the commands of OpenSSL is beyond the scope of this document. We'll just be covering the basics to get a certificate setup. There are many websites to explain other uses and options for OpenSSL if you are interested. Installing OpenSSL should only take a few minutes depending on your internet connection.

Setting up OpenSSL

At this point you should have OpenSSL installed - use the link mentioned in the previous section if you need to install on Windows. The first step is to create a configuration file for OpenSSL if you don't have one already. You only need to do this once and you may place it in the directory you installed OpenSSL into. Here's a sample openssl.cnf file to get you started if you need one. This file is also embedded in this PDF so you can download it from this document rather than cut & paste.

```
#
# SSLey example configuration file.
# This is mostly being used for generation of certificate requests.
#
RANDFILE              = .rnd
#####
[ ca ]
default_ca             = CA_default          # The default ca section
#####
[ CA_default ]
dir                   = demoCA              # Where everything is kept
certs                 = $dir\certs          # Where the issued certs are kept
crl_dir               = $dir\crl           # Where the issued crl are kept
database              = $dir\index.txt     # database index file.
new_certs_dir         = $dir\newcerts      # default place for new certs.
certificate           = $dir\cacert.pem    # The CA certificate
serial                = $dir\serial        # The current serial number
crl                   = $dir\crl.pem       # The current CRL
private_key           = $dir\private\cakey.pem # The private key
RANDFILE              = $dir\private\private.rnd # private random number file
x509_extensions       = x509v3_extensions  # The extensions to add to the cert
default_days          = 365                # how long to certify for
default_crl_days      = 30                 # how long before next CRL
default_md             = md5               # which md to use.
preserve              = no                 # keep passed DN ordering
# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy                 = policy_match
# For the CA policy
```

Digital Signatures

```
[ policy_match ]
countryName           = optional
stateOrProvinceName  = optional
organizationName      = optional
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional
# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName           = optional
stateOrProvinceName  = optional
localityName          = optional
organizationName      = optional
organizationalUnitName = optional
commonName            = supplied
emailAddress          = optional
#####
[ req ]
default_bits          = 1024
default_keyfile        = privkey.pem
distinguished_name    = req_distinguished_name
attributes            = req_attributes
[ req_distinguished_name ]
countryName           = Country Name (2 letter code)
countryName_min       = 2
countryName_max       = 2
stateOrProvinceName  = State or Province Name (full name)
localityName          = Locality Name (eg, city)
0.organizationName    = Organization Name (eg, company)
organizationalUnitName = Organizational Unit Name (eg, section)
commonName            = Common Name (eg, your website's domain name)
commonName_max        = 64
emailAddress          = Email Address
emailAddress_max      = 40
[ req_attributes ]
challengePassword     = A challenge password
challengePassword_min = 4
challengePassword_max = 20
[ x509v3_extensions ]
# under ASN.1, the 0 bit would be encoded as 80
nsCertType            = 0x40
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName
#nsCertSequence
#nsCertExt
#nsDataType
```

The end user will not need to do anything special to use certificates you create but they will not be trusted certificates. They have the option to trust your certificate, if they wish, but they do not have to. In either case, Reader

will report whether or not the document has been modified since it was signed.

Creating Self-Signed Certificates

Now that OpenSSL is setup, here are the steps to create a self-signed certificate. Note that there are a variety of security companies that sell self-signed certificates. However, we'll use OpenSSL here to show you how to create your own in just a few short steps.

1. Open a DOS window or a shell in Linux/Unix.
2. Be sure your PATH environment variable contains the executable for OpenSSL. This will be the directory you installed it into. If not set, you can type this at the DOS prompt:

```
path=%path%;c:\(openssl-directory)
```

Where the "(openssl-directory)" is replaced with the directory containing the binary openssl.exe program. This should be the directory you installed the program into along with the path of \bin at the end of that.

3. Create the public and private key files by running the following:

```
openssl req -x509 -new -config openssl.cnf -days 365 -out mykey.pem -keyout mykey_pk.pem -newkey rsa:2048
```

The file mykey_pk.pem is the private key you'll use for the SIGNPKFILE option. You may set the number of days for expiration to whatever you want. In the example, we've used 1 year but you may set for whatever you like. This is just the expiration for the certificate. Be sure to put the full path to openssl.cnf in the line above if it is not in your current directory. The -newkey rsa:2048 (or rsa:4096) is optional if you want to create larger encryption keys than the 1024 default size.
4. Create the certificate by running this command:

```
openssl x509 -in mykey.pem -inform PEM -out mykey.der -outform DER
```

The file mykey.der is the binary certificate. This contains the public key that will be used to verify your signature in the PDF. Now you should have mykey_pk.pem and mykey.der on your system.

Passing Signature Information

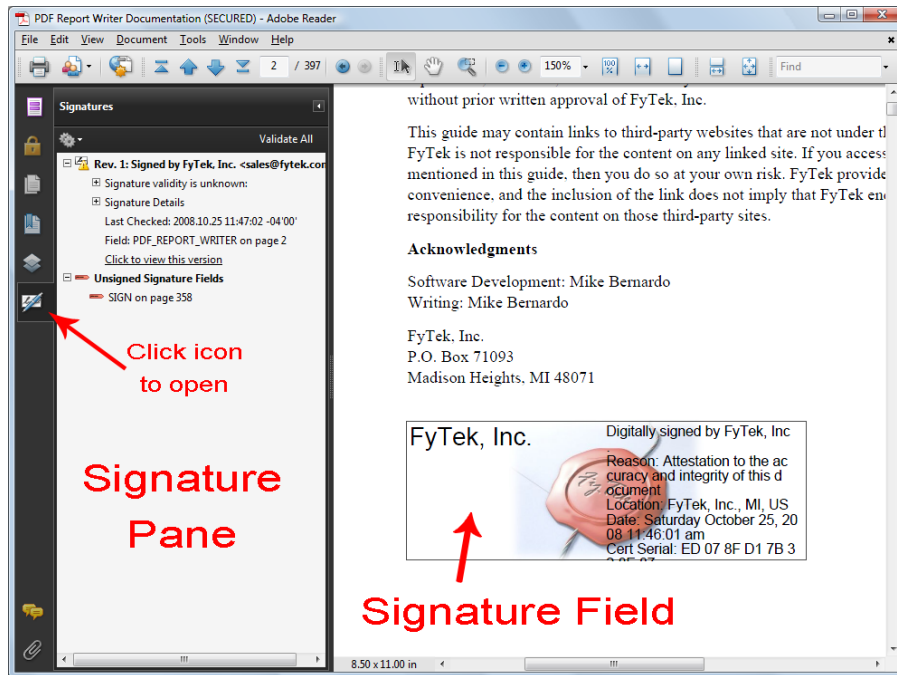
Now that you have the files, use them on the command line or the SetSign method in PDF Forms. On the command line, -signpkfile would be set to mykey_pk.pem and -signderfile would be mykey.der using the example above. You do not have to pass the signing password to the program. If you leave it out you'll be prompted for it when it is needed to encode.

The password may be supplied in a couple of ways. You may pass the password itself, such as -signpwd "abc123" on the command line. You may also pass the password from an environment variable such as -signpwd "env:pwd" on the command line. In this case, prefix the variable with env: and the system will look for the environment variable called "pwd" (or

whatever name you choose) and use its value as the password.

Trusting Certificates

You will see something similar to the following after you sign a document for the first time.

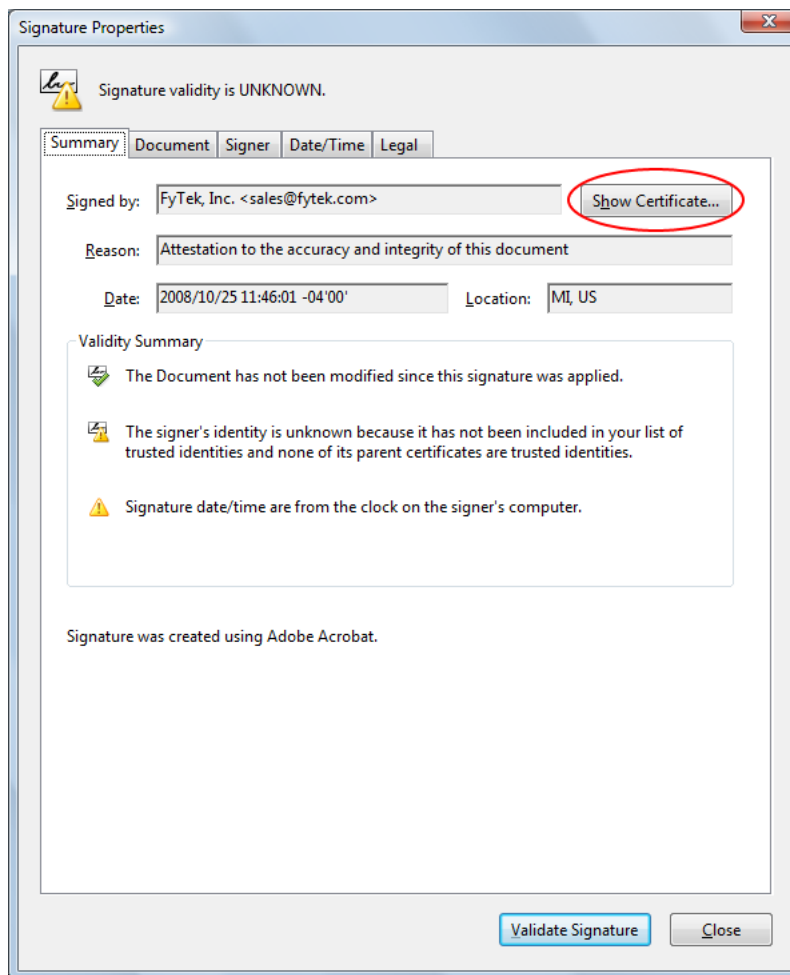


Note the icon with the yellow warning icon in the signature pane. This is because the certificate has not yet been trusted by Reader. Once you have trusted the certificate the icon will change and all future signatures in PDFs with this certificate will be recognized. The first step is to click on the signature field to bring up the dialog box shown. Your dialog boxes may differ slightly in options depending on the version of Adobe Reader you are using. These examples use Adobe Reader version 9. Note this document is signed so you can follow the steps below for this PDF if you like.

Digital Signatures

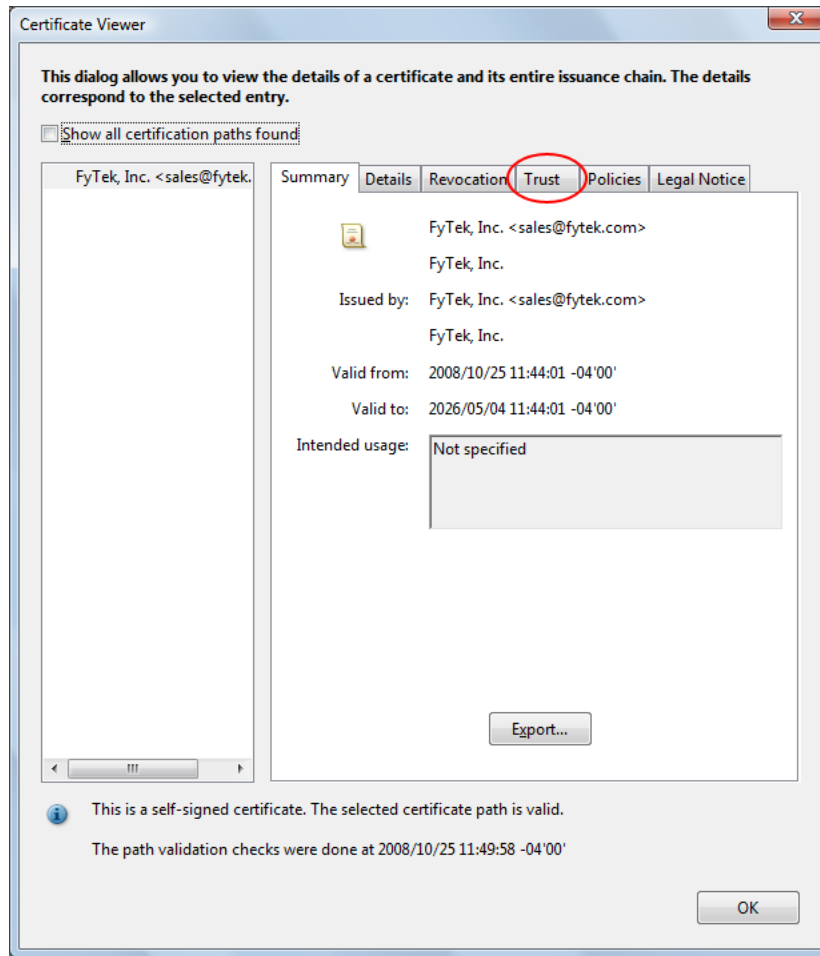


This is the dialog box that appears once you click the signature field. Click the "Signature Properties..." button to continue.



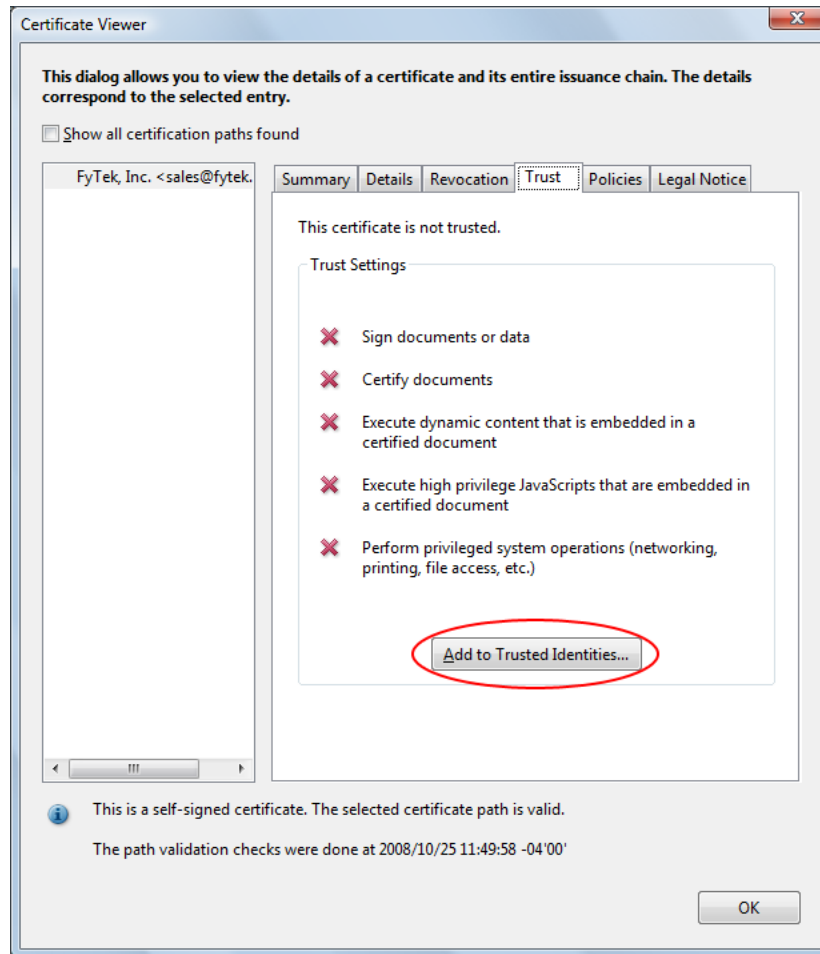
This is the signature property dialog for the certificate. Across the top of the dialog area is a set of tabs you can click on to view various information. For now, click the "Show Certificate..." button to continue.

Digital Signatures

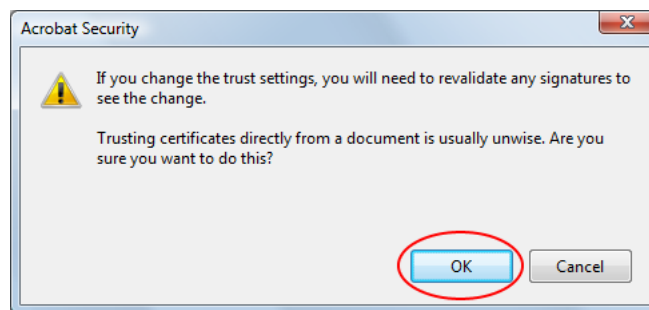


Another dialog box will open containing a set of tabs across the top. Click on the "Trust" tab.

Digital Signatures

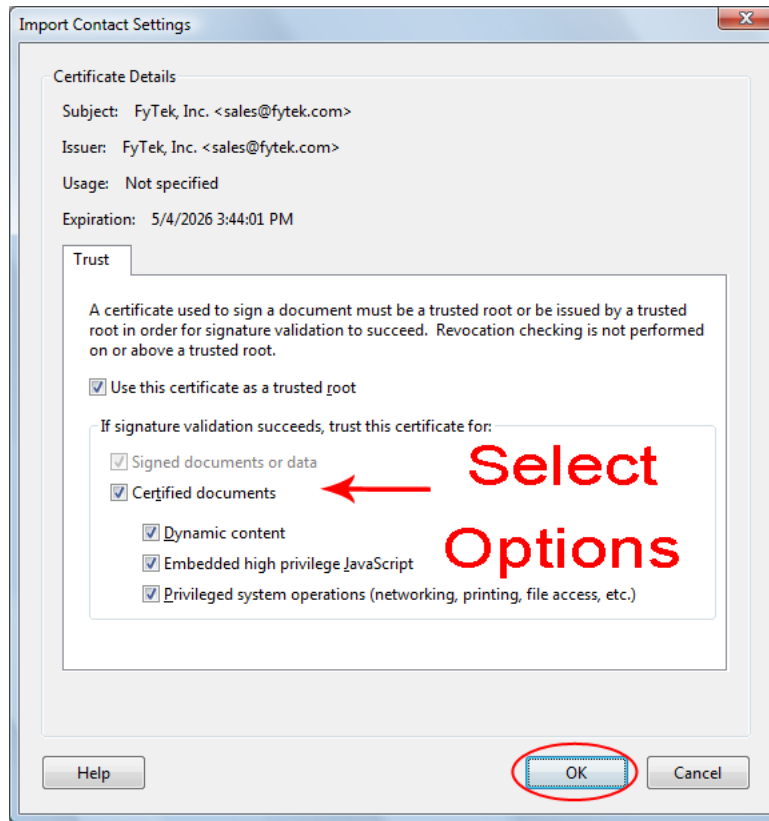


The trust tab shows what trusts you have enabled for the certificate. In this case, no trusts have been established. To trust this certificate, click the "Add to Trusted Identities..." button.



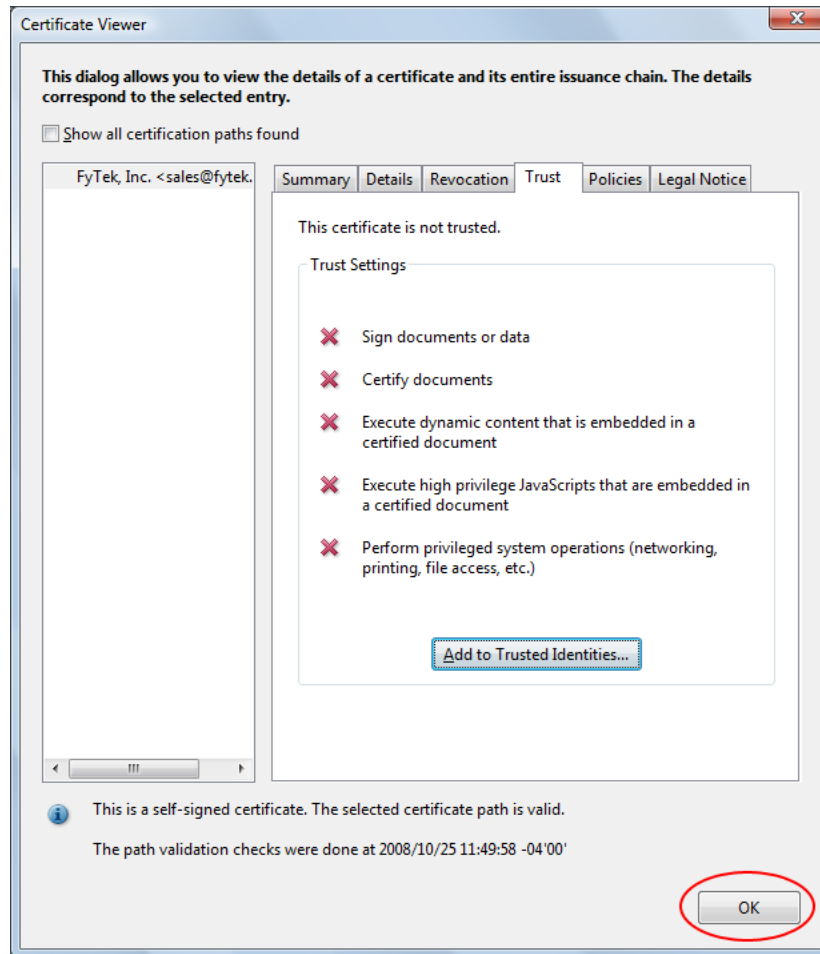
You will likely receive a warning box. Be sure to only trust certificates when you are certain of their source.

Digital Signatures



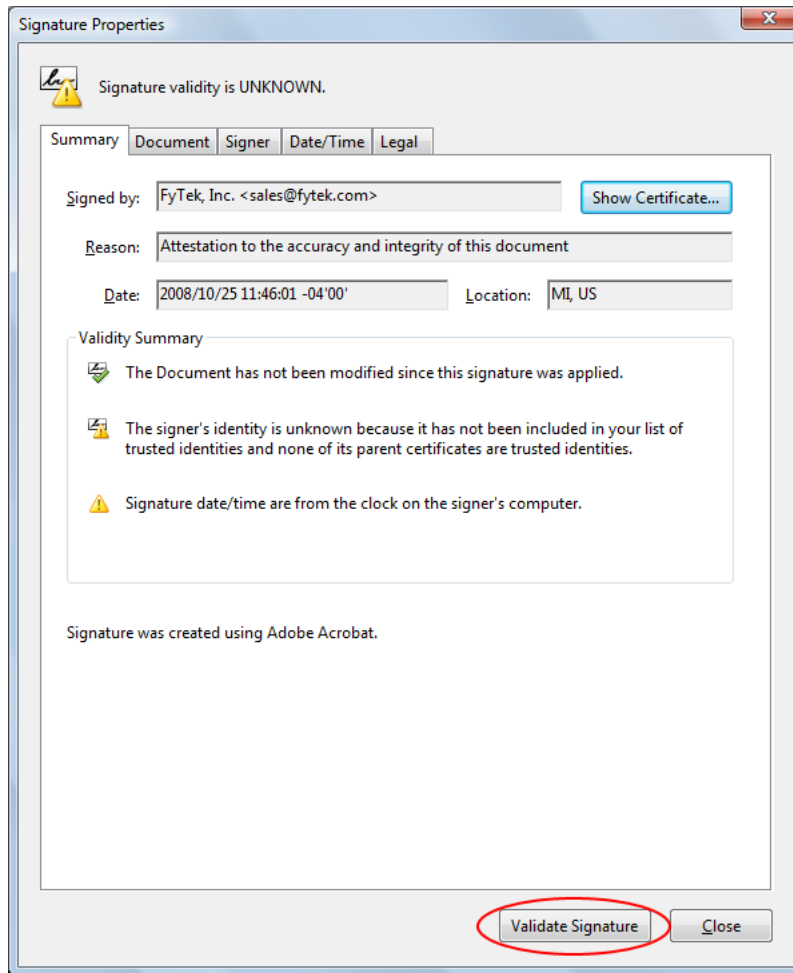
Select what items you want to trust the certificate for by clicking the checkboxes.

Digital Signatures



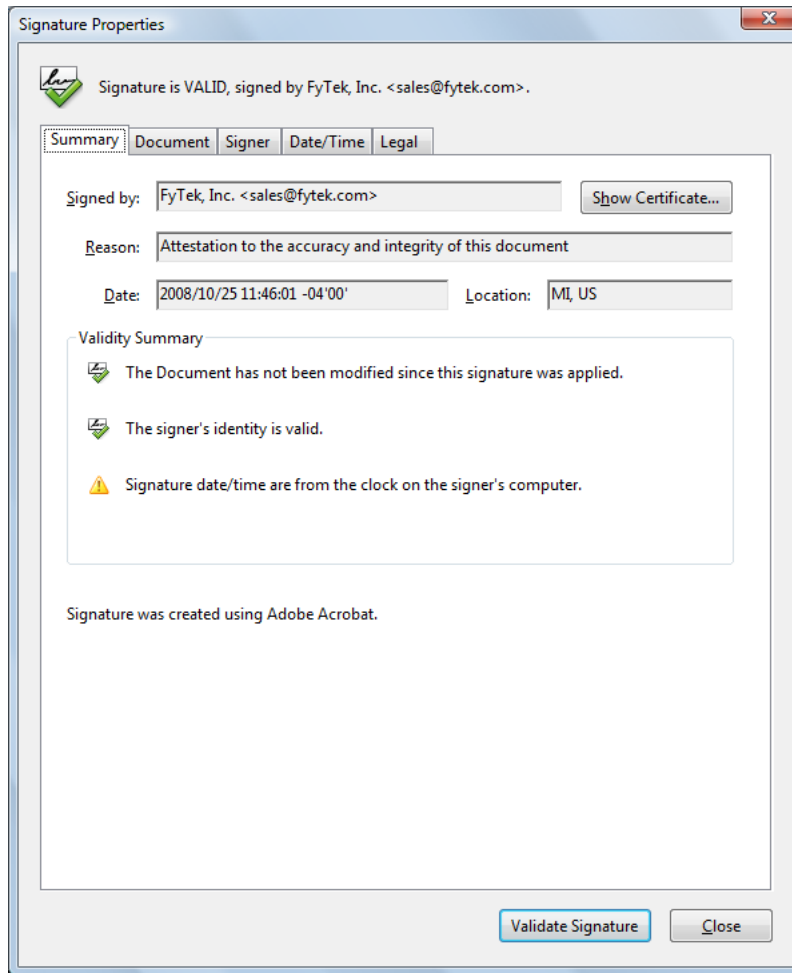
Click the "OK" button to continue. Note the red X's will remain until we revalidate the signatures.

Digital Signatures



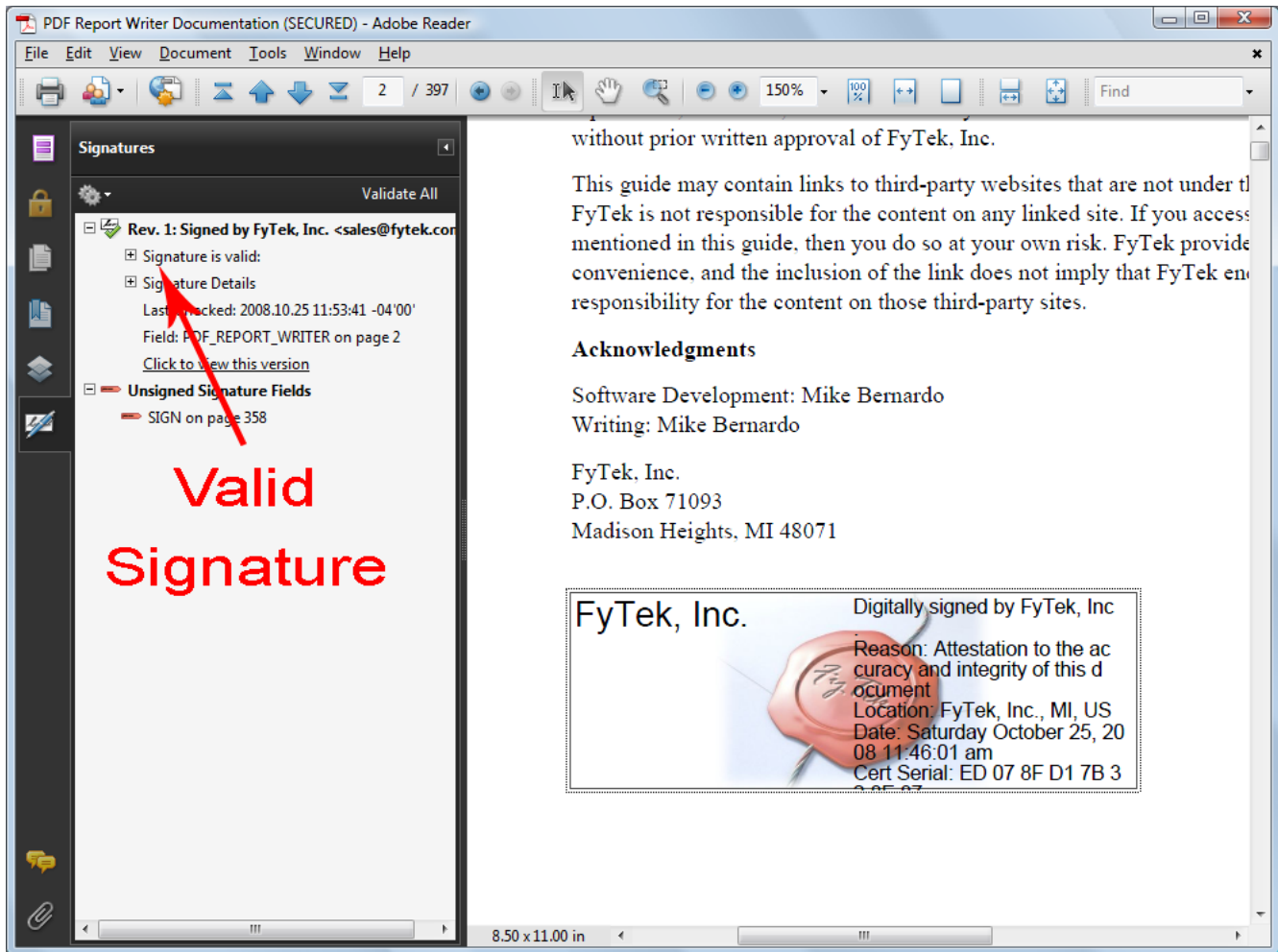
Click the "Validate Signatures" button to validate the signature we just setup the trusts for.

Digital Signatures



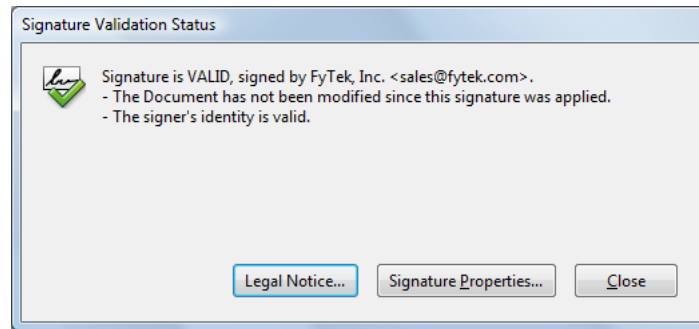
A green check icon now shows in the signature properties dialog.

Digital Signatures



A green check icon also shows in the signature pane. All future signings using this certificate will be trusted. The signature pane on the left will show what signings have taken place on the document and what signatures are open for signing. In this case there is one signature so far but an open signature box remains. You may also follow through on the dialog boxes by clicking the second signature (once signed) to view any changes to the document that happened between the time of the first and second signature.

Digital Signatures



This is what you will see now when you first click the signature field, assuming the signature is valid and the PDF has not been tampered with.

XPS Documents

XPS Document format is a paginated-document specification developed by Microsoft. This format is similar to PDF in the sense it is a finalized output not intended to be edited, unlike a document saved from a word processor where you can re-open and perform text or layout edits. As Vista and newer Windows operating systems roll out, you may want to save output in both PDF and XPS or give your users an option on what format they want for their reports.

Viewers for Windows XP are available from Microsoft at the site <http://www.microsoft.com/whdc/xps/viewxps.mspx>. The functionality is built into Microsoft Vista so no download is necessary as XPS documents will open in Internet Explorer 7 or higher on that platform

To create an XPS file (which is always in addition to the PDF output), use the `-xps` option or `SetXPSFile` method. You may use the `-xpsback` option or `SetXPSBackground` method to place a background on your XPS output. This is similar to the `-pdf` or `SetPDF` background option for PDFs.

If you have an existing PDF you want to use as an XPS background, the easiest way to convert it is to simply print the PDF as an XPS document. You'll need to install Microsoft's XPS Document Writer on XP based systems first. Simply open the PDF in Reader, select Print and choose the XPS Document Writer as the output. You'll be prompted for a file name to save as.

XPS Document files may be larger than PDFs in some cases, especially when the PDF doesn't contain any added images or fonts. The reason is the PDF viewer contains 14 built-in fonts so those do not need to be included in the PDF. XPS format, however, requires that fonts or subsets of fonts be included in all cases. The inclusion of the fonts is what makes some XPS files larger than PDF.

SQL Queries

SQL Queries are used to extract information from a data source using SQL commands. The SQL command file will contain tag-based commands such as [QUERY](#) and [LOOP](#). These commands are used to define the queries and actions to perform during iteration of the data. If you have a comma separated data file, for example, you can loop through the file and create a PDF page for each row in the file placing data where needed. The same method can be applied to database tables. In addition, you may use conditional statements to format as needed based on the results of the query.

The `-sqlquery` command line option or `SetSQLQuery` method is used to pass in the contents of the SQL command file. Contained within the SQL command file are one or more `QUERY` tags (select statements), `LOOP` statements (which iterate over the results), `PUT` statements to specify the row/column to place a particular field and a `PAGE` tag to denote where a new page should begin. Some commands begin with `RW` as they are shared with FyTek's PDF Report Writer command set and function in the same way.

The [PUT](#) tag is used to write a data element to the PDF page - either static text or a variable. Pound signs are used to enclose column names from queries when you want to display the data from the query. For example, if you have a column named "city" in a query named "customers" you would use `#customer.city#` between an opening and closing `PUT` tag. The string `#customers.city#` will be replaced with the data for the current record. The syntax in the input query file for PDF Forms would look something like:
<PUT COL=5 MAXLENGTH=15>#customers.city#</PUT>

In calculations or commands like [RWGET](#), use `$customers.city` instead. The reason for this is to prevent issues with embedded quotes. With the above example, `#customers.city#` is the same as the tag `<RWGET $customers.city>`. Both will return the same value. In a calculation, using `<RWIF #customers.city# ne "">`, the program will perform the replacement first and you may have something like `<RWIF Los Angeles ne "">` which will not work as there are no quotes around Los Angeles. When using `<RWIF $customers.city ne "">` instead, the program will use a variable with the value equal to Los Angeles during the evaluation process and work as expected.

You may define functions within your code and perform if/then/else processing as well. All other interaction with data from queries and in functions use Perl syntax. Perl has three data types you should be aware of.

SQL Queries

The first is a scalar which is used for numbers or strings. Variables of this type will all start with a \$ as in \$myvar. The second is an array type. Variables of this type will all start with a @ such as @myarray. The last is a hash type and those start with a %. It is beyond the scope of this documentation to cover programming in Perl. If you are at all familiar with JavaScript then much of the syntax will look similar.

The sample program fytek.sql that comes bundled with PDF Forms shows how to use many of these options. Use -none as the input file in this case. From the command line, you would run:

```
pdfforms -none fytekout.pdf -sqlquery fytek.sql  
-open
```

Some common operators are:

<u>Operator</u>	<u>Description</u>
+	Addition
-	Subtraction
*	Multiplication
/	Division
\$a = \$b	Assignment
\$a == \$b	Check if two numbers are equal
\$a != \$b	Check if two numbers are not equal
\$a eq \$b	Check if two strings are equal
\$a ne \$b	Check if two strings are not equal
\$a lt \$b	Check if a is less than b (in Perl, you can use < but since < also starts a tag, lt should be used instead)
\$a gt \$b	Check if a is greater than b
\$a le \$b	Check if a is less than or equal to b
\$a ge \$b	Check if a is greater than or equal to b
cond1 cond2	Logical OR
cond1 && cond2	Logical AND

The [LOOP](#) tag is the indicator to PDF Forms that the following content should be repeated for the specified query. Each row of the query result is one iteration of the LOOP section. Within the LOOP section you would place commands such as PUT to display the query fields.

Use the -debug command line option (SetDebug DLL method) to report on Perl errors. The debug output will show any queries it processed and the time to process along with the number of rows returned.

Database Connection

The database connection, specified with the `-sqldb` or `SetSQLDB` method, is used to set the parameters necessary for connecting to your database. Each type of connection has different options. In addition, you may specify the connection options on the [QUERY](#) tag. The `-sqldriver` option or `SetSQLDriver` method should be set to one of the below drivers (such as "Oracle"). Based on the driver below, determine what you should pass to `-sqldb` or `SetSQLDB` based on the description.

CSV

Specify the directory where the files are located with the syntax `"f_dir=directory"` where "directory" is the path to the files. To specify the current directory you would use `"f_dir=."` Use `"csv_sep_char="` to specify the field separator when other than a comma. For example, `"f_dir=.;csv_sep_char=|"` will set the pipe character as the field separator. Assign your tables using the [RWTABLE](#) tag.

Oracle (Windows Only)

Pass just the database or leave the option off and pass as part of the user such as `"username@XE"`. Or, you may pass the host, SID and port as in `"host=localhost;sid=XE;port=1521"`. Oracle version 8 through 11g should be able to connect.

mysql (Windows Only)

Specify the database name and the host IP address. For example, `"database=mydb;host=localhost"`. If necessary, you may specify the port as well such as `"database=mydb;host=localhost;port=3306"`. Both `mysql` and `mysqlPP` work against MySQL databases though `mysql` may give slightly better performance on Windows systems.

mysqlPP

Specify the database name and the host IP address. For example, `"database=mydb;host=localhost"`. If necessary, you may specify the port as well such as `"database=mydb;host=localhost;port=3306"`. Both `mysql` and `mysqlPP` work against MySQL databases though `mysql` may give slightly better performance on Windows systems.

ODBC (Windows Only)

Specify the DSN you wish to connect.

XML

You do not need to pass anything when using XML. Assign your tables using the [RWTABLE](#) tag.

```
<RWTABLE  
  FILE=text  
  TABLE=text  
  FORMAT=text  
  FLAGS=text  
  COLNAMES=text  
  EOL=text>
```

Only used when using CSV or XML files as the data source. You do not need this for other databases such as Oracle or MySQL.

Used to define a mapping between files on disk and logical table names in queries. For example, you might want to map a file called "employees.csv" to the table "empl". This allows you to then refer to the table "empl" in your queries. Add one of these tags for each table you will need.

<u>Parameter</u>	<u>Description</u>
FILE=text	The name of the physical file on disk. This may also be a web page that starts with http://.
TABLE=text	The logical name for this data to use in the QUERY tag. This is the name to use as the name of the table.
FORMAT=text	Set to one of the following: CSV XML
FLAGS=text	Use to specify the root node for XML. Specify by using the string record_type=>'node'. For example, if the nodes are called item that you are interested in looping through, set FLAGS="record_type=>'item'".
COLNAMES=text	Optional. Only include if the file you are using does not contain field names in the first row of the file. Set to a space separated list of column names you want to use to refer to each column in the file. For example, COLNAMES="name addr1 addr2 city state zip".
EOL=text	Optional. The End Of Line separator for CSV files. Set to \r\n for DOS files or \n for Unix formatted files.


```
<QUERY
  NAME=text
  PROCEDURE=text
  SQLDRIVER=text
  SQLDB=text
  USERID=text
  PASSWORD=text>
</QUERY>
```

Used to define a query. Place your SQL select statement between the opening and closing QUERY tags. Only the NAME is required. The connection options may be passed on the command line or via DLL methods. You may enter connection options on the query tag if you want to use different settings.

Use the PROCEDURE option to provide the procedure name when calling a stored procedure. In this case, rather than SQL, you place [QPARAM](#) tags between the opening and closing QUERY tags. Each [QPARAM](#) tag corresponds to a variable passed in or out of the procedure.

Use a select statement to return a value from a database function. For example, if you have an Oracle function named myfunct that takes two parameters as input you would write the select like this:

```
<QUERY NAME="runfunct">
select myfunct(<QPARAM $var1>, <QPARAM $var2>) res from dual
</QUERY>
```

You can then refer to the result with the variable \$runfunct.res in your code.

<u>Parameter</u>	<u>Description</u>
NAME=text	Provide a unique name for the query. The name is used in the LOOP statement to iterate over the result set.
PROCEDURE=text	Only specify this if you are calling a stored procedure. Provide the name of the procedure only. The parameters will be passed using the QPARAM tags before the closing </QUERY> tag. For example, if you have a stored procedure in Oracle named VALID_USER(ID NUMBER, LEVL OUT VARCHAR2) you would set PROCEDURE="VALID_USER" (or PROCEDURE="SCHEMA.VALID_USER" where SCHEMA is the schema for the procedure). You would then provide two QPARAM tags - the first for ID (the input) and the second of LEVL (the output).

SQL Queries

<u>Parameter</u>	<u>Description</u>
SQLDRIVER=text	<p>The data source. This is a case-sensitive string. Entries with a * are only available for Windows operating systems. Valid values are:</p> <ul style="list-style-type: none">CSVOracle*mysql*mysqlPPODBC*XML <p>mysql may give slightly better performance over mysqlPP on Windows systems.</p>
SQLDB=text	<p>The database schema or driver information. See the Database Connection section for details.</p>
USERID=text	<p>The user id (if any) for the database connection. For Oracle, you may also specify the password and/or schema in this field. For example, "user/pwd" or "user/pwd@prod".</p>
PASSWORD=text	<p>The password (if any) for the database connection.</p>

<QPARAM variable>

-- or --

**<QPARAM
TYPE=text
VAR=text
VALUE=text>
</QPARAM>**

Used to prevent SQL injection by passing parameters to the query during execution rather than placing directly in the SQL statement. The "variable" is the variable you want to use, either \$var (from Perl) or \$query.column (from a prior query) syntax. This option is also used to pass parameters with stored procedures.

For example, when you do this:

```
<QUERY NAME="getCities">
select city, state, zip from cities
  where company = '<RWGET $comp>'
</QUERY>
```

The result that is parsed is this which is not valid because of the quotes:

```
<QUERY NAME="getCities">
select city, state, zip from cities
  where company = 'O'Reilly'
</QUERY>
```

When you use QPARAM like this:

```
<QUERY NAME="getCities">
select city, state, zip from cities
  where company = <QPARAM $comp>
</QUERY>
```

The result is now this with a placeholder where the value will go:

```
<QUERY NAME="getCities">
select city, state, zip from cities
  where company = ?
</QUERY>
```

The ? is used as a placeholder that is evaluated later during processing. It is replaced by the string "O'Reilly" when needed. This also protects against unwanted commands being passed if your variables are based on user input.

For stored procedures, there are several other options for QPARAM. For example, assume you have a stored procedure in Oracle setup as VALID_USER(ID NUMBER, LEVL OUT VARCHAR2). You would then provide two QPARAM tags for the parameters. For example: The result that is parsed is this which is not valid because of the quotes:

```
<QUERY NAME="chkUser" PROCEDURE="VALID_USER">
```

SQL Queries

```
<QPARAM TYPE="IN" VAR="p_id" VALUE="100">
<QPARAM TYPE="OUT" VAR="p_level" VALUE="$plevel">
</QUERY>
```

The value for ID is set to 100 in this example. This could also be a Perl value that was set from a prior RWSET or a value from a prior query if you like. In that case, you would have a variable such as \$pid or \$myquery.column. The value output from the query will be placed in the Perl variable \$plevel. You can then display it or use \$plevel in other functions as if you defined it with an RWSET statement.

<u>Parameter</u>	<u>Description</u>
TYPE=text	Set to IN, OUT or INOUT. This is only used for stored procedure. Represents the type of parameter this is for the procedure.
VAR=text	The variable placeholder name for the stored procedure. This is simply used to name the placeholder so it does not have to correspond to any prior variable. Each VAR should be unique though within the QUERY section.
VALUE=text	The value to pass in specified as either a hard-coded value or a Perl variable that has been initialized. Or, this is the name of the variable to receive the output of the stored procedure. Output or input-output parameters should always specify a Perl variable in the form \$name where "name" is the name of your variable.

```
<LOOP
  QUERY=text
  FROM=number
  TO=number
  -- or ---
  INDEX=text
  ARRAY=text
  SORTSTR[=text]
  SORTNUM[=text]
  SORTDESC
  -- or ---
  INDEX=text
  LIST=text
  DELIMITER=text
  -- or ---
  INDEX=text
  FILE=text>
</LOOP>
```

Used to loop over the result set for the named query, an array, a list, or a file. Use the # symbol to enclose variables you want replaced by the query results.

For example, if your query looks like this:

```
<QUERY NAME="getCities">
select city, state, zip from cities
</QUERY>
```

You can use #getCities.city#, #getCities.state#, or #getCities.zip# as variables for display between the opening and closing LOOP tags. Be sure to use the name of the query followed by a dot then the field name.

In addition, you can use \$getCities.city in any Perl code or [QPARAM](#) statements. The query.column syntax used in any Perl code will be converted to a variable holding the field value.

You can also nest [QUERY](#) and LOOP commands inside of a loop. This allows you base lower level queries on an upper level one. You may use variables in your SQL when inside a loop to create a dynamic query. For example:

```
<QUERY NAME="getCities">
select city, state, zip from cities
</QUERY>
<LOOP QUERY="getCities">
  ...other commands (can use #getCities# here)...
  <QUERY NAME="getBusinesses">
    select business_name, business_type from businesses
```

SQL Queries

```
    where city = <QPARAM $getCities.city>
    order by business_name
</QUERY>
<LOOP QUERY="getBusinesses">
    ...other commands (can use #getCities#
        or #getBusinesses# here)...
</LOOP>
...other commands (can use #getCities# here)...
</LOOP>
```

If you just need to output a variable and don't need to loop over anything, use the LOOP command without any options to display the value or perform command processing. For example, if you just want to print the value of a variable passed in you may do this:

```
<LOOP>
<TEXT>
    inpvar = <RWGET defined($inpvar)>
</TEXT>
</LOOP>
```

Note the LOOP command goes outside of the TEXT or TD tags. If you are already inside an outer loop, there is no need to an inner loop. Without the LOOP command above, the text would display the RWGET command rather than process it.

<u>Parameter</u>	<u>Description</u>
QUERY=text	This is the only mandatory option when looping through a query result set. Provide the name of a previously defined QUERY statement you wish to loop through.
FROM=number	The starting row number (default is the first).
TO=number	The ending row number (default is the last).
INDEX=text	The variable name when looping through an array, list, or file. Use a \$ in front of the variable name. For example, \$i.
ARRAY=text	The array to loop through. This may be a simple array or an array of hashes. The LOOP command would be: <RWSET @ary = ("apple","orange","banana")> <LOOP INDEX="\$idx" ARRAY=@ary>
SORTSTR[=text]	Sort the array as a list of strings. Optionally, if you have an array of hashes, specify the hash column.
SORTNUM[=text]	Sort the array numerically. Optionally, if you have an array of hashes, specify the hash column.
SORTDESC	Sort in decending rather than the default of ascending.

SQL Queries

<u>Parameter</u>	<u>Description</u>
LIST=text	The list to loop through. A list can be a variable or static list. For example, LIST="a,b,c" or LIST="\$mylist". The LOOP command would be: <LOOP INDEX="\$idx" LIST="a,b,c">
DELIMITER=text	The delimiter to use for the text. The default is a comma.
FILE=text	The file to loop through. Each line of the file is one iteration of the loop. The LOOP command would be: <LOOP INDEX="\$idx" FILE="c:\temp\myfile.dat">

```
<PUT
  ROW=number
  COL=number
  ALIGN=text
  MAXLENGTH=number
  MINROWS=number
  MAXROWS=number>
</PUT>
-- or --
<PUT
  X=number
  Y=number
  MAXLENGTH=number
  FACE=text
  SIZE=number
  COMP=number
  COLOR=text>
</PUT>
```

This tag is used to output a variable or static text. The contents to display go between the opening and closing PUT tags. Specify the ROW/COL when placing text in the normal flow of the page. The standard document font and size will be applied. Use the X/Y values instead to place text at a specific position on the page along with the settings for size, color, and font type.

<u>Parameter</u>	<u>Description</u>
ROW=number	Optional. The row (or line) number for the output. The current line based on the ROW tag or current position is used if this is not specified.
COL=number	The column number for the left edge of the output.
ALIGN=text	Optional. Set to Right to right align the text. In this case, COL should be the rightmost column for the data rather than the leftmost. For instance, if you want a string of text to be placed within columns 2-15 and right aligned, set COL=15 rather than 2.
MINROWS=number	Optional. The minimum number of ROWS to take up.
MAXROWS=number	Optional. The maximum number of ROWS to take up. If set, text will wrap based on the MAXLENGTH value for the number of rows needed. Text is split on the space character. If the text still overflows on the last line (the remaining text is longer than MAXLENGTH), it is simply truncated.

PDF Forms

SQL Queries

<u>Parameter</u>	<u>Description</u>
MAXLENGTH=number	The maximum length of the output. The output is truncated to be this size if it is longer.
X=number	Left position for the text in points (1/72 of an inch) or current units setting.
Y=number	Top position to start the text in points (1/72 of an inch) or current units setting.
FACE=text	The font number or a font file on disk. 1 - Courier 2 - Courier Bold 3 - Courier Italics 4 - Courier Bold-Italics 5 - Helvetica 6 - Helvetica Bold 7 - Helvetica Italics 8 - Helvetica Bold-Italics 9 - Times Roman 10 - Times Roman Bold 11 - Times Roman Italics 12 - Times Roman Bold-Italics 13 - Symbol 14 - Zapf Dingbats 15 - 3 of 9 Barcode 16 - UPC Size 1 17 - UPC Size 2 18 - UPC Size 3
SIZE=number	The font point size.
COMP=number	The text compression percentage as a whole number (for example, enter 80 for 80%).
COLOR=text	The color for the text.

SQL Queries

**<ROW
VALUE=number
INC=number>**

This tag is used to specify the starting row (or line) number on the page. The PUT tags will default to this starting position. Note the ROW and [NEXTROW](#) tags have to do with text placement on the page and do not refer to query rows.

<u>Parameter</u>	<u>Description</u>
ROW=number	The row (or line) number to use for the next PUT statement.
INC=number	Optional. The number of rows to increment by when a NEXTROW tag is encountered. Defaults to 1 if not specified.

**<NEXTROW
INC=number>**

Simulates one or more line-feeds, depending on what value was set for INC on the last [ROW](#) command. This tag is used to move to the next row on the page - that is, drop down 1 or more lines from the current position. The INC value from the ROW tag is used to determine the number of rows to advance.

You may also set the number of rows to advance which will override the INC setting.

<u>Parameter</u>	<u>Description</u>
INC=number	Optional. The number of rows to increment. The INC value of the last ROW tag will be used if this is not specified. If INC was not specified either, then this defaults to 1.

SQL Queries

**<UNITS
VALUE="text">**

The UNITS tag is used to set the unit of measure for the PUT tag when using X/Y positioning.

<u>Option</u>	<u>Description</u>
VALUE="text"	Used to set the unit of measure. Default is points. Set this option first so latter PUT tags use the proper unit setting. Valid settings are: pt - points (1/72 of an inch) in - inches cm - centimeters mm - millimeters

<PAGE>

This tag is used to specify a page break. Normally this would go just before the outermost closing LOOP tag though you can place it at any point you want a page break to occur. This option adds a form-feed character to the output.

```
<RWSUB  
  NAME=text>  
</RWSUB>
```

Defines a subroutine for commonly used [PUT](#) or other statements. Place statements between the opening and closing RWSUB tags that you want to use in your input file. For example, you might want to place heading data to print here when you have conditional page breaks. That way, you can call the routine initially then call it again after a PAGE command so you are not repeating the same set of PUT statements in more than place in your input file.

<u>Parameter</u>	<u>Description</u>
NAME=text	The name of the subroutine. Use this name in the RWRUN statement to call the routine.

SQL Queries

**<RWRUN
NAME=text>**

Calls the named subroutine defined with an earlier [RWSUB](#) tag.

<u>Parameter</u>	<u>Description</u>
NAME=text	The name of the subroutine to run. See the RWSUB tag.

<RWSET variable=text>

Used to assign some value to a variable. The variable can be a string, array, or hash. For example, `$i = 1` will set the value of `$i` to 1. Assuming we have a query named `customers` that contains a column called `city`, then `$c = $customers.city` will set `$c` to the current value of `city`. The current value is based on what row we happen to be on. In this case, the `RWSET` statement would be inside of the [LOOP](#) that iterates over the query result set. You may use any valid Perl assignments or short-hand notation such as `$i += 1` which sets `$i` to itself plus 1.

<RWGET text>

Retrieves a variable for display which may be a Perl variable or a query column value. In addition, you may use this tag to perform other functions before displaying the value. For example, <RWGET \$c> will retrieve the value for \$c. Use the notation \$query.column when referring to a column from a query rather than #query.column#. You may also use <RWGET currency_format(\$customer.payment)>. In this case, it is assumed you have a Perl function defined in the input called currency_format. This function will take the value of payment from the customer query and perform some type of formatting. The result is then displayed in the report.

You do not need to use RWGET in other statements such as [RWSET](#). This statement is only a wrapper for applying additional formatting before display or to display user defined variables set with RWSET.

```
<RWIF condition>  
  ...statements...  
  [<RWELSEIF condition>  
    ...statements...]  
  [<RWELSE>  
    ...statements...]  
</RWIF>
```

This statement is used to provide conditional processing. Any valid Perl constructs may be used to create the condition to check. Do not use pound signs (#) around variable names when using as part of the if statement. You may use also use lt for <, gt for >, lte for <=, and gte for >=. This is so PDF Forms does not match the < or > symbols as the end of the tag. You may nest other RWIF statements as part of the ...statements... above. Also, you may include all the other commands such as [RWGET/RWSET](#) as well as [LOOP](#) and [QUERY](#). This allows you to conditionally parse other queries and loop through the results.

This example shows what you might display based on the current customer's balance.

```
<RWIF $customer.balance gt 0>  
  <QUERY NAME="inv">  
    select invoice, due_date from invoices  
      where id = '#customer.id#'  
  </QUERY>  
  <LOOP QUERY="inv">  
    <PUT>Inv#: #inv.invoice# Due: #due_date#</PUT>  
    <NEXTROW>  
  </LOOP>  
<RELSEIF $customer.balance lt 0>  
  <PUT>Refund is due to customer.</PUT>  
<RELSE>  
  <PUT>Customer does not have a balance.</PUT>  
</RWIF>
```

<RWSCRIPT> </RWSCRIPT>

Used to define Perl values or sub-routines that can be called with the [RWGET](#) tag. Place any valid Perl syntax between the opening and closing RWSCRIPT tags.

The function names are case-sensitive. For example, this script places commas in a number to format it for display.

```
<RWSCRIPT>
  sub addcommas {
    my $input = shift;
    $input = reverse $input;
    $input =~ s<(\d\d\d)(?=\d)(?!\\d*\\.)><$1,>g;
    return reverse $input;
  }
</RWSCRIPT>
```

LineNum

"linenum" is a built-in function that returns the current line number on the page you are positioned at. You can use this function to determine if you need to page break when dealing with a detailed list such as on a billing statement or order line listing. For example:

```
<LOOP QUERY="cities">  
  <RWIF linenum() gt 25>  
    <PAGE>  
  </RWIF>  
  <PUT COL=10>#cities.name#</PUT>  
  <NEXTROW>  
</LOOP>
```

RecordNum

"recordnum" is a built-in function that returns the current row number you are on while looping through a query. Specify the query name followed by a period then the text recordnum. It is treated as a column name that is available in all queries. For example:

```
<LOOP QUERY="cities">  
  <PUT COL=10>#cities.recordnum#</PUT>  
  <PUT COL=20>  
ALLOWPERL was not specified</PUT>  
</LOOP>
```

SQL Queries

RecordCount

"recordcount" is a built-in function that returns the total number of rows matched by a query. Specify the query name followed by a period then the text recordcount. It is treated as a column name that is available in all queries. For example:

```
<PUT COL=10>#cities.recordcount#</PUT>  
<PUT COL=20>  
ALLOWPERL was not specified</PUT>
```

first(text)

"first" is a built-in function to determine if you are on the first iteration of your query. Provide the query and column you want to check in the form query.column. If you are at the first record then the result is true. Pass in the column you want to check.

For example, <RWIF first('customer.state')> would result in true if this is the first record in the loop. You should sort your results by using an order clause in a way that this function can use it. In this case, we would be sorting customers by state. As we loop through the query, the first iteration of the loop will result a value of true.

firstof(text)

"firstof" is a built-in function to determine if you are on the first iteration of a particular value in your query. Provide the query and column you want to check in the form query.column. If you are at the first record or the prior record had a different value for the field then the result is true. Pass in the column you want to check.

For example, <RWIF firstof('customer.state')> would result in true if this state is the first for it's value. You should sort your results by using an order clause in a way that this function can use it. In this case, we would be sorting customers by state. As we loop through the query, the first time we hit any given state this results to a value of true. If we have 20 entries for California, the first entry we come across for CA will result in this being true. The next 19 entries for CA will result in this result being false.

last(text)

"last" is a built-in function to determine if you are on the last iteration of your query. Provide the query and column you want to check in the form query.column. If you are at the last record then the result is true. Pass in the column you want to check.

For example, <RWIF last('customer.state')> would result in true if this is the last record in the loop. You should sort your results by using an order clause in a way that this function can use it. In this case, we would be sorting customers by state. As we loop through the query, the last iteration of the loop will result a value of true.

lastof(text)

"lastof" is a built-in function to determine if you are on the last iteration of a particular value in your query. Provide the query and column you want to check in the form `query.column`. If you are at the last record or the prior record had a different value for the field then the result is true. Pass in the column you want to check.

For example, `<RWIF lastof('customer.state')>` would result in true if this state is the last for its value. You should sort your results by using an order clause in a way that this function can use it. In this case, we would be sorting customers by state. As we loop through the query, the last time we hit any given state this results to a value of true. If we have 20 entries for California, the last entry we come across for CA will result in this being true. The previous 19 entries for CA will result in this result being false.

removespaces(text)

"removespaces" is a built-in function to remove extra spaces and line-feeds from text. Use this when you have XML data, for example, that spans several lines in the XML file. This function will remove spaces from the front and back of the text as well as collapse any multiple spaces within the text down to one. For example:

```
<LOOP QUERY="books" >  
  <PUT><RWGET removespaces($books.description)></PUT>  
</LOOP>
```