



*Text2PDF*

*FyTek, Inc.*

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

## **FyTek's Text2PDF**

### **Trademarks**

FyTek, FyTek Text2PDF 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 © 2000-2022 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

Text2PDF is a program to convert plain text files to PDF (Portable Document Format) files that can be opened and printed by anyone with software such as the free Adobe Reader. In addition, you may also create XPS (Microsoft's XML Paper Specification format) files as output. XPS is the new format from Microsoft that is similar to PDF. Most, if not all, of the layout from PDF is retained in XPS. Some PDF specific options such as encryption are not part of XPS currently.

Certain tags (commands entered between < and > characters) may be used for changing font face or size, coloring, images, bookmarks, and more. The tags are placed in the input file (or commands from the DLL) along with the text to render. This document describes the commands available and what parameters may be passed to the program to create a PDF (Portable Document Format) file.

The executable version of Text2PDF can be setup to run either stand-alone or in server mode. Under Windows, Text2PDF can be setup in server mode as a Windows service allowing users to build their reports centrally off of one computer. Unix users can setup the server in a similar fashion by starting the server as a background process. Requests are sent to the program via TCP/IP when running in server mode. See the [Client-Server](#) section for detailed information on this type of setup. The DLL text2pdf\_20.dll is both a 32-bit and 64-bit .NET DLL. Use the Microsoft utility regasm to register it as a COM DLL as well. With it you may embed the functionality Text2PDF in the programming language of your choice that is able to access a .NET or COM DLL. You may also process requests where Text2PDF is running a Linux box for example. Or, if your license agreement allows, have multiple Text2PDF servers running on different boxes and easily access them through the DLL.

The program text2pdf.exe (or text2pdf64.exe) is the executable version and is run from the DOS prompt or you can use in a DOS batch file. Similarly, text2pdf is the Linux/Unix executable version of the program. If you are running Text2PDF in server mode, clients can use the program text2pdf\_tcp or text2pdf\_gui\_tcp (or their 64-bit equivalents). The file text2pdf.dll is the standard DLL and text2pdfdn.dll is the .NET DLL. The DLL versions contain the same functions but are meant to be called from another application such as Visual Basic, ASP, C or Java. You use options on the command line with the executable, like -open, call methods with the DLL version, like AutoOpen.

Text2PDF takes text and processes it, looking for line breaks unless otherwise specified, and creates a PDF from that text. Text may be either "plain" text — that is, no markup tags or other information — or it may

## Introduction

contain various tags to control the font face or size and other features. Preformatted plain text will have a line feed character at the end of each line that Text2PDF will use to break on. Free flowing text will fill a line and Text2PDF will decide when it's necessary to break. The line feeds are treated as white space in this case and not used to break lines, though tags such as `<BR>` can be used to force a line break. With the use of options like `-pre [plain|html|white]` (or the DLL method `Pre`) you can specify the file format. There is also a tag to control this setting.

An [initialization file](#) may be used to perform text replacement in the input. You can use this feature to convert any incoming codes to other text or tags for the output. For example, escape sequences in the input file can be converted to Text2PDF tags for text underline, font/color changes or any other text conversion. This can also be used to simply remove the codes or text from the input.

Text can be standard ASCII or Unicode. Unicode text must be formatted as UTF-8 or use the syntax `&#9999;` where 9999 is the decimal Unicode value, `&#x9999;` where 9999 is the hexadecimal Unicode value or `&#o9999;` where 9999 is the octal Unicode value. For example, `&#1575;` is the same as `&#x0627;` and `&#o3047.` Be sure to use the `-utf8` option or UTF8 method when your input contains this syntax for Unicode. In addition, you'll need to supply a font (using the [FONT](#) tag) that contains the required Unicode characters. An example would be the `arial.ttf` font that is supplied with Windows.

You can encrypt and optionally password protect your output PDF. The `-o` and `-u` options with the executable set the owner and user passwords. Use the `setOwner` and `setUser` methods with the DLL. In addition, you can control what rights are enabled with `-noprint`, `-nochange`, `-nocopy`, `-noannotate`, `-nofillin`, `-noextract`, `-noassemble` and `-nodigital`. Use the `-e128` option or `setEncrypt128` method to specify 128-bit encryption. You may also use `-aes 128` or `-aes 256` for AES encryption.

## Using the Executable

The executable is designed to run from a command prompt. The server version of the executable can run without any user interaction in a batch/script file or on a web site. The program is named text2pdf.exe on Windows systems and text2pdf on Linux/Unix. Be sure to mark text2pdf as executable on Unix/Linux systems with the chmod command (for example, "chmod 775 text2pdf").

The program text2pdf.exe is used to create the PDF from an existing file or the Windows clipboard. You may create the file using a text editor or use another application to populate it. Using the latter approach you can create PDFs dynamically from any application that can create a text file. Execute the program once the file is created by running:

```
text2pdf.exe filein.txt fileout.pdf
```

where "filein.txt" is the name of your input file and "fileout.pdf" is the PDF output. You may also use directory names in place of the input and output file. All documents in the input directory matching the file mask set with the -mask parameter will be converted to PDF.

You may also convert what's on the Windows clipboard. Execute the program after a file, text or an image has been copied to clipboard by running:

```
text2pdf.exe -clip fileout.pdf
```

You may also provide any of the other command line options such as -pre, -open, -pagew, etc. The output file name is optional and is not used if there are a list of files on the clipboard (such as from doing a copy on multiple files from Windows Explorer). If there are a list of files on the clipboard, the output PDFs are named based on the input file name and are placed in the temp directory. The default name for text or bitmap images on the clipboard converted to PDF is t2p###.pdf where the #'s represent a unique number (may be more than 3 numbers). Again, this file will be in the temp directory.

Set up a shortcut on your desktop with the common settings you use along with the -clip option to quickly create a PDF from what you have stored on the clipboard. Note that text is converted as plain text without any special formatting from the source application (like bolding or color information from MS Word).

Use the "-pre" option on the command line if your text input is preformatted for line breaks and form feeds. Use "-pre plain" for faster processing of text without tags when you don't want the program to line break at the right margin. Use "-pre html" for text with tags when you still don't want the

### *Executable*

program to line break at the right margin. Use "-pre white" for text with tags when you want the program to line break at the right margin and you want to preserve white space within the text. The plain and html options work best for legacy or mainframe type reports which usually print on greenbar paper. Be sure to select a courier or fixed spaced font in this case.

Use the "-c1" option on the command line if your text input contains carriage control and form feed information in column one. Like the -pre option, the -c1 option is primarily for legacy or mainframe type reports. Do not use the -pre option along with -c1 as it will override the -c1 setting. This type of input is typically fixed size font (courier) and uses the following codes in the first column of the input data:

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

Running text2pdf.exe with no parameters will bring up a file open dialog box and allow you select an input file. You may also send some parameters with no file names to apply those options to the file. For example, "text2pdf.exe -pre" or "text2pdf.exe -pre plain" to use the file open dialog box and apply a preformatted option.

You may also send input to the program from standard input (STDIN). Use a dash as the file name in this case. For example:

```
"text2pdf - myfile.pdf"
```

Then type in some text and end with a single dot or ctrl-D (ctrl-Z on DOS). In DOS, you'll need to set the program as a console program first using wintype.exe (available for free from our web site - see the FAQ section at <http://www.fytek.com> to download and for instructions). Using this approach you can also pipe a file into the program like this:

```
"cat sample.txt | text2pdf - myfile.pdf -force"
```

or, in DOS, like this:

```
"type sample.txt | text2pdf.exe - myfile.pdf -force"
```

You'll be prompted (via a dialog box) for your input and output files names when running under Windows if you leave out the input and output file names.

Other options you can pass to text2pdf are:

## Executable

-aes 128 256	Sets AES encryption method. Pass 128 for 128-bit encryption or 256 for 256-bit encryption. Files encrypted with AES 128-bit encryption can only be opened with Acrobat or Adobe Reader 7.0 or above. Files encrypted with AES 256-bit encryption can only be opened with Acrobat or Adobe Reader 9.0 or above.
-align <i>L/R/C/J</i>	The alignment to use for the text. The options are: L = Left (default) R = Right C = Center J = Justify
-author <i>text</i>	Sets the document author.
-autosize [ <i>lines</i> ]	Used to automatically set the font point size and text width compression. Use this option along with -pre plain or -c1 options to check the input and reduce the font size if necessary. The "lines" value is optional. If used, set to the number of lines to read from the input file to determine the optimal font size. For example, 100 to read the first 100 lines from the input file and assume the rest of the file is set up in the same manner. Set to at least one page worth of data or leave off to scan the entire input file. The font is never increased in size, only reduced so you should set the point size to the maximum point size you wish to use.
-b1 <i>text</i>	Sets the text for button 1 (default is "Cancel").
-b2 <i>text</i>	Sets the text for button 2 (default is "Break on next page").
-barcodes	Use this option when you are using one of the built-in barcode fonts.
-bkypass <i>password</i>	The owner password for the background PDF. This is only needed when the background PDF is encrypted.
-bm <i>number</i>	Bottom margin in inches or units. Default is .5 inches.

## Executable

<code>-border <i>width</i>[,<i>color</i>[,<i>padding</i>]]</code>	Used to draw a border on the page. The color and padding are optional. The width and padding are specified in points (1/72 of an inch). Any valid <a href="#">color</a> may be used for the border.
<code>-c1</code>	Used to specify the first column contains a vertical print instruction. The input is treated as preformatted text so no line breaks are inserted by the software. Be sure to select a courier or fixed spaced font 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
<code>-ckjwidth <i>number</i></code>	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.
<code>-clip</code>	The source is the current contents of the Windows clipboard. If multiple files are on the clipboard then one PDF will be created for each one. These PDFs will be found in the temp directory.
<code>-cols <i>number</i></code>	The number of text columns per page. Default is 1.
<code>-colsp <i>number</i></code>	The amount of spacing in inches or units when using two or more columns.



## Executable

<code>-comp <i>number</i></code>	The compression factor to apply to the text width. Default is 100. Smaller numbers make the text thinner while larger ones make it wider.
<code>-comp15</code>	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.
<code>-copies <i>number</i></code>	Number of copies to print when using the <code>-print</code> or <code>-printer</code> commands. Default is 1.
<code>-cwd <i>text</i></code>	Changes the working directory to the specified directory.
<code>-defwidth <i>number</i></code>	Enter the default width for base fonts. 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>-down <i>number</i></code>	The distance to move the contents of each page down. The number is in units of 1/72 of an inch or the unit setting. May be positive or negative value.
<code>-e128</code>	Sets 128-bit encryption method. Files encrypted with 128-bit encryption can only be opened with Acrobat or Adobe Reader 5.0 or above. The default encryption is 40-bit which works with Acrobat and Adobe Reader 4.0 and above.
<code>-fcolor <i>color</i></code>	Specifies the font color. See the <a href="#">color</a> section for valid values.

## Executable

-fit	Fits the page height to the contents of the page. The -pageh option, in this case, will be the maximum page size.
-font <i>fontname</i>	The font to use for the output PDF. For example, "courier", "helvetica" or "times".
-force <i>text</i>	Turns off the dialog box prompting to overwrite the output file if it exists. If a PDF by the same name as the output PDF already exists it is overwritten.
-forwardref	Used to specify the document contains forward references. Use this option when you have links in the document that reference anchors further in the document. For example, if you have <A HREF="#mylink"> on a page but don't have <A NAME="mylink"> until after that link, you'll want to use this option. This is so the program knows to scan for all the links first and then create the PDF. If all of your links are at the end of the document (like with an index page) then you don't need this option.
-greenbar "[ <i>top</i> ],[ <i>height</i> ],[ <i>color</i> ]"	Prints green bar lines on the page. <i>Top</i> is the starting position in inches (or units) from top of page. The default is 0. <i>Height</i> of bar is in inches (or units). The default is 1/2 inch. <i>Color</i> is the color of the bar, default is #C0FFC0.
-guioff	Suppresses the dialog window that shows the current build progress.

## Executable

- `-img "file,x,y[,scalex,scaley]"` Background image or watermark to use with the output PDF. Supply the file name (with full path), x position from the left edge of the page in points, y position from bottom edge of page in points. For example, "myfile.jpg",72,144 will place the lower left corner of image "myfile.jpg" 1 inch (72 points) from the left edge and 2 inches from the bottom. Optionally pass in a scaling factor for the x/y axis. The default for scalex and scaley is 100.
- `-ini path-file` An initialization file containing text replacement. Use this when you have a file with various codes for things such as underlines or colors. The file is processed just before the report build so it will override any duplicated command line options. This file can be used to specify the conversion between the text and tag. See the [Initialization File](#) section for details.
- `-inicmds path-file` An initialization file containing command line options. Pass the path and name of a file containing commands you would pass on the command line. For example, `-font 'courier' -open`. Do not add any special formatting such as commas between commands. The file should contain command line options as if you were passing the file contents directly to the executable program.
- `-inirun path-file` An initialization file containing text replacement. This is different from the `-ini` setting in that the settings are processed immediately. This allows you to override settings in this file by placing them further down in the command line options. Use this when you have a file with various codes for things such as underlines or colors. This file can be used to specify the conversion between the text and tag. See the [Initialization File](#) section for details.

## Executable

-keywords <i>text</i>	Sets the document keywords.
-lm <i>number</i>	Left margin in inches or units. Default is .5 inches.
-linspace <i>number</i>	The amount of space (always in points - 1/72 of an inch) to leave between lines. Default is 2.
-mail	Opens the user's email program to a composition window with the newly created PDF attached. May not work with all email programs. None of the other mail options (such as -mailsmtp) are necessary with this option. The default settings for to, cc and bcc addresses as well as the subject and body are taken from the options described below.
-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 an address in the form of name@somecompany.com. Separate multiple addresses with a comma.

## Executable

<code>-mailbody text</code>	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.
<code>-mailcc text</code>	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.
<code>-mailfakecc text</code>	The CC address to show for the email. The default is the CC address(es). (SMTP only)
<code>-mailfakefrom text</code>	The from address to show for the email. The default is the FROM address. (SMTP only)
<code>-mailfaketo text</code>	The to address to show for the email. The default is the TO address(es). (SMTP only)
<code>-mailfiles text</code>	A comma separated list of file names to include with the mailing. The path must be fully qualified for each file.
<code>-mailfrom text</code>	The from address for the email. Must be an address in the form of somename@mycompany.com. (SMTP only)
<code>-maillog text</code>	The name of a log file to use for date/time emails were sent as well as any errors. This is optional. (SMTP only)

## Executable

<code>-mailnodialog</code>	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 <code>-mailsmtp</code> option instead to send the email via SMTP without user intervention.
<code>-mailpri text</code>	The message priority. Set to either HIGH or LOW. Leave this option off for normal priority. (SMTP only)
<code>-mailreply text</code>	The reply to address for the email. Must be an address in the form of <code>somename@mycompany.com</code> . The default is the FROM address. (SMTP only)
<code>-mailretry number</code>	The number of times to retry sending the email if an error is encountered. The default is 0, meaning try sending the email only once.
<code>-mailscr file</code>	For Unix/Linux systems. 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.
<code>-mailsmtp text</code>	The SMTP server to use for sending the mail. Used to send the PDF via SMTP rather than interactively. For example, <code>mail.yourdomain.com</code> . 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, <code>mail.yourdomain.com:465</code> . Must also supply the <code>-mailfrom</code> and <code>-mailto</code> options. The <code>-mailfrom</code> must be a valid email account on the SMTP server.
<code>-mailsub text</code>	The subject of the email. Enclose in quotes.
<code>-mailto text</code>	The address(es) to send the email to. Must be an address in the form of <code>name@somecompany.com</code> . Separate multiple addresses with a comma.

## Executable

-mask <i>text</i>	File names to match when batch converting. Specify an input and output directory rather than input and output file. Use wildcards to match on file names. For example, *.txt will match all files with an extension of .txt. A mask of t*.dat will match all files starting with the letter t and having an extension of .dat.
-noannotate	Disables add/change of form fields or annotations.
-noassemble	(128-bit only) Disables assembly (insert, rotate, delete pages or create bookmarks) when -nochange is used.
-nochange	Disables changes to the document.
-nocopy	<i>40-bit :</i> Disables copying of text and/or graphics from the document. <i>128-bit :</i> Disables copying of text and/or graphics from the document other than in support of accessibility to disabled users or for other purposes.
-nodigital	(128-bit only) Disables printing at digital quality - can only print low resolution. The -noprint option overrides this option so you'll want to use -noprint or -nodigital but not both.
-noextract	(128-bit only) Disables extraction of information in support of accessibility to disabled users or for other purposes.
-nofillin	(128-bit only) Disables fill in interactive fields when -noannotate is used.

## Executable

-noprnt	<p>Disables printing of the document (even low resolution). To create a PDF with both printing and copying disabled for the user you would run something similar to:</p> <pre>text2pdf filein.txt fileout.pdf -o abc123 -u xyz -noprnt -nocopy</pre> <p>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.</p>
-norights	<p>Turns off all rights (default is all are granted). Setting of options such as -noprnt or -nocopy turns those rights on rather than off. Use this if you typically are turning off most or all of the rights. Note that setting -norights and -noprnt will allow high resolution printing. Setting -norights and -nodigital will allow low resolution printing. Setting only -norights will disallow printing.</p>
-nosoftbreak	<p>Only page break on a hard page break (like a PAGE command). Text may flow past the bottom margin if there is more than would normally fit on the page with this option.</p>
-np	<p>Turn off the box that shows how far along the program is in building the pdf.</p>
-o <i>password</i>	<p>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 -noprnt was specified, then it is impossible for the owner to print the document.</p>



## Executable

<code>-ooserver <i>server:port</i></code>	The server and port that OpenOffice server is running on. For when you have an OpenOffice server running to convert Excel/Word/etc. to PDF for use by Text2PDF. Pass the server and port number. For example, <code>-ooserver "myooserver:2050"</code> .
<code>-open</code>	Automatically opens Acrobat and loads the newly created PDF.
<code>-openscr <i>file</i></code>	For Unix/Linux systems. 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.
<code>-opt</code>	Optimize the output PDF for fast web viewing. Note this typically increases the size of the output by a few hundred bytes or so. The PDF is optimized for viewing on the web as opposed to shrinking the physical size. As a rough guide, you should only optimize a PDF that will be viewed on the web when it contains more than 10 pages and its size is bigger than a megabyte. Additionally, you must create the PDF to a file rather than stream the output to the browser. The setting "fast web view" will be set to yes for optimized PDFs when you open in Reader and check the properties. This means the first page of the PDF is sent to the user and made viewable while the rest of the pages continue to download in the background.
<code>-pageh <i>number</i></code>	The height of the page in inches or units. Default is 11 inches.

## Executable

`-pagenumfont[i] font,pointsize`

The [i] is optional - it allows for more than one entry. For example, `-pagenumfont "times,8"` `-pagenumfont2 "times,9"` and so on. You'll need a corresponding `-pagestr#` for each `-pagenumfont#`. Sets the font and point size for the page numbering string. Pass the font name (must be one of the standard Courier, Helvetica or Times fonts) along with the point size. For example, `-pagenumfont times,9` will use Times Roman at a point size of 9. See the [FONT](#) tag for a list of the fonts.

You may also pass in a tag based setting that matches the format of the `<FONT>` tag. This allows you to set your own TrueType or OpenType font if you wish. In this case, pass only one parameter. For example, `-pagenumfont "<FONT SRC='myfont.ttf' SIZE=20>"`.

`-pagepos[i] x,y`

The [i] is optional - it allows for more than one entry. For example, `-pagepos "2,5"` `-pagepos2 "3,1"` and so on. You'll need a corresponding `-pagestr#` for each `-pagepos#`. 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 `-pagestr` is used without this option.

## Executable

<code>-pagestr[i] text</code>	Text for page numbering string. The [i] is optional - it allows for more than one entry. For example, <code>-pagestr "string 1" -pagestr2 "string 2"</code> and so on. Use %p for the current page. For example, "Page: %p". Use two %'s in a row if you're running the program from within a batch file (i.e. "Page: %%p"). You may also use %t for total pages. Using %t will require more processing time since the PDF must be built twice - once to determine the number of pages then once again to use the number in the string for the final output.
<code>-pagetxtfont face,number</code>	Sets the font and point size for the page text. Pass the font name (must be one of the standard Courier, Helvetica or Times fonts) along with the point size. For example, <code>-pagetxtfont times,9</code> will use Times Roman at a point size of 9. See the <a href="#">FONT</a> tag for a list of the fonts.  You may also pass in a tag based setting that matches the format of the <FONT> tag. This allows you to set your own TrueType or OpenType font if you wish. In this case, pass only one parameter. For example, <code>-pagetxtfont "&lt;FONT SRC='myfont.ttf' SIZE=20&gt;"</code> .
<code>-pagew number</code>	The width of the page in inches or units. Default is 8.5 inches.
<code>-pbm message</code>	Sets the message of the progress box.
<code>-pbt title</code>	Sets the title of the progress box.

## Executable

<code>-pcl</code>	<p>Removes PCL commands from the text file. Currently, the only PCL commands that are processed are horizontal and vertical position, point size, bolding and page break. The horizontal and vertical positioning are done by replacing the PCL commands with Text2PDF X and Y tags. For example, Esc&amp;a1440v2880H is replaced with &lt;Y VALUE=2&gt;&lt;X VALUE=4&gt;. The decipoints of 1440 and 2880 are converted to inches (1440 / 720 and 2880 / 720) to get the 2 and 4. The current left margin and top margin are also added in so you can use them to offset the actual text placement.</p>
<code>-pclskip <i>text</i></code>	<p>Skips the specified PCL commands. For example, use <code>-pclskip "FONT,B"</code> to bypass setting the font size and bolding specified from the PCL instructions. Use along with the <code>-pcl</code> option. Valid entries to pass to <code>-pclskip</code> are:</p> <ul style="list-style-type: none"><li>FONT</li><li>LINESPACE</li><li>PAGE</li><li>B</li><li>I</li><li>X</li><li>Y</li></ul>

## Executable

<code>-pdf <i>pdf-file</i></code>	<p>Background PDF to use with the output PDF. Use quotes around the name if you have spaces in the path or file name. Remember to supply the owner password, if this PDF is encrypted, using the <code>-bkpass</code> parameter.</p> <p>Note that the background PDF may contain page rotations or similar viewer settings that will be discarded when used as a background. For example, the PDF may have its contents displayed upside down if the original PDF contained a page rotation of 180 degrees. You'll need to modify the PDF either in the application that created it or by using an application such as FyTek's PDF Meld (visit <a href="http://www.fytek.com">http://www.fytek.com</a> for information on PDF Meld).</p>
<code>-pdfbm <i>text</i></code>	<p>A single bookmark to use for the PDF. Pass in the text for the bookmark. The bookmark will point to page 1. Use this option when creating a set of PDFs that will later be merged into one where you want a bookmark entry for each PDF.</p>
<code>-pdfpage <i>number</i></code>	<p>Page number from background PDF to use. This is optional and can be set when needed by the PDFPAGE option on the <a href="#">PAGE</a> tag.</p>
<code>-point <i>number</i></code>	<p>The point size of the font for the output PDF.</p>
<code>-pointpct <i>number</i></code>	<p>A percentage to scale the point size by. For example, <code>-pointpct 50</code> will reduce the point size by 50% throughout the document.</p>

## Executable

<code>-pre [plain/html/white/whitenb]</code>	Specifies the text is preformatted and existing line breaks (ASCII 10) and form feeds (ASCII 12) should be used. The program converts these into the   and <PAGE> tags respectively. Use the "plain" option for faster processing of text without tags when you don't want the program to line break at the right margin. Use the "html" option for text with tags when you still don't want the program to line break at the right margin. Use the "white" option for text with tags when you want the program to line break at the right margin and you want to preserve white space within the text. Use the "whitenb" option to do what "white" does but also not attempt to line break (for use with preformatted text). The plain and html options work best for legacy or mainframe type reports which usually print on greenbar paper. Be sure to select a courier or fixed spaced font in this case.
<code>-print</code>	Automatically prints the newly created PDF to the default printer. Must have Acrobat or Reader installed.
<code>-printcopies n</code>	Works with Acrobat or Reader 8.0 or higher. The number of copies to be printed when the print dialog is opened for this file. Supported values are the integers 2 through 5. Values outside this range are ignored.
<code>-printdlg</code>	Brings up the Acrobat print dialog box and allows printer selection. This only works when the user has Acrobat or Reader associated with PDFs on their machine. Otherwise the user's viewer is opened with the document and they will need to print from there.

## Executable

- `-printduplex "text"` Works with Acrobat or Reader 8.0 or higher. The paper handling option to use when printing the file from the print dialog. The following values are valid:  
Simplex Print single-sided  
DuplexFlipShortEdge Duplex and flip on the short edge of the sheet  
DuplexFlipLongEdge Duplex and flip on the long edge of the sheet
- `-printer printer device port` 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 tings 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 file` 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.

## Executable

-printpagerange "from,to,..."	Works with Acrobat or Reader 8.0 or higher. The page numbers used to initialize the print dialog box when the file is printed. The first page of the PDF file is denoted by 1. Each pair consists of the first and last pages in the sub-range. An odd number of integers causes this entry to be ignored. Negative numbers cause the entire array to be ignored.
-printpicktray	Works with Acrobat or Reader 8.0 or higher. Specifies the PDF page size is used to select the input paper tray. This setting influences only the preset values used to populate the print dialog presented by a PDF viewer application. If used, the check box in the print dialog associated with input paper tray is checked.
-printscale "text"	Works with Acrobat or Reader 7.0 or higher. Valid values are None, which indicates that the print dialog should reflect no page scaling, and AppDefault, which indicates that applications should use the current print scaling.
-printscr <i>file</i>	For Unix/Linux systems. 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.
-producer <i>text</i>	Sets the document producer.
-removectl	Removes any lines from the input file that begin with the ASCII character 27 (escape character).
-rend <i>number</i>	0 = Fill text (default) 1 = Stroke text (outline) 2 = Fill then stroke 3 = No fill or stroke (invisible)
-right <i>number</i>	The distance to move the contents of each page to the right. The number is in units of 1/72 of an inch or the unit setting. May be positive or negative value.
-rm <i>number</i>	Right margin in inches or units. Default is .5 inches.



## Executable

-rtf	Specifies the input file is in RTF format. Only a small subset of the RTF format is supported so not every RTF will format correctly.
-rtfbm <i>left/center/right</i>	Specifies the alignment to look for in the RTF to determine the text to use for bookmarks. For example, set to "center" to look for center aligned text on the page.
-rtffont "name,src[,bold,italic,bold-italic]"	Call this method for each font you wish to add. The "name" parameter must match the name used in the RTF document. Set "src" to the path and file name of the TrueType font you want to use. Optionally include the path and file to the bold, italic, and bold-italic versions of the font.
-rtfsize <i>number</i>	Specifies a scaling factor (100 is the default) to use against the font sizes found in the RTF file.
-s	Include subdirectories when batch converting. Specify an input and output directory rather than input and output file. For example, <code>text2pdf.exe c:\myfiles c:\output -s -mask *.txt</code> will convert all files with a .txt extension in c:\myfiles and all of its subdirectories.
-scale <i>number</i>	The scaling factor to apply to each page. The default is 100 or no scaling. A value of 50 will scale the contents to 50% of their original size.
-scolor <i>color</i>	Specifies the stroke color (only for special render modes). See the <a href="#">color</a> section for valid values.
-server	Starts up Text2PDF in server mode. Text2PDF will run as a separate process and take commands from the TCP/IP port it is connected to. The <a href="#">Client-Server</a> section describes this operation in more detail.
-signbgcolor <i>text</i>	Optional. The background color for the signature field. See the <a href="#">Digital Signature</a> section for details.

## Executable

-signderfile <i>text</i>	The path and name of the der-encoded signing certificate. For example, "c:\keys\mykey.der". See the <a href="#">Digital Signature</a> section for details.
-signhide	Optional. Keeps the signature field from showing on the first page of the output PDF. See the <a href="#">Digital Signature</a> section for details.
-signimg <i>text</i>	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 <a href="#">Digital Signature</a> section for details.
-signkeepratio	Optional. Keep the image x/y scaling ratio when using an image with a signature field. See the <a href="#">Digital Signature</a> section for details.
-signname <i>text</i>	Optional. The name of the signature field to use in the PDF. For example, "sig1". See the <a href="#">Digital Signature</a> section for details.
-signpkfile <i>text</i>	The path and name of the private key file. For example, "c:\keys\mykey_pk.pem". See the <a href="#">Digital Signature</a> section for details.
-signpwd <i>text</i>	Optional. The password for the signing certificate private key. See the <a href="#">Digital Signature</a> section for details.
-signrsn <i>text</i>	Optional. The reason for signing the document. Default is "Attestation to the accuracy and integrity of this document". See the <a href="#">Digital Signature</a> section for details.
-signsize <i>text</i>	Optional. The fontsize for the text of the signature (0 for no text). Default is 12. See the <a href="#">Digital Signature</a> section for details.
-signssl <i>text</i>	The path and file name of the OpenSSL program. For example, "c:\openssl\bin\openssl.exe". See the <a href="#">Digital Signature</a> section for details.

## Executable

-subject <i>text</i>	Sets the document subject.
-tababsspace <i>number</i>	Absolute number of spaces to use for tabs found in the input file. Any tab is simply replaced by this number of spaces.
-tabspace <i>number</i>	Maximum number of spaces to use for tabs found in the input file. The number of spaces depends on the position in the line where the tab is found. The modulo of the position and the value of this parameter is used to determine the number of spaces. Use with pre-formatted text where a tab stop is used to create columns.
-title <i>text</i>	Sets the document title.
-tm <i>number</i>	Top margin in inches or units. Default is .5 inches.
-u <i>password</i>	Sets the user password for the PDF. No password is prompted for when opening the PDF if only an owner password was specified. This will allow you to restrict users from printing, for example, without requiring a password to open the document.
-units <i>in/cm/mm/pt</i>	The units you wish to use when specifying options like page height and width. Be sure to place this option before other options on the command line. Use one of the following: in = inches (default) cm = centimeters mm = millimeters pt = point (1/72 of an inch)

## Executable

- `-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 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.

## Executable

-xpspage *number*

Page number from background XPS to use. This is optional and can be set when needed by the XPSPAGE option on the [PAGE](#) tag.

-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.

*DLL (Dynamic Link Library)*

## Using the DLL (Dynamic Link Library)

The DLL is compiled as a 32 and 64-bit .NET DLL. You may register it with regasm to make it available from a program that uses a COM DLL such as VBScript or PHP. The file text2pdf\_20.dll is the compiled DLL. The source code is available on [GitHub](#) if you want to make changes.

The .NET DLL is a wrapper for the executable text2pdf.exe or text2pdf64.exe (which is the default). It allows you to easily add the functionality of Text2PDF to any existing code that can access a .NET or COM DLL. If you start a Text2PDF server then the DLL may be used to call the running server instead.

The DLL also has methods to start and stop a Text2PDF server though you may also do so from the command line (see the -server option). This provides the added benefit of keeping the program in memory for quick access without the cost of program startup and shutdown each time you build a PDF. In addition you can control the number of simultaneous builds to minimize memory or CPU usage. If your license allows, you may run multiple servers on the same or different boxes and the DLL will cycle requests between the running servers. Each server instance for Text2PDF requires a separate license when purchasing individually.

Use the startServer method to bring up a Text2PDF server. Text2PDF then waits in memory listening for commands on a port (default is 7080) that you specify when starting the server. The DLL sends commands to Text2PDF on that port in order to build the PDF. You may have Text2PDF save the PDF to disk or send it back to the DLL as byte array.

DLL (Dynamic Link Library)

`addText(text)`

Commands to execute (when not using an input file). Call this method for each command you wish to execute. You could store your commands in an array then loop through it calling this method for each element. You may also string a bunch of commands together separated by a carriage-return and line-feed. Leave the input file blank and send commands to `addText` if you are creating them on the fly and just want to pass them to the program.

`buildPDF bool waitForExit = true  
 ,String saveFile = ""`

Call this method to build the PDF after setting all of your other options. The server will be used if `setServer` or `setServerFile` was used. Set the first parameter to true when you want to wait for Text2PDF to finish building the PDF before returning from this method when using the executable (non-server) version. The `waitForExit` is used to mean wait and return the byte array of the PDF when using the server version. Optionally pass a `saveFile` which will be used to save the file locally when running as a server. You may also set `saveFile` when not using a sever to bypass the `setOutFile` method call. Returns an object (when `waitForExit` is true). The object has these properties:  
`bytes[]` Bytes  
`int` Pages (only when using a Text2PDF server)  
`String` Msg (only when using a Text2PDF server)

The Bytes property is the raw bytes of the output PDF. Use this to store in a database or stream to a browser. The Msg is a message with an error or "OK" if PDF was generated (though may still have issues if bad information was passed) without error. Pages is the number of pages in the output PDF.

## DLL (Dynamic Link Library)

<code>licInfo String licName     ,String licPwd     ,int autoDownload</code>	Specify your license name, password and optionally if you want to download the license configuration automatically. You only need to specify this on the server startup if you are using the <code>startServer</code> method in which case users of the server do not need to include this information.
<code>sendFileTCP String fileName,     ,String filePath = ""     [,byte[] contents = null]</code>	The name and location of a file to send when using Text2PDF as a service. Use this when the server is on a different box or you don't have knowledge of what directory the service is running under. For example, you might use <code>setInFile("somefile.txt")</code> where you are using "somefile.txt" as a placeholder. You then call <code>sendFileTCP "somefile.txt", "c:\temp\myrealfile.txt"</code> . This sends <code>c:\temp\myrealfile.txt</code> to the server and instructs it to treat it as a file named <code>somefile.txt</code> . You may leave the path to the file empty and instead send a byte array of the file instead. This allows you to pass the file contents from memory rather than disk.
<code>serverStatus(bool allServers = false)</code>	Provides a status on the server (or all servers). Use <code>setServer</code> first to assign the host and port if they differ from the default.
<code>setAlign (char L R C J)</code>	The alignment to use for the text. The options are: L = Left (default) R = Right C = Center J = Justify
<code>setAuthor(String text)</code>	Sets the document author.
<code>setAutoFit</code>	Fits the page height to the contents of the page. The <code>PageSize</code> height, in this case, will be the maximum page size.



## DLL (Dynamic Link Library)

setAutosize([int lines])	Used to automatically set the font point size and text width compression. Use this option along with setPre('Plain') or setColOne methods to check the input and reduce the font size if necessary. The "lines" value is optional. If used, set to the number of lines to read from the input to determine the optimal font size. For example, 100 to read the first 100 lines from the input and assume the rest of the file is set up in the same manner. Set to at least one page worth of data or leave off to scan the entire input. The font is never increased in size, only reduced so you should set the point size to the maximum point size you wish to use.
setBarcodes	Use this method when you are using one of the built-in barcode fonts.
setBkgImg String file ,Double x ,Double y [,Double scalex Double scaley]	Background image or watermark to use with the output PDF. Supply the file name (with full path), x position from the left edge of the page in points, y position from bottom edge of page in points. For example, "myfile.jpg", 72, 144 will place the lower left corner of image "myfile.jpg" 1 inch (72 points) from the left edge and 2 inches from the bottom. Optionally pass in a scaling factor for the x/y axis. The default for scalex and scaley is 100.
setBkgPage(int number)	Page number from background PDF to use. This is optional and can be set when needed by the PDFPAGE option on the <a href="#">PAGE</a> tag.
setBkgPassword(String text)	The owner password for the background PDF. This is only needed when the background PDF is encrypted.

DLL (Dynamic Link Library)

setBkgPDF(String text)	<p>Background PDF to use with the output PDF. Remember to supply the owner password, if this PDF is encrypted, using the setBkgPassword.</p> <p>Note that the background PDF may contain page rotations or similar viewer settings that will be discarded when used as a background. For example, the PDF may have its contents displayed upside down if the original PDF contained a page rotation of 180 degrees. You'll need to modify the PDF either in the application that created it or by using an application such as FyTek's PDF Meld (visit <a href="http://www.fytek.com">http://www.fytek.com</a> for information on PDF Meld).</p>
setColOne	<p>Used to specify the first column contains a vertical print instruction. The input is treated as preformatted text so no line breaks are inserted by the software. Be sure to select a courier or fixed spaced font in this case. The first character is checked to see how to print the line based on the following:</p> <ul style="list-style-type: none"><li>0 = Advance 2 lines then print</li><li>- = Advance 3 lines then print</li><li>1 = Start a new page</li><li>+ = Don't advance the line before printing</li><li>blank = Advance a line and print</li></ul>
setColumns(int number)	<p>The number of text columns per page. Default is 1.</p>
setColumnSpace(int number)	<p>The amount of spacing in inches or units when using two or more columns.</p>
setComp15	<p>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.</p>

## DLL (Dynamic Link Library)

setEncrypt128	Sets 128-bit encryption method. Files encrypted with 128-bit encryption can only be opened with Acrobat or Adobe Reader 5.0 or above. The default encryption is 40-bit which works with Acrobat and Adobe Reader 4.0 and above.
setEncryptAES(String 128 256)	Sets AES encryption method. Pass 128 for 128-bit encryption or 256 for 256-bit encryption. Files encrypted with AES 128-bit encryption can only be opened with Acrobat or Adobe Reader 7.0 or above. Files encrypted with AES 256-bit encryption can only be opened with Acrobat or Adobe Reader 9.0 or above.
setExe(String fileName)	Supply the path and filename of the Text2PDF executable, either text2pdf.exe or text2pdf64.exe. This is only needed when calling the program directly to build a PDF rather than using a server. Also needed if you are using the DLL to start a Text2PDF server.
setFileMask(String text)	File names to match when batch converting. Specify an input and output directory rather than input and output file with setInFile and setOutFile methods. Use wildcards to match on file names. For example, *.txt will match all files with an extension of .txt. A mask of t*.dat will match all files starting with the letter t and having an extension of .dat.
setFillColor(String color)	Specifies the fill color which is the font color. See the <a href="#">color</a> section for valid values.
setForce	Turns off the dialog box prompting to overwrite the output file if it exists. If a PDF by the same name as the output PDF already exists it is overwritten.

## DLL (Dynamic Link Library)

setForwardRef	Used to specify the document contains forward references. Use this option when you have links in the document that reference anchors further in the document. For example, if you have <A HREF="#mylink"> on a page but don't have <A NAME="mylink"> until after that link, you'll want to use this option. This is so the program knows to scan for all the links first and then create the PDF. If all of your links are at the end of the document (like with an index page) then you don't need this option.
SetGUIOff	Suppresses the dialog window that shows the current build progress.
setInFile(String path-file)	Full path and name of the input file. You set the input file only if you want to read the commands from an existing file (as opposed to using the addText method).
setIniFile(String filename)	An initialization file containing text replacement. Use this when you have a file with various codes for things such as underlines or colors. This file can be used to specify the conversion between the text and tag. See the <a href="#">Initialization File</a> section for details.
setKeyCode(String keyCode)	Specify your key code to use when running a licensed version. You only need to specify this on the server startup if you are using the startServer method in which case users of the server do not need to include this value.
setKeyName(String keyName)	Specify your key name to use when running a licensed version. You only need to specify this on the server startup if you are using the startServer method in which case users of the server do not need to include this value.
setKeywords(String text)	Sets the document keywords.
setLineSpace(Double number)	The amount of space (always in points - 1/72 of an inch) to leave between lines. Default is 2.

## DLL (Dynamic Link Library)

setMail	Opens the user's email program to a composition window with the newly created PDF attached. May not work with all email programs.
setMargins Double topMargin ,Double rightMargin ,Double bottomMargin ,Double leftMargin	Top, Right, Bottom and Left margin in inches or units.
setNoAnnote	Disables add/change of form fields or annotations.
setNoAssemble	<i>(128-bit only)</i> Disables assembly (insert, rotate, delete pages or create bookmarks) when setNoChange is used.
setNoChange	Disables changes to the document.
setNoCopy	<i>40-bit :</i> Disables copying of text and/or graphics from the document. <i>128-bit :</i> Disables copying of text and/or graphics from the document other than in support of accessibility to disabled users or for other purposes.
setNoDigital	<i>(128-bit only)</i> Disables printing at digital quality - can only print low resolution. The NoPrint method overrides this option so you'll want to use setNoPrint or setNoDigital but not both.
setNoExtract	<i>(128-bit only)</i> Disables extraction of information in support of accessibility to disabled users or for other purposes.
setNoFillIn	<i>(128-bit only)</i> Disables fill in interactive fields when setNoAnnote is used.
setNoPrint	Disables printing of the document (even low resolution).

*DLL (Dynamic Link Library)*

setNoRights	Turns off all rights (default is all are granted). Calling of methods such as NoPrint or setNoCopy turns those rights on rather than off. Use this if you typically are turning off most or all of the rights. Note that setting setNoRights and setNoPrint will allow high resolution printing. Setting setNoRights and NoDigital will allow low resolution printing. Setting only setNoRights will disallow printing.
setNoSoftBreak	Only page break on a hard page break (like a PAGE command). Text may flow past the bottom margin if there is more than would normally fit on the page with this option.
setNumCopies(int number)	Number of copies to print when using the setPrint or setPrinter methods. Default is 1.
setOpen	Automatically opens Acrobat and loads the newly created PDF.
setOptimize(bool compress)	Optimize the output PDF for fast web viewing. Optionally pass a true value to further compress the PDF. Note this typically increases the size of the output by a few hundred bytes or so. The PDF is optimized for viewing on the web as opposed to shrinking the physical size. As a rough guide, you should only optimize a PDF that will be viewed on the web when it contains more than 10 pages and its size is bigger than a megabyte. Additionally, you must create the PDF to a file rather than stream the output to the browser. The setting "fast web view" will be set to yes for optimized PDFs when you open in Reader and check the properties. This means the first page of the PDF is sent to the user and made viewable while the rest of the pages continue to download in the background.

*DLL (Dynamic Link Library)*

setOutFile(String path-file)	<p>Full path and name of the output file. You can leave the output blank and have the PDF stream returned to a variable in your program. The returned string is binary and may come back as a two-byte string depending on how the calling program treats the string.</p> <p>Also, you may use the special keyword "membuild" like this: setOutFile("membuild"). Using "membuild" returns the output PDF as a byte array to the buildPDF method.</p>
setOwner(String password)	<p>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.</p>
setPageBorder Double width [,String color [,Double padding]]	<p>Used to draw a border on the page. The color and padding are optional. The width and padding are specified in points (1/72 of an inch). Any valid <a href="#">color</a> may be used for the border.</p>
setPageDown(Double number)	<p>The distance to move the contents of each page down. The number is in units of 1/72 of an inch or the unit setting. May be positive or negative value.</p>

DLL (Dynamic Link Library)

setPageNum String text ,String font ,Double pointsize ,Double x ,Double y [,int i]	<p>Text for page numbering string. Use %p for the current page. For example, "Page: %p". You may also use %t for total pages. Using %t will require more processing time since the PDF must be built twice - once to determine the number of pages then once again to use the number in the string for the final output.</p> <p>Pass the font name (must be one of the standard Courier, Helvetica or Times fonts) along with the point size. For example, setPageNum "times",9 will use Times Roman at a point size of 9. See the <a href="#">FONT</a> tag for a list of the fonts.</p> <p>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.</p> <p>You may also pass in a tag based setting that matches the format of the &lt;FONT&gt; tag. This allows you to set your own TrueType or OpenType font if you wish. In this case, pass 0 for pointsize. The i value is optional - use it to set a different layout starting on that page where i is the page number.</p>
setPageRight(String number)	<p>The distance to move the contents of each page to the right. The number is in units of 1/72 of an inch or the unit setting. May be positive or negative value.</p>
setPageScale(Double number)	<p>The scaling factor to apply to each page. The default is 100 or no scaling. A value of 50 will scale the contents to 50% of their original size.</p>
setPageSize Double number ,Double number	<p>The width and height of the page in inches or units. Default is 8.5 by 11 inches.</p>



DLL (Dynamic Link Library)

setPageTxtFont String font ,Double pointsize	Sets the font and point size for the page text. Pass the font name (must be one of the standard Courier, Helvetica or Times fonts) along with the point size. For example, "times",9 will use Times Roman at a point size of 9. See the <a href="#">FONT</a> tag for a list of the fonts.  You may also pass in a tag based setting that matches the format of the <FONT> tag. This allows you to set your own TrueType or OpenType font if you wish. In this case, pass only one parameter. For example, "<FONT SRC='myfont.ttf' SIZE=20>".
setPointPct(Double number)	A percentage to scale the point size by. For example, 50 will reduce the point size by 50% throughout the document.
setPre([String text])	Specifies the text is preformatted and existing line breaks (ASCII 10) and form feeds (ASCII 12) should be used. The program converts these into the   and <PAGE> tags respectively. Use the "Plain" option for faster processing of text without tags when you don't want the program to line break at the right margin. Use the "HTML" option for text with tags when you still don't want the program to line break at the right margin. Use the "white" option for text with tags when you want the program to line break at the right margin and you want to preserve white space within the text. Use the "whitenb" option to do what "white" does but also not attempt to line break (for use with preformatted text). The Plain and HTML options work best for legacy or mainframe type reports which usually print on greenbar paper. Be sure to select a courier or fixed spaced font in this case.
setPrint	Automatically prints the newly created PDF to the default printer. Must have Acrobat or Reader installed.

## DLL (Dynamic Link Library)

<code>setPrinter String printer [,String device ,String port]</code>	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 off device and port to use the default settings for the printer. For example: <code>setPrinter "Accounting Printer", "HP LaserJet 5", "lpt1:"</code> or <code>setPrinter "Shipping Printer"</code> 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: <code>setPrinter "\\server\printer"</code>
<code>setProducer(String text)</code>	Sets the document producer.
<code>setRemoveCtl</code>	Removes any lines from the input file that begin with the ASCII character 27 (escape character).
<code>setRender(int number)</code>	0 = Fill text (default) 1 = Stroke text (outline) 2 = Fill then stroke 3 = No fill or stroke (invisible)
<code>setServer String host = "localhost" int port = 7080</code>	Assigns the host and port for the server to use for the duration of the connection to the DLL object. The defaults are shown above. For example: <code>setServer "192.168.0.5", 7080</code>

## DLL (Dynamic Link Library)

setServerFile(String fileName)	<p>A list of IP addresses and port numbers where Text2PDF servers are running. The DLL will cycle between these for requests. The list of servers are stored in a static list in the DLL meaning once set all users that access the DLL will use the same settings without having to include this method call. Lines that begin with # are ignored. Use exe for the IP port to specify where the executable is located when not running a server. For example:</p> <pre>localhost 7080 192.168.0.5 7080 192.168.0.8 7085</pre> <p>Assigns three servers (note you must have a license for each server that is running Text2PDF). To instead specify where the executable is located when not using a server enter a line like this:</p> <pre>exe c:\mypath\text2pdf64.exe</pre>
setStrokeColor(String color)	<p>Specifies the stroke color (only for special render modes). See the <a href="#">color</a> section for valid values.</p>
setSubDir	<p>Include subdirectories when batch converting. Specify an input and output directory rather than input and output file.</p>
setSubject(String text)	<p>Sets the document subject.</p>
setTabAbsSpace(int number)	<p>Absolute number of spaces to use for tabs found in the input file. Any tab is simply replaced by this number of spaces.</p>
setTabSpace(int number)	<p>Maximum number of spaces to use for tabs found in the input file. The number of spaces depends on the position in the line where the tab is found. The modulo of the position and the value of this parameter is used to determine the number of spaces. Use with pre-formatted text where a tab stop is used to create columns.</p>
setTextCompress(Double number)	<p>The compression factor to apply to the text width. Default is 100. Smaller numbers make the text thinner while larger ones make it wider.</p>

## DLL (Dynamic Link Library)

<code>setTitle(String text)</code>	Sets the document title.
<code>setUnits(String in cm mm pt)</code>	The units you wish to use when specifying values for methods like page height and width. Be sure to call this method before other methods requiring a value based on units. Use one of the following: in = inches (default) cm = centimeters mm = millimeters pt = point (1/72 of an inch)
<code>setUser(String password)</code>	Sets the user password for the PDF. No password is prompted for when opening the PDF if only an owner password was specified. This will allow you to restrict users from printing, for example, without requiring a password to open the document.
<code>setUTF8 String language [,String convert]</code>	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 to UTF-8 you would pass <code>setUTF8("zh-cn,Y")</code> .
<code>setWorkingDir(String text)</code>	Changes the working directory to the specified directory.

*DLL (Dynamic Link Library)*

<pre>startServer String host = "localhost"     ,int port = 7080     ,int pool = 5     ,String log = ""</pre>	<p>Starts a Text2PDF server on the box you are issuing this command from. The default values are shown above. The default executable is text2pdf64.exe. If you want to use text2pdf.exe (32-bit) or need to specify the path where text2pdf64.exe is located then use setExe(path-file). Additionally, you'll need to include a key-name/key-code combination or have a software subscription. Use setKeyName/setKeyCode or licInfo methods to specify. You may also use setKeyName("demo") if you are using the demo version of the software. The "log" option is the path and file name of the file for the server log output. If you running a server for access by other computers then do not use localhost. Instead, use the name of the box or its IP address.</p>
<pre>stopServer</pre>	<p>Stops the Text2PDF server. Use setServer first to assign the host and port if they differ from the default.</p>

### *DLL (Dynamic Link Library)*

Here is an example of calling the DLL using VBScript.

```
Dim t2pobj
set t2pobj = CreateObject("FyTek.Text2PDF")
t2pobj.setOutFile "c:\temp\hello.pdf"
t2pobj.addText "times", 10
t2pobj.addText ("Text line 1<BR>")
t2pobj.addText ("Text line 2<BR>")
t2pobj.addText ("<PAGE WIDTH=11 HEIGHT=8.5 COLS=2 COLSPACE=1>")
t2pobj.addText ("More text...<BR>")
t2pobj.buildPDF()
```

Visit [GitHub](#) for more sample code.

## Runtime Version

FyTek sells licenses for Text2PDF which allow you to distribute the registered exe or DLL to an end users. The runtime versions require a combination of a key code and key name be passed to the software in order for it to work.

### Exe version

The following command line options are used with the runtime version:

-kn keyname  
-kc keycode

These values will be provided to you by FyTek, Inc.

For example, to create a report you would run something similar to the following:

```
text2pdf.exe sample.txt sample.pdf -kn mycompany -kc ABC123ABC123ABC123
```

Note that neither the key name or key code should be made visible to the end user (via a .bat file for instance).

### DLL version

The following methods are used with the runtime version:

setKeyName (keyname)  
setKeyCode (keycode)

These values will be provided to you by FyTek, Inc.

For example, to create a report you would run something similar to the following:

```
Dim outPdf As String  
Set pdfCr = CreateObject("fytek.Text2PDF")  
pdfCr.OutputFile "c:\temp\hello.pdf"  
pdfCr.SetKeyName ("mycompany")  
pdfCr.SetKeyCode ("ABC123ABC123ABC123")  
pdfCr.PageTxtFont "times", 10  
pdfCr.PDFCmd ("Text...")  
pdfCr.PDFCmd ("More text...")  
pdfCr.buildPDF
```

### *Runtime*

Note that neither the key name or key code should be made visible to the end user.

The DLL for the runtime version is text2pdf.dll. While it is named different from the developer file (text2pdf.dll) it still has the same function names internally. For this reason, if you have both installed on the same machine (for development and testing) you'll need to run regsvr32 on the one you want to work with. For instance, to work with the development version run:

```
regsvr32 text2pdf.dll
```

The file should reside in your windows or winnt system32 sub-directory. Passing the extra parameters for key name and key code will simply be ignored by the developer version. To work with the runtime version, run:

```
regsvr32 text2pdf.dll
```

This will register the runtime DLL with your system and your application will then reference that program.



## Commands

All commands must be enclosed in angle brackets. Commands may be entered in upper or lowercase.

Any tags not recognized are ignored.

Tags may not span lines. You may place as many tags as you wish on a single line but the closing ">" for any tag must appear in the same line as the opening "<".

Use single or double quotes when entering a text value with spaces as a parameter. For example, <TAG DESCR="My Test Description">.

Do not leave a space between a parameter name, the = sign and its value.

Correct <PAGE WIDTH=8.5 HEIGHT=11>

Incorrect <PAGE WIDTH = 8.5 HEIGHT= 11>

Note that either the " or ' character can be used to enclose a string. You must use the corresponding character to close the string that you used to open it with however.

You may also use quotes around numeric values and use a /> to close a tag. This is for compatibility with an XML syntax approach. Any of the following are acceptable tags for Text2PDF and all work the same:

<page height=11 width=8.5>

<page height="11" width="8.5"/>

<page height="11" width="8.5" />

## Document Level Commands

**<HTML  
LANG=text>**

Optional. Sets the base language for Japanese, Chinese, or Korean when providing UTF-8 encoded text.

<u>Parameter</u>	<u>Description</u>
LANG	Enter 'ja' for Japanese, 'zh' for Chinese, or 'ko' for Korean. UTF-8 characters will be converted based on the language entry. When used with the <a href="#">META</a> tag, it is not necessary to specify text with one of the Japanese, Chinese, or Korean font faces. That is, you do not need to use a font face such as J1, C1, or K1. This tag is optional though it is recommended when using UTF-8 with one of these languages.

## Document Level Commands

**<META  
CONTENT=text>**

Sets the document content type.

<u>Parameter</u>	<u>Description</u>
CONTENT=text	Example: CONTENT="charset=utf-8". The phrase "charset=utf-8" must appear somewhere in the string to specify the document contains text that is UTF-8 encoded. This will instruct the program to allow for additional character sets for Chinese, Japanese, and Korean. You may also specify -utf8 on the command line (or call the method UTF8).

## Document Level Commands

```
<CREATOR  
  VALUE=text  
  LANGUAGE=text  
  COUNTRY=text>
```

Sets the text for creator of the document.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Text to use for document creator.
LANGUAGE=text	Specify a 2-character ISO 639 language code - for example, EN for English or JA for Japanese. Text is assumed to be Unicode (2-character format) when this is used. The complete list of codes are available through <a href="http://www.iso.ch">http://www.iso.ch</a> .
COUNTRY=text	Optional, used with the LANGUAGE option above. A 2-character ISO 3166 country code - for example, US for the United States or JP for Japan.

## Document Level Commands

```
<SUBJECT  
  VALUE=text  
  LANGUAGE=text  
  COUNTRY=text>
```

Sets the text for subject of the document.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Text to use for document subject.
LANGUAGE=text	Specify a 2-character ISO 639 language code - for example, EN for English or JA for Japanese. Text is assumed to be Unicode (2-character format) when this is used. The complete list of codes are available through <a href="http://www.iso.ch">http://www.iso.ch</a> .
COUNTRY=text	Optional, used with the LANGUAGE option above. A 2-character ISO 3166 country code - for example, US for the United States or JP for Japan.

## Document Level Commands

```
<AUTHOR  
  VALUE=text  
  LANGUAGE=text  
  COUNTRY=text>
```

Sets the text for author of the document.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Text to use for document author.
LANGUAGE=text	Specify a 2-character ISO 639 language code - for example, EN for English or JA for Japanese. Text is assumed to be Unicode (2-character format) when this is used. The complete list of codes are available through <a href="http://www.iso.ch">http://www.iso.ch</a> .
COUNTRY=text	Optional, used with the LANGUAGE option above. A 2-character ISO 3166 country code - for example, US for the United States or JP for Japan.

## Document Level Commands

```
<TITLE  
  VALUE=text  
  LANGUAGE=text  
  COUNTRY=text>
```

Sets the text for title of the document.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Text to use for document title.
LANGUAGE=text	Specify a 2-character ISO 639 language code - for example, EN for English or JA for Japanese. Text is assumed to be Unicode (2-character format) when this is used. The complete list of codes are available through <a href="http://www.iso.ch">http://www.iso.ch</a> .
COUNTRY=text	Optional, used with the LANGUAGE option above. A 2-character ISO 3166 country code - for example, US for the United States or JP for Japan.

## Document Level Commands

```
<KEYWORDS  
  VALUE=text  
  LANGUAGE=text  
  COUNTRY=text>
```

Sets the text for the document keywords.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Text to use for document keywords.
LANGUAGE=text	Specify a 2-character ISO 639 language code - for example, EN for English or JA for Japanese. Text is assumed to be Unicode (2-character format) when this is used. The complete list of codes are available through <a href="http://www.iso.ch">http://www.iso.ch</a> .
COUNTRY=text	Optional, used with the LANGUAGE option above. A 2-character ISO 3166 country code - for example, US for the United States or JP for Japan.



## Document Level Commands

```
<BOOKMARK  
  LEVEL=number  
  DESCR=text  
  LANGUAGE=text  
  COUNTRY=text  
  CLOSED>
```

Used to set up bookmarks for the document.

<u>Parameter</u>	<u>Description</u>
LEVEL=number	The level of the bookmark. 1 is the top level, 2 would be a sub-level to 1, etc.
DESCR=text	The description that appears in the bookmarks pane.
LANGUAGE=text	Specify a 2-character ISO 639 language code - for example, EN for English or JA for Japanese. Text is assumed to be Unicode (2-character format) when this is used. The complete list of codes are available through <a href="http://www.iso.ch">http://www.iso.ch</a> .
COUNTRY=text	Optional, used with the LANGUAGE option above. A 2-character ISO 3166 country code - for example, US for the United States or JP for Japan.
CLOSED	Adding this option will cause the initial display of the bookmark to be closed.

## Document Level Commands

### <ZOOM

VALUE=number|FITPAGE|FITWIDTH>

Used to set the initial zoom factor. Default is dependant on user tings.

<u>Parameter</u>	<u>Description</u>
VALUE	The zoom factor to open the document at. Enter 100 for 100 percent.
	FITPAGE = open the document sized so the entire page fits in the window.
	FITWIDTH = open the document sized so the width of the page fits in the window.

**<PAGE**  
**HEIGHT=inches**  
**WIDTH=inches**  
**FIT**  
**LM=number**  
**RM=number**  
**TM=number**  
**BM=number**  
**BORDER=number**  
**BORDERCOLOR=color**  
**BORDERPADDING=number**  
**COLS=number**  
**COLSPACE=number**  
**Y=number**  
**SHADING=text**  
**PDFPAGE=number**  
**XPSPAGE=number**  
**CLEARBACKGROUND>**

Starts a new page. All of the parameters are optional. If you are doing a page break and you are changing the width or height, you'll probably need to use CLEARBACKGROUND if the background was based on a different page size.

<u>Parameter</u>	<u>Description</u>
WIDTH=number	Sets width of page in inches or units. Default is 8.5 inches.
HEIGHT=number	Sets height of page in inches or units. Default is 11 inches.
FIT	Fits the page height to the contents of the page. The HEIGHT option, in this case, will be the maximum page size.
LM=number	Sets the left margin in inches or units. Default is .5 inches.
RM=number	Sets the right margin in inches or units. Default is .5 inches.
TM=number	Sets the top margin in inches or units. Default is .5 inches.
BM=number	Sets the bottom margin in inches or units. Default is .5 inches.
BORDER=number	The width of the border to draw on the page. This value is specified in points (1/72 of an inch).
BORDERCOLOR=color	Enter a valid <a href="#">color</a> for the border.

## Page Level Commands

<u>Parameter</u>	<u>Description</u>
BORDERPADDING=number	The amount of padding to allow for the border between the page contents and the border. This value is specified in points (1/72 of an inch).
COLS=number	Sets the number of columns for the page. Default is 1. Set this to 2 or more to divide the printable area into equally spaced columns for the text to flow.
COLSPACE=number	Sets the spacing in inches or units between each of the columns.
Y=number	Used for conditional page break. A page break will only occur if the current Y position is below this value. The value is specified in inches or units from the top of the page. The PAGE tag is ignored if the current Y position is above this value.
SHADING=text	Optional. Enter the name of a <a href="#">shading pattern</a> .
PDFPAGE=number	The page number from the background PDF to use as the background for the page. Set to 0 to turn off the background PDF. Use the -pdf option or BkgPDF method to assign an existing PDF for the background.
XPSPAGE=number	The page number from the background XPS to use as the background for the page. Set to 0 to turn off the background XPS. Use the -xpsback option or XPSBack method to assign an existing XPS for the background.
CLEARBACKGROUND	Turns off the current background. See the <a href="#">BACKGROUND</a> tag for more info.

## <BACKGROUND>

Marks the page as a background page. The text for the page will not appear as a page by itself in the PDF. Any text or images will be saved and used to paint the background for the following pages. The background can be turned off by issuing a PAGE command with the CLEARBACKGROUND option .

A section marked as a background will not page break - any text that flows beyond the end of the page will simply not be shown. You may, however, use the [Y](#) tag to move up or down the page as needed. This will allow you place text in the header area then jump down to the footer area for more text or images.

## Page Level Commands

**<INSERTPDF**  
**SRC=text**  
**PAGES=text>**

Inserts a PDF. The current page is rendered and the inserted PDF starts on the next page. Any text after this tag starts on a new page after the inserted PDF.

<u>Parameter</u>	<u>Description</u>
SRC=text	The full path and file name of the PDF to insert.
PAGES=text	A comma separated list of page numbers to include. Pages are inserted in the order listed. All pages are included if this option is left out.

## <SHADING

**NAME=text**  
**COLOR1=color**  
**COLOR2=color**  
**COLOR3=color**  
**COLOR4=color**  
**COLOR5=color**  
**COLORARY=text**>

Used to define a gradient shading pattern. The shading pattern can then be used with the PAGE, BGCOLOR and RECT tags. You may specify from two to five colors.

<u>Parameter</u>	<u>Description</u>
NAME=text	The name for the shading pattern. Must be unique within the document.
COLOR1=color	The starting color. Any valid <a href="#">color</a> code.
COLOR2=color	Any valid <a href="#">color</a> code.
COLOR3=color	Optional. Any valid <a href="#">color</a> code.
COLOR4=color	Optional. Any valid <a href="#">color</a> code.
COLOR5=color	Optional. Any valid <a href="#">color</a> code.
COLORARY=text	A comma separated list of 4 numbers. The default is 0,0,1,0. These represent the $X_0, Y_0, X_1, Y_1$ matrix coordinates for the shading pattern. A matrix of 0,0,1,0 goes from left to right from COLOR1 to COLOR <sub>n</sub> . A matrix of 0,0,0,1 goes from top to bottom. You may use decimals or negative numbers as well to offset where the middle of the gradient lies.

## Text Commands

These are the variables you may use in your document.

<u>Variable</u>	<u>Sample</u>	<u>Description</u>
&amp;	&	Ampersand
&nbsp;		Space
&reg;	®	Registered trademark symbol
&trad;	™	Trademark symbol
&copy;	©	Copyright symbol
&mdash;	—	emdash
&lt;	<	Less-than symbol
&gt;	>	Greater-than symbol
&cent;	¢	Cent
&pound;	£	Pound
&euro;	€	Euro
&yen;	¥	Yen
&deg;	°	Degree
&lt;	«	Guillemet (left)
&gt;	»	Guillemet (right)
&oslash;	ø	O with slash



## Text Commands

**<BR  
VALUE=number>**

Used to insert a line break. This forces the current line to stop and the next line will begin from the left margin on the following line.

<u>Parameter</u>	<u>Description</u>
VALUE	An optional value to specify the height of the break in inches or units. The font height plus the line spacing amount is used when VALUE is not specified.

## *Text Commands*

**<P>**

Used for a new paragraph. This tag is equivalent to using <BR><BR>.

## Text Commands

```
<A  
  NAME=text  
  HREF=text>  
</A>
```

Used to add a web or file link. Also used to link to an anchor point in the PDF. Use either the NAME or HREF attribute but not both in the same A tag.

<u>Parameter</u>	<u>Description</u>
NAME=text	The name of the anchor for this place on the current page. You refer to this location with other links that reference this name in the HREF attribute. Do not use a closing </A> tag with anchors.
HREF=text	The document location such as http://www.fytek.com or doc\extfile.doc. Or, this can be an anchor in the current document. In this instance, use a # in front of the name.

Here is an example of a link: [FyTek, Inc.](#)

Here's how you might specify a link to another page in the PDF:

```
<PAGE>  
<A NAME="chap2">  
Text ... text ... text  
<PAGE>  
Index  
<A HREF="#chap2">Chapter 2</A>
```

## Text Commands

**<COMP  
VALUE=number>**

Used to specify the compression percentage for text.

<u>Parameter</u>	<u>Description</u>
VALUE=number	A percentage to compress the text by. A value less than 100 compresses text while a value greater than 100 expands text.

**This text has been expanded by 150 percent of its original size. Text will remain at this percentage until another COMP tag is issued.**

## Text Commands

### <MARGINS

**LM=number**

**RM=number>**

Changes the left and/or right margin. Only works when positioned at the left margin currently such as after a <BR> command. Also, the current page columns must be set at one.

<u>Parameter</u>	<u>Description</u>
LM=number	Sets the left margin in inches or units. Default is .5 inches.
RM=number	Sets the right margin in inches or units. Default is .5 inches.

**<TABSTOP**  
**VALUE=number**  
**BORDER=number**  
**OFFSET-LEFT=number**  
**OFFSET-BOTTOM=number**  
**ALIGN=L|R|C**  
**WIDTH=number**  
**SIZE=number>**

Used to set tab stops for printing data in columns. This command sets the position for use by the [TAB](#) command. Be sure to set all the TABSTOPS first before using the TAB command. The tab stops can be cleared with the [CLEARTABS](#) command.

<u>Parameter</u>	<u>Description</u>
VALUE=number	The position in inches or units from the left margin where the tab should be placed.
BORDER=number	The line width to use for drawing a border around the tabbed contents. Use a fraction of a number as well for a thin line (for example, 0.5). You only need to set this on one of the TABSTOP entries.
BORDERCOLOR=color	Enter a valid <a href="#">color</a> .
OFFSET-LEFT=number	An amount in points to offset the left side of the border to provide some space between the contents and the border. For example, OFFSET-LEFT=20 to position the left edges of all tab borders. You only need to set this on one of the TABSTOP entries.
OFFSET-TOP=number	An amount in points to offset the bottom line of the border to provide some space between the contents and the border. For example, OFFSET-BOTTOM=3 to position the bottom edges of all tab borders. You only need to set this on one of the TABSTOP entries.
ALIGN=L R C	The alignment to use at the tab. L=Left R=Right C=Center
WIDTH=number	The column width in inches or units when using a border. Optional if your alignment is "Left" for all of your columns but you may still use it in this case. Otherwise, it is necessary to provide this information so the program will know how to size the border.

## Text Commands

<u>Parameter</u>	<u>Description</u>
SIZE=number	The maximum text or image height in points (1/72 of an inch) which will be used at any tab. This is only needed if you are varying the point size at each tab to let the program know what the maximum height is. This will keep the text horizontally aligned for the entire line.

## Text Commands

### <TAB>

Jumps to the next tab stop set with the [TABSTOP](#) command. You must set the position for all tabs first with the TABSTOP command. The text that follows this tag up until the next TAB or BR command is placed at the position specified by TABSTOP. Once at that position, the text is aligned. The program then moves back up one line of text and positions itself for the next tab stop. The text doesn't drop down to the next line until a [BR](#) command is issued.

It's important to note you may potentially go beyond the margin tings if your TABSTOPS are not set correctly. For example, ting a TABSTOP at the left margin (VALUE=0) with a right alignment will cause the text to potentially start somewhere off the page and end up at the left margin. Too much text at a tab stop may potentially overwrite text printing at other tab positions.



## *Text Commands*

**<CLEARTABS>**

Clears out any [TABSTOP](#) tings.

## Text Commands

```
<FONT
  FACE=text
  SIZE=number
  COLOR=color
  COMP=number
  REND=number
  SRC=text
  BARCODE=text
  SUPP=text
  FIXED=number
  CODEPAGE=number
  ENCODING=text
  UNICODE>
```

Used to set the current font and/or point size.

<u>Parameter</u>	<u>Description</u>
FACE=text	Specifies the font to use. The text from the FACE column in the following <a href="#">table</a> . Note you do not have to supply the check digit when using a UPC font. Text2PDF will calculate it for you if you leave it out. You may have to adjust line spacing when using barcode fonts.
SIZE=number	The point size for the font.
COLOR=color	Enter a valid <a href="#">color</a> .
COMP=number	A percentage to compress the text by. A value less than 100 compresses text while a value greater than 100 expands text.
REND=number	0 = Fill text (default) 1 = Stroke text (outline) 2 = Fill then stroke 3 = No fill or stroke (invisible)
SRC=text	Required if this is custom font you're adding. Use any unique text string you wish for the FACE parameter in this case. The SRC only needs to be entered the first time you reference the font but it doesn't hurt to include it on further uses of the font. For TrueType fonts specify the font file (ex. "c:\windows\fonts\myfont.ttf"). Also, for OpenType fonts specify the font file (ex. "c:\windows\fonts\myfont.otf"). For Type 1 fonts specify the file name without the extension (ex. "c:\windows\fonts\myfont"). Type 1 fonts have several different files associated with them and the software will handle locating the individual files.

## Text Commands

<u>Parameter</u>	<u>Description</u>
BARCODE="text"	For use if adding your own barcode. Pass "3of9" if the barcode is a 3 of 9 barcode type. Otherwise, this can be blank.
SUPP="text"	The optional 2 or 5 character supplemental value for UPCA barcodes. Only use with FACE="upca025", "upca050", or "upca100".
FIXED=number	Use when you want to turn a variable width font into a fixed width font. Pass in the width to use for each character. The built-in courier font, for example, has a fixed width of 600 units. The larger the number, the more space will be left between characters.
CODEPAGE=number	<p>The codepage to use (1252 Windows Latin-1 is used by default). This option is valid only when embedding your own TrueType or OpenType font. Must be a codepage that is included in the font. Currently, the other codepages supported by Text2PDF are:</p> <ul style="list-style-type: none"><li>1250 (Central European)</li><li>1251 (Cyrillic)</li><li>1253 (Greek)</li><li>1254 (Turkish)</li><li>1255 (Hebrew)</li><li>1256 (Arabic)</li></ul> <p>Note you may also use the UNICODE option and pass text in UTF-8 format.</p>
ENCODING=text	The encoding to use for the custom font. WinAnsiEncoding is used if not specified. This value is inserted directly into the PDF as typed so case is important. If you are not sure what value to use, leave this option out. The default should be fine for most cases. Possible values are WinAnsiEncoding, StandardEncoding, MacRomanEncoding or PDFDocEncoding.
UNICODE	Specifies this font will be used for Unicode characters. For example, you can use this option with the arial.ttf font that is supplied with most versions of Windows to access the Unicode characters for Hebrew, Arabic, etc. See the <a href="#">DIV</a> tag to specify Right-to-Left text.

# FyTek's Text2PDF




---

## Text Commands

There are 14 built-in fonts (Times, Helvetica, Courier — each in standard, bold, italics and bold-italics — and symbol and zapfdingbats). Use the value from the "Face" column below to specify the font face you wish to use. Note you may use the <B> and <I> commands for bold and/or italics, but only with the first 12 base fonts (Courier, Times and Helvetica). Here are the built-in fonts and their values:

<u>Face</u>	<u>Name</u>	<u>Sample</u>
Courier	Courier	ABCDEFGF abcdefg 12345
Helvetica	Helvetica (Arial)	ABCDEFGF abcdefg 12345
Times	Times Roman	ABCDEFGF abcdefg 12345
Courier-Bold	Courier Bold	<b>ABCDEFGF abcdefg</b> <b>12345</b>
Helvetica-Bold	Helvetica Bold	<b>ABCDEFGF abcdefg 12345</b>
Times-Bold	Times Roman Bold	<b>ABCDEFGF abcdefg 12345</b>
Courier-Italics	Courier Italics	<i>ABCDEFGF abcdefg</i> <i>12345</i>
Helvetica-Italics	Helvetica Italics	<i>ABCDEFGF abcdefg 12345</i>
Times-Italics	Times Roman Italics	<i>ABCDEFGF abcdefg 12345</i>
Courier-Bold-Italics	Courier Bold-Italics	<b><i>ABCDEFGF abcdefg</i></b> <b><i>12345</i></b>
Helvetica-Bold-Italics	Helvetica Bold-Italics	<b><i>ABCDEFGF abcdefg 12345</i></b>
Times-Bold-Italics	Times Roman Bold-Italics	<b><i>ABCDEFGF abcdefg 12345</i></b>
Symbol	Symbol	ABXΔEΦΓ αβχδεφγ 12345
ZapfDingbats	Zapf Dingbats	☆♣%⊗⊕◆◇ ✽*✽*✽*✽*✽* ✍✎✓✓✕
3of9	3 of 9 Barcode (Must use the -barcodes option or Barcodes method to include this font in your PDF)	 0 1 2 3 4 5 A B C A B C

## Text Commands

<u>Face</u>	<u>Name</u>	<u>Sample</u>
UPCA025	UPC Barcode (Must use the -barcodes option or Barcodes method to include this font in your PDF)	
UPCA050	UPC Barcode (Must use the -barcodes option or Barcodes method to include this font in your PDF)	
UPCA100	UPC Barcode (Must use the -barcodes option or Barcodes method to include this font in your PDF)	

The following Asian fonts are also available. You'll need to install the Chinese, Japanese or Korean font packs from Adobe in order to view a PDF with these characters. The font packs are available (free of charge) at:

<http://www.adobe.com/products/acrobat/acrrasianfontpack.html>

Only use these fonts if your input has been converted to the encodings shown. Do not use for UTF-8 encoded files. If your file is in UTF-8 format and you have a META tag to specify that or are using -utf8 (command line) or UTF8 (method) then you should not need to specify these fonts in your input. The correct font will be used in this case.

C1	STSong-Light (Chinese font) GBK-EUC-H encoding
C1B	STSong-Light Bold (Chinese font)
C1I	STSong-Light Italics (Chinese font)
C1BI	STSong-Light Bold-Italics (Chinese font)
C2	MSung-Light (Chinese font) ETen-B5-H encoding
C2B	MSung-Light Bold (Chinese font)
C2I	MSung-Light Italics (Chinese font)
C2BI	MSung-Light Bold-Italics (Chinese font)
C3	MSung-Light (Chinese font) ETen-B5-H encoding

## Text Commands

<u>Face</u>	<u>Name</u>	<u>Sample</u>
C3B	MHei-Medium Bold (Chinese font)	
C3I	MHei-Medium Italics (Chinese font)	
C3BI	MHei-Medium Bold-Italics (Chinese font)	
J1	HeiseiMin-W3 (Japanese font) 90ms-RKSJ-H encoding	
J1B	HeiseiMin-W3 Bold (Japanese font)	
J1I	HeiseiMin-W3 Italics (Japanese font)	
J1BI	HeiseiMin-W3 Bold-Italics (Japanese font)	
J2	HeiseiKakuGo-W5 (Japanese font) 90ms-RKSJ-H encoding	
J2B	HeiseiKakuGo-W5 Bold (Japanese font)	
J2I	HeiseiKakuGo-W5 Italics (Japanese font)	
J2BI	HeiseiKakuGo-W5 Bold-Italics (Japanese font)	
K1	HYGoThic-Medium (Korean font) KSC-EUC-H encoding	
K1B	HYGoThic-Medium Bold (Korean font)	
K1I	HYGoThic-Medium Italics (Korean font)	
K1BI	HYGoThic-Medium Bold-Italics (Korean font)	
K2	HYSMyeongJo-Medium (Korean font) KSC-EUC-H encoding	
K2B	HYSMyeongJo-Medium Bold (Korean font)	
K2I	HYSMyeongJo-Medium Italics (Korean font)	
K2BI	HYSMyeongJo-Medium Bold-Italics (Korean font)	

## Text Commands

### <DIV

#### DIR=text>

Used to set the text flow as Right-to-Left for Unicode. The Unicode text must be encoded as UTF-8 (or be in the `&#9999;` (decimal), `&#x9999;` (hexadecimal) or `&#o9999;` (octal) format).

In addition, your input file (or the commands you send using PDFCmd) must start with the 3 hex characters EF BB BF (ASCII 239 187 191). You'll need to use the `-utf8` option or UTF8 method if you have UTF-8 text but do not have the required 3 hex characters at the beginning of the file. This is so the program knows to perform the required preprocessing of the text.

<u>Parameter</u>	<u>Description</u>
DIR=text	Specify RTL for Right-to-Left text. Use LTR to return to the default Left-to-Right text flow. This only affects UTF-8 text. Note the default alignment also changes to Right when RTL is specified.

## Text Commands

**<ALIGN  
VALUE=text>**

Used to set the current alignment. The ALIGN value affects the current line so you should place a BR before a long section of text using the ALIGN tag.

Parameter

Description

ALIGN=L|R|C|J    Sets the alignment to Left, Center, Right or Justified.

For example:

<BR><ALIGN VALUE=L>

Text on the left

<BR><ALIGN VALUE=C>

Centered Text

<BR><ALIGN VALUE=R>

Text on the right<BR>

</TEXT>

Text on the left

Centered Text

Text on the right



## *Text Commands*

**<SUB>**  
**</SUB>**

Used to turn subscripting on and off.

Here is a line using the <sub>subscripting</sub> tag.

## *Text Commands*

**<SUP>**  
**</SUP>**

Used to turn superscripting on and off.

Here is a line using the <sup>superscripting</sup> tag.

## Text Commands

**<LINESPACE  
VALUE=number>**

Used to specify the line spacing in 1/72 of an inch.

<u>Parameter</u>	<u>Description</u>
VALUE=number	The amount of space between lines of text specified in units of 1/72 of an inch. Default is 2.

This text has a linespace value set at 12 which is one more than the current point size of 11. Notice how far each line drops down when the text wraps.

This is roughly double spaced text.

## Text Commands

```
<IMG  
  SRC="text"  
  WIDTH=number  
  HEIGHT=number  
  XL|XR=number  
  YT|YB=number  
  BORDER=number  
  BORDERCOLOR=number  
  KEEPRATIO  
  SMASK  
  CACHE  
  HIRES>
```

Used to insert an image. This can be a jpeg, gif, tif, or png. Not all variations of the supported image formats are supported so you may need to modify the format if the image does not show. Images scanned at over 72 dpi should have WIDTH and HEIGHT set to scale the size of the image down.

<u>Parameter</u>	<u>Description</u>
SRC="text"	Required. The path and file name of the image you wish to include. The image itself will be embedded once in the PDF regardless of the number of times it is actually displayed. Place quotes around this value.
WIDTH=number	The amount to compress or expand the image by in the X direction if a % is used after the number. Values less than 100 will compress and values greater than 100 will expand. The value in points (1/72 of an inch) is taken as the width if a % is not used.
HEIGHT=number	The amount to compress or expand the image by in the Y direction if a % is used after the number. Values less than 100 will compress and values greater than 100 will expand. The value in points (1/72 of an inch) is taken as the height if a % is not used.

## Text Commands

<u>Parameter</u>	<u>Description</u>
XL XR=number	<p>Use the XL XR and YT YB options when you want to place the image on the page as a background at a precise location and not inline with text. Text or other images do not flow around the image in this case. Nor is the current text position modified. The image will be placed at the current text position on the page when XL XR &amp; YT YB options are not used. Use either XL or XR but not both. XL is the position from the left edge of the page in inches or units to place the left edge of the image. XR is the position from the right edge of the page in inches or units to place the right edge of the image. The WIDTH setting in inches or units can be used to set the width of the image. Note this is slightly different from the point or percentage definition of WIDTH without the X/Y options. The default width is simply the width of the image.</p> <p>For example, <code>&lt;IMG SRC="myimage.jpg" XL=1 YT=2 WIDTH=2 HEIGHT=3&gt;</code> will place the image 1 inch from the page left edge and the image top 2 inches down from the page top with a width of 2 inches and a height of 3 inches.</p>
YT YB=number	<p>Used along with the XL or XR setting. Use either YT or YB but not both. YT is the position from the top edge of the page in inches or units to place the top edge of the image. YB is the position from the bottom edge of the page in inches or units to place the bottom edge of the image. The HEIGHT setting in inches or units can be used to set the height the of the image. Note this is slightly different from the point or percentage definition of HEIGHT without the X/Y options. The default height is simply the height of the image.</p>
BORDER=number	<p>Adds a border to image using the specified thickness. Set the value of BORDER to the thickness of the border.</p>
BORDERCOLOR=color	<p>Specify the border color by name (black, white, blue, ...) or in HEX format (#99CC33, #7B68EE, ...).</p>
KEEPRATIO	<p>Preserves the aspect ratio when shrinking the image to fit on a page.</p>

## Text Commands

<u>Parameter</u>	<u>Description</u>
SMASK	Specify this option to use the included soft-mask (transparency) for PNG files. This is a secondary image used to define the transparency for the PNG. Not all PNGs contain this information so only use this option if the transparency is stored with the PNG and you require the transparency. Note that using this option alters the graphics state for the page and tends to cause normal text on the page to render darker than usual.
CACHE	Use this option on PNG, TIF, or GIF images that do not change often but are often included in your PDF files. This will cut down on the time needed to convert the image to PDF format. For example, you might use this for logos or similar images that are static and included in multiple runs of the program. A converted image file is stored in the temporary directory with an extension of .rw\$. The cached version is automatically re-created if the size or date/time stamp changes on the base image.
HIRES	May need to include this tag on certain hi-resolution jpeg images. Do not specify this option unless the image is not displaying properly without it.

## Text Commands

**<BGCOLOR  
VALUE=color  
SHADING=text>**

Used to set the background color for highlighting text.

<u>Parameter</u>	<u>Description</u>
VALUE=color	Enter a valid <a href="#">color</a> . The color is used to outline, rather than fill, the area when used with a shading pattern. You may choose to use only a shading pattern and leave the VALUE option out.
SHADING=text	Optional. Enter the name of a <a href="#">shading pattern</a> .

Here is some text that has been highlighted using the BGCOLOR command.

Using these values:

```
<SHADING NAME="bluered" COLOR1=#FF6C9F COLOR2=#DDEEFF  
COLOR3=#9CCFFF COLORARY="1,.8,0,.3">
```

Here is some text that has been shaded using the BGCOLOR command.

## *Text Commands*

**<B>**

**</B>**

Used to turn bold face font on and off. You may also use a **<STRONG>** tag in place of this. Only works with the built-in font faces - Courier, Helvetica and Times Roman fonts.

Here is some **bold** text.



## Text Commands

`<I>`

`</I>`

Used to turn italics on and off. You may also use a `<EM>` tag (emphasis) in place of this. Only works with the built-in font faces - Courier, Helvetica and Times Roman fonts.

Here is some *italics* text.

## *Text Commands*

**<U>**

**</U>**

Used to turn underlining on and off.

Here is a line with an underline in it.

## Text Commands

```
<PRE  
  PLAIN|HTML>  
</PRE>
```

Sets preformatted mode. In this mode, spaces are retained and lines break based on the input data. A form feed (ASCII 12) will start a new page.

Text is always left aligned in a PRE block. Use the PLAIN option for text without tags. This allows a faster build but no parsing of tags or checks for line wrap are performed. Be sure to select a courier or fixed spaced font in this case.

Use the HTML option for plain text with tags. Note this is different from just PRE without HTML as line breaks will not occur for long text lines when using HTML.

<u>Parameter</u>	<u>Description</u>
PLAIN	Optional. Text is plain text with no tags so none will be parsed. No check is made if text goes beyond right margin.
HTML	Optional. Text is plain text with tags. No check is made if text goes beyond right margin.

## Text Commands

**<HR**  
**WIDTH=number**  
**COLOR=color>**

Draws a horizontal rule (line) from the left margin to the right margin.

<u>Parameter</u>	<u>Description</u>
WIDTH=number	The width of the line to draw in points (1/72 of an inch).
COLOR=color	The <a href="#">color</a> of the line.

## Text Commands

**<LINE**  
**X1=number**  
**Y1=number**  
**X2=number**  
**Y2=number**  
**WIDTH=number**  
**COLOR=color**  
**TOP**  
**FG>**

Draws a line from the point X1, Y1 to X2, Y2.

<u>Parameter</u>	<u>Description</u>
X1=number	X1 is the position from the left edge of the page in inches or units.
Y1=number	Y1 is the position from the bottom edge of the page in inches or units. Or, Y1 is the position from the top edge of the page when the TOP option is used.
X2=number	X2 is the position from the left edge of the page in inches or units.
Y2=number	Y2 is the position from the bottom edge of the page in inches or units. Or, Y2 is the position from the top edge of the page when the TOP option is used.
WIDTH=number	Optional. The width of the line to draw in points (1/72 of an inch).
COLOR=color	Optional. The <a href="#">color</a> of the line.
TOP	Optional. Used to specify the top of the page is Y position 0. The bottom of the page is considered 0 by default for the Y1 and Y2 values without this option.
FG	Optional. Used to specify the line should be drawn in the foreground. Drawing items are placed on the page first by default so text can be placed overtop. This option forces the items to be drawn after any text or images.

## Text Commands

**<RECT**  
**X1=number**  
**Y1=number**  
**X2=number**  
**Y2=number**  
**WIDTH=number**  
**COLOR=color**  
**BGCOLOR=color**  
**SHADING=text**  
**TOP**  
**FG>**

Draws a rectangle with corners at the points X1, Y1 and X2, Y2.

<u>Parameter</u>	<u>Description</u>
X1=number	X1 is the position from the left edge of the page in inches or units.
Y1=number	Y1 is the position from the bottom edge of the page in inches or units. Or, Y1 is the position from the top edge of the page when the TOP option is used.
X2=number	X2 is the position from the left edge of the page in inches or units.
Y2=number	Y2 is the position from the bottom edge of the page in inches or units. Or, Y2 is the position from the top edge of the page when the TOP option is used.
WIDTH=number	Optional. The width of the rectangle to draw in points (1/72 of an inch).
COLOR=color	Optional. The <a href="#">color</a> of the rectangle.
BGCOLOR=color	Optional. The background fill <a href="#">color</a> for the rectangle.
SHADING=text	Optional. Enter the name of a <a href="#">shading pattern</a> .
TOP	Optional. Used to specify the top of the page is Y position 0. The bottom of the page is considered 0 by default for the Y1 and Y2 values without this option.
FG	Optional. Used to specify the rectangle should be drawn in the foreground. Drawing items are placed on the page first by default so text can be placed ovetop. This option forces the items to be drawn after any text or images.

## Text Commands

**<CIRCLE**  
**X=number**  
**Y=number**  
**RADIUS=number**  
**ASPECTX=number**  
**ASPECTY=number**  
**WIDTH=number**  
**COLOR=color**  
**BGCOLOR=color**  
**TOP**  
**FG>**

Draws a circle centered at the point X, Y with a radius of Z.

<u>Parameter</u>	<u>Description</u>
X=number	X is the position from the left edge of the page in inches or units.
Y=number	Y is the position from the bottom edge of the page in inches or units. Or, Y is the position from the top edge of the page when the TOP option is used.
RADIUS=number	The radius of the circle in inches or units.
ASPECTX=number	Optional. Multiplier to compress or expand the circle along the X axis. A number between 0 and 1 compresses while numbers greater than 1 expand.
ASPECTY=number	Optional. Multiplier to compress or expand the circle along the Y axis. A number between 0 and 1 compresses while numbers greater than 1 expand.
WIDTH=number	Optional. The width of the circle to draw in points (1/72 of an inch).
COLOR=color	Optional. The <a href="#">color</a> of the circle.
BGCOLOR=color	Optional. The background fill <a href="#">color</a> for the circle.
TOP	Optional. Used to specify the top of the page is Y position 0. The bottom of the page is considered 0 by default for the Y value without this option.
FG	Optional. Used to specify the circle should be drawn in the foreground. Drawing items are placed on the page first by default so text can be placed overtop. This option forces the items to be drawn after any text or images.

## Text Commands

**<X  
VALUE=number>**

Used to change the X position in a block of text.

<u>Parameter</u>	<u>Description</u>
VALUE=number	<p>The position in inches or units of where to move horizontally from the left margin. The current Y position remains the same. Remaining text will drop down to the next line if you go beyond the right margin. The X position is set back to the left margin on a line break or BR command. Use the option -noresetx to override that functionality so the X position is not reset on a line break or BR.</p> <p>May also set as a percentage such as 50%. The X value will be computed by taking the printable area (page width - margins) times the percentage value and adding to left margin.</p>



## Text Commands

### **VALUE=number>**

Used to change the Y position in a block of text.

<u>Parameter</u>	<u>Description</u>
VALUE=number	<p>The position in inches or units of where to move vertically from the top of the page. Do not set below the margin unless you're on the <a href="#">background</a> page. Setting below the bottom margin will cause a page break otherwise.</p> <p>May also set as a percentage such as 50%. The Y value will be computed by taking the page height times the percentage value.</p>

## Text Commands

### <UNITS

VALUE=in|cm|mm|pt>









Used to set the value for units. The command affects all following commands that use a value based on the units ting.

<u>Parameter</u>	<u>Description</u>
VALUE=text	Use one of the following: in = inches (default) cm = centimeters mm = millimeters pt = point (1/72 of an inch)

## Colors

Parameters such as COLOR or BGCOLOR take a color value. Colors may be entered in any of the following ways:

- You may specify the red, green and blue components as decimal values from 0 to 1, separated by a comma. In this case 0,0,0 is black and 1,1,1 is white.
- 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.
- You may specify the cyan, magenta, yellow and black components (CMYK) as decimal values from 0 to 1, separated by a comma. In this case 0,0,0,1 or 1,1,1,0 is black and 0,0,0,0 is white.
- You may specify the cyan, magenta, yellow and black components (CMYK) as decimal values from 0 to 1, separated by a comma. In this case 0,0,0,255 or 255,255,255,0 is black and 0,0,0,0 is white.
- You may specify one of the RGB colors in the table below.

Color	Name
	Black
	Silver
	Gray
	White
	Maroon
	Red
	Purple
	Fuchsia

Color	Name
	Green
	Lime
	Olive
	Yellow
	Navy
	Blue
	Teal
	Aqua

## Initialization File

An initialization file is used to hold initial settings and conversions from the input text to other text or tags. This is a separate file from the input that you include via the `-ini "file"` (or `-inirun "file"`) command line or `IniFile("file")` DLL method.

The layout for the file is an XML-like format containing tags such as `FONT` or `REPLACE`. All tags must appear only once per initialization file with the exception of the `REPLACE` tag which can be repeated. Use this tag to convert text in the input file from one string to another or to a tag. For example, escape sequences in the input file can be converted to tags for text underline, font size/color changes or any other text replacement.

The following tags are allowed in the Initialization File:

<code>COLONE</code>	Used to specify the input file has print instructions in column 1.
<code>FONT</code>	Used to specify the initial font for the PDF.
<code>LINESPACE</code>	Used to specify the initial line spacing.
<code>PAGE</code>	Used to specify the initial page and margin size.
<code>PAGENUM</code>	Used to specify the page numbering and format.
<code>PDF</code>	Used to specify a background PDF and page.
<code>PRE TEXT HTML</code>	Used to specify the initial file format.
<code>REPLACE</code>	Used to specify the text replacement.
<code>UNITS</code>	Used to specify the unit of measure for other tags in this file.

The following section details each tag. Some tags (like `FONT`) are duplicates of tags you can include in the input file. A link is available from this section to the corresponding tag in the Commands Section in this case.

## *Initialization File*

### <COLONE>

This tag is only valid in the initialization file - not in the source text file.

Used to specify the first column contains a vertical print instruction. Same as using -c1 on the command line or the DLL method ColOne.

## *Initialization File*

### **FONT**

Allows for the initial FONT setting. Only use one FONT tag in the initialization file.

See the [FONT](#) tag in the Commands Section for details.

## *Initialization File*

### **LINESPACE**

Allows for the initial LINESPACE setting.

See the [LINESPACE](#) tag in the Commands Section for details.

## *Initialization File*

### **PAGE**

Allows for the initial PAGE size and margin settings.

See the [PAGE](#) tag in the Commands Section for details.



## Initialization File

```
<PAGENUM  
  X=number  
  Y=number  
  FORMAT="text"  
  FACE="text"  
  SIZE=number>
```

This tag is only valid in the initialization file - not in the source text file.

Specifies page numbers should be printed along with their location and format.

<u>Parameter</u>	<u>Description</u>
X=number	The X position in inches or units from the left edge of the page.
Y=number	The Y position 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.
FORMAT="text"	Text for page numbering string. Use %p for the current page. For example, "Page: %p". Use two %'s in a row if you're running the program from within a batch file (i.e. "Page: %%p"). You may also use %t for total pages. Using %t will require more processing time since the PDF must be built twice - once to determine the number of pages then once again to use the number in the string for the final output.
FACE="text"	Sets the font for the page numbering string. Pass the font name (must be one of the standard Courier, Helvetica or Times fonts). See the <a href="#">FONT</a> tag for the list of font names.
SIZE=number	The point size for the page number font.

```
<PDF  
  SRC="text"  
  PAGE=number  
  PASSWORD="text">
```

This tag is only valid in the initialization file - not in the source text file. Used to specify the background PDF. Same as using the -pdf or BkgPDF method.

You may use Excel, Word, or PowerPoint files as well under one of the following two conditions. First, if you have Excel/Word 2007 or higher with the free Office "Save as PDF" add-on installed, the Windows version of Text2PDF can connect to Excel or Word to perform the conversion.

Second, OpenOffice can be used to convert these document types to PDF. Under Windows, Text2PDF will attempt to connect to OpenOffice using the Windows COM sub-system to perform the conversion. This means that the standard OpenOffice installation should be all you need on Windows so Text2PDF can convert Excel/Word documents. The command line utility "unoconv" (part of OpenOffice) can also be used on any operating system if you have OpenOffice installed as a server. Be sure the PATH environment variable for the user running Text2PDF contains the location where unoconv is located. This is useful on Linux systems to convert Excel/Word to PDF. See the OpenOffice documentation for instructions on setting up the OpenOffice server.

While you can manually use one of these options to convert to PDF first then pass the PDF to Text2PDF, it may be more convenient to let Text2PDF run this process for you by simply passing your Office document to Text2PDF. Users do not need anything special on their local machine to read or convert Excel/Word to PDF as the process will be handled by the Text2PDF server with one of these options setup on a server running Text2PDF in Client-Server TCP/IP mode.

<u>Parameter</u>	<u>Description</u>
SRC="text"	The path and file name of the background PDF to use. Note that the background PDF may contain page rotations or similar viewer settings that will be discarded when used as a background. For example, the PDF may have its contents displayed upside down if the original PDF contained a page rotation of 180 degrees. You'll need to modify the PDF either in the application that created it or by using an application such as FyTek's PDF Meld (visit <a href="http://www.fytek.com">http://www.fytek.com</a> for information on PDF Meld).

## *Initialization File*

<u>Parameter</u>	<u>Description</u>
PAGE=number	Page number from background PDF to use. This is optional and can be set when needed by the PDFPAGE option on the <a href="#">PAGE</a> tag.
PASSWORD="text"	The owner password for the background PDF. This is only needed when the background PDF is encrypted.

## *Initialization File*

### **PRE**

Allows for the initial PRE setting. See the [PRE](#) tag in the Commands Section for details.

Use <PRE HTML> if you are using the COLONE option (-c1 command line or ColOne DLL method) and are using the REPLACE tag to insert markup tags (like B or FONT) in the report. This will ensure the tags are processed and treated as plain text.

```
<REPLACE
  FROMTEXT="text"
  FROMHEX="text"
  TOTEXT="text"
  TOHEX="text"
  TOTAG="text"
  TOTAG2="text"
  SKIPLINE
  WILDCARD
  REGEX
  IGNORECASE
  FIRSTONLY
  FROMCOL=number
  THRUCOL=number
  KEEPBEFORE
  KEEPAFTER>
```

This tag is only valid in the initialization file - not in the source text file.

Specifies a replacement string or tag for input text. The FROMTEXT or FROMHEX options are used to specify the source text string. The TOTEXT, TOHEX and TOTAG options are used to specify the resulting text string.

You may have as many REPLACE tags in the initialization file as you want. Each REPLACE tag must have either a FROMTEXT or FROMHEX (but not both) along with one of TOTEXT, TOHEX or TOTAG. Other optional parameters may be used as well.

<u>Parameter</u>	<u>Description</u>
FROMTEXT="text"	A text string to search for in the input file. This text is case sensitive unless IGNORECASE is used. Wherever this text is found, it will be replaced with the TOTEXT, TOHEX or TOTAG text. Do not use FROMHEX along with this option.
FROMHEX="text"	A text string to search for in the input file specified in HEX. Use values from 00 to FF for each character. For example, "1F05ca" is a 3 character string consisting of 1F, 05 and CA in hex. The hex letters themselves are not case sensitive. Wherever this text is found, it will be replaced with the TOTEXT, TOHEX or TOTAG text. Do not use FROMTEXT along with this option.
TOTEXT="text"	The text string to replace the FROMTEXT or FROMHEX text with. Do not use TOHEX or TOTAG along with this option.

## Initialization File

<u>Parameter</u>	<u>Description</u>
TOHEX="text"	The text string (in hex) to replace the FROMTEXT or FROMHEX text with. Use values from 00 to FF for each character. Do not use TOTEXT or TOTAG along with this option.
TOTAG="text"	A tag to replace the FROMTEXT or FROMHEX text with. The opening < and closing > will be automatically added so you don't need to include them. For example, set TOTAG="font COLOR='blue'" to have the FROMTEXT or FROMHEX replaced with <font COLOR='blue'>. You may use a two pipes in a row (the   character) to separate multiple tags. For example: TOTAG="page  font COLOR='black'" will become: <page><font COLOR='black'>. Do not use TOTEXT or TOHEX along with this option.
TOTAG2="text"	For use with the TOTAG option. An optional closing tag to use when you want to surround the FROMTEXT or FROMHEX text with a set of tags. The matched text is kept in this case and is placed between the TOTAG and TOTAG2 values. For example, if FROMTEXT="ABC" and TOTAG="U" and TOTAG2="/U" then all instances of ABC will come out underlined in the PDF. Do not use TOTEXT or TOHEX along with this option.
SKIPLINE	Skip the line if the FROMTEXT or FROMHEX text is found. Treats the line as if it isn't in the input file. There is no need to specify a TOTEXT, TOHEX or TOTAG option in this case.

## Initialization File

<u>Parameter</u>	<u>Description</u>
WILDCARD	<p>Optional, for use with FROMTEXT and FROMHEX. Allows the use of a "*", "." and "?" in the FROMTEXT or FROMHEX for character matching, rather than the character itself.</p> <ul style="list-style-type: none"><li>• The "." is used to match any single character.</li><li>• The "*" is used to match zero or more characters.</li><li>• The "?" is used to match zero or more characters up until the first occurrence of the character following the "?".</li></ul> <p>With the exception of ".", don't use two or more wildcard characters next to one another without some other character in-between. For example, you can use FROMTEXT="ESC.1" to match "ESC", followed by any 2 characters, then a "1", such as "ESC%C1".</p> <p>The "*" will match zero or more characters. In this case, FROMTEXT="-*-" will match "--", "---", "-abc-", "-test- -abc-" or any text found between two dashes (as shown in red) including other sets of dashes you may not want to match so use with care.</p> <p>Use a "?" to limit the match. For example, FROMTEXT="-?-" will match "-test-" and "-test- -abc-" but the latter will be 2 separate matches rather than 1 as with "*".</p> <p>Use a slash (the \ character) in front of any *, . or ? you want treated as itself rather than a wildcard. For example, FROMTEXT="\*?\*?\*" will match "****", "**abc**", or anything with at least 2 *'s at the front and back. The * is not treated as a wildcard in this case.</p>
REGEX	<p>Allows you to use a regular expression for the FROMTEXT option. For example, FROMTEXT="\d+" to match one or more numbers. Or FROMTEXT="^[A-Z].*?ing " to match all words starting with a capital letter and ending with ing that appear at the start of a line. Any valid Perl regular expression may be used.</p>
IGNORECASE	<p>Makes the search string case insensitive. This applies to both FROMTEXT and FROMHEX.</p>
FIRSTONLY	<p>Matches only the first occurrence found on the line rather than all (the default).</p>

## Initialization File

<u>Parameter</u>	<u>Description</u>
FROMCOL=number	Optional, looks for the FROMTEXT or FROMHEX starting at the specified text column number. This is the starting substring position to search for matching text in the line. The first character on a line is counted as column 1.
THRUCOL=number	Optional, looks for the FROMTEXT or FROMHEX only up to the specified text column number. This is the ending substring position to search for matching text in the line. The first character on a line is counted as column 1.
KEEPBEFORE	Optional, for use with TOTAG. Using this option will keep the text specified with FROMTEXT or FROMHEX and place it before the tag insertion. The TOTAG is inserted in this case after the original text.
KEEPAFTER	Optional, for use with TOTAG. Using this option will keep the text specified with FROMTEXT or FROMHEX and place it after the tag insertion. The TOTAG is inserted in this case before the original text.



## Initialization File

Here is an example of the REPLACE tag:

1. <REPLACE FROMTEXT="z1" TOTAG="U">
2. <REPLACE FROMTEXT="z0" TOTAG="/U">
3. <REPLACE FROMTEXT="Page:"  
TOTAG="page| |IMG SRC='hdr.jpg' " KEEPAFTER>
4. <REPLACE FROMTEXT="c15\*" TOTEXT="Duplicate">
5. <REPLACE FROMTEXT="reg()" TOTEXT="®">
6. <REPLACE FROMHEX="1F65CC" TOTEXT="Copy">
7. <REPLACE FROMHEX="55552C" TOHEX="223F0A">
8. <REPLACE FROMHEX="----\*" TOTAG="HR" WILDCARD>

Numbers 1 and 2 will replace *z1* with <U> and *z0* with </U>.

Number 3 will replace *Page:* with <page><IMG SRC='hdr.jpg'>Page:.

Number 4 will replace *c15\** with the word Duplicate.

Number 5 will replace *reg()* with &reg; (to print the ® symbol). Note that you can use any of the & [variables](#) in the text string. Use the &amp; variable if you really want the & character instead of processing the value.

Number 6 will replace the 3 character string of 1F65CC (in hex) with the word Copy.

Number 7 will replace the 3 character hex string 55552C with the 3 characters in the hex string 223F0A.

Number 8 will replace any four dashes followed by anything else to the end of the line with an <HR> tag (horizontal rule or line).

## *Initialization File*

### **UNITS**

Allows for the initial UNITS setting for use with other tags in the initialization file, such as PAGE.

See the [UNITS](#) tag in the Commands Section for details.

## Initialization File

```
<XPS  
  SRC="text"  
  PAGE=number
```

This tag is only valid in the initialization file - not in the source text file. Used to specify the background XPS. Same as using the -xpsback or BkgXPS method.

<u>Parameter</u>	<u>Description</u>
SRC="text"	The path and file name of the background XPS file to use.
PAGE=number	Page number from background XPS file to use.

## Client-Server TCP/IP

Running Text2PDF as a server is a way to startup the program and have it remain idle until it receives a request via TCP/IP to build a PDF. Once it completes its request it will process any other waiting requests (unless a pool is specified to allow multiple users at once) until there are no more. The program will then go back into wait mode until another request comes in. The advantage to running Text2PDF this way is you bypass the startup time for each run of the program. This may not be an issue if you perform a few builds each day but if you are running hundreds it could add up. In addition, the processing happens on the server so client machines are not using CPU time building reports.

There are several programs (note do not include the .exe under Unix) used for running in server mode. They are:

### Server Programs

text2pdf.exe (or text2pdf64.exe) - used to start a server from the command line (by passing -server as the first option)

text2pdf\_srv.exe (or text2pdf\_srv64.exe) - used to install a server as a Windows service

### Client Programs

text2pdf\_tcp.exe (or text2pdf\_tcp64.exe) - used to submit a client request to the server

text2pdf\_gui\_tcp.exe (or text2pdf\_gui\_tcp64.exe) - same as text2pdf\_tcp.exe but with a progress dialog box

The -server option is used to start up Text2PDF in server mode like this:

```
C:\>text2pdf -server -v -pool 5
-log "c:\logs\t2plog.txt" -host "localhost"
-port 7080 -licname "fytek-inc" -licpwd "abc12345"
-licweb
```

This starts the program in a DOS or Unix command session where it will remain until cancelled or a -quit command is sent. The preferred way to run under Windows is to use the [Text2PDF Service](#). The program is installed as a Windows service that any user with network access and permission may use. This section contains all the options that apply to both this method and the service.

You can run in the background like this (note the & at the end of the command) on Unix platforms:

```
$ text2pdf -server -v -pool 5 &
```

```
-log "/logs/t2plog.txt" -host "localhost"  
-port 7080 -licname "fytek-inc" -licpwd "abc12345"  
-licweb &
```

The program will startup and wait for commands on the specified port (7080 is the default if not set). The `-server` option must be the first option passed to the program. In addition, you must pass in your subscription (`-licname` and `-licpwd` options) or server key (`-kn` and `-kc` options). You do not need to include the subscription or server key information on client requests.

Text2PDF should then start and wait for commands. You issue commands by sending them to the TCP/IP port. Requests will be handled in sequence as they arrive unless the `-pool` option is used. This may allow for much faster processing as the program is already running in the background waiting for a request rather than starting up a separate process, performing its task, then shutting back down each time.

You may use any program to send the commands to the TCP/IP port. Pass `BUILDPDF` followed by a line feed (ASCII 10) to the port to indicate all information has been sent and Text2PDF should start processing. Or you may use the included `text2pdf_tcp.exe` (`text2pdf_tcp` on Unix) to perform the call to the server. This program will take care of opening the port, sending the parameters you give it and including the `BUILDPDF` command. The program `text2pdf_tcp.exe` does not build the output - it simply sends the commands to the port for processing by the server. To use `text2pdf_tcp`, first start the server as described above. Pass any valid Text2PDF commands to `text2pdf_tcp` and optionally include the `-host` and `-port` commands. Here is an example:

```
text2pdf_tcp -host localhost -port 7080  
file1.txt fileout.pdf
```

If the Text2PDF server is running, it will process the command otherwise an error will be returned. Note that the Text2PDF server is processing the request so you may need to provide the full path of your input and output files otherwise file names will be relative to the directory where the Text2PDF server is running. You can also use the `-cwd` or `-currdir` options to change the working directory. Also, the files must be available from the server rather than the client. That is, if you are sending commands from a local Windows client to process on a Linux server, the PDF files must be available on the Linux server (vs. the Windows client) since Text2PDF is running on the Linux server in this case. The file pathing in this example should be based on the Linux directory structure and not Windows.

You may wish to send input files to the server if the Text2PDF server is running on a different computer from the client. To send files to the server for processing you will need to pass them to the TCP/IP port with a special syntax if you are writing your own program (`text2pdf_tcp` handles this behind the scenes for you). Issue the command `-send --binaryname--_l~`

`_filename>--binarybegin--<binary data here>--binaryend--`. Note that base64 encoding may be used as well - substitute the text "base64" for "binary" in `binaryname`, `binarybegin`, and `binaryend`. The `<filename>` must match the name of a file being processed as input. The binary data for that file can come from any file on the client you wish to use to represent that file. For example, here's how you would pass 2 files (using Perl syntax):

```
use IO::Socket;
my $host = 'localhost'; # host server is running on
my $port = '12345'; # port server is running on
my $sock = new IO::Socket::INET (
    PeerAddr => $host,
    PeerPort => $port,
    Reuse => 1,
    Type => SOCK_STREAM,
    Proto => 'tcp',
);
print $sock "a.txt -pdf b.pdf t.pdf -send
--binaryname--a.pdf--binarybegin--(a.txt contents)--binaryend--";
print $sock " -send
--binaryname--b.pdf--binarybegin--(b.pdf contents)--binaryend--";
print $sock "\nBUILDPDF\n";
```

You may use the option `-return` to receive the file back via TCP/IP from the Text2PDF server. Specify the path and file name you wish to store the output under on the client. The output will not be stored on the server in this case. This allows you to receive the output PDF on the client side that you can then save or process accordingly. Of course, the larger your files the longer it will take to process as your connection speed will play a role in the time it takes to send and receive large PDFs.

The `text2pdf_tcp` program makes it easier to accomplish the above when transferring files. You may use `-send filename.pdf` where `filename.pdf` is the name of the PDF to send. The program will take care of sending the contents of the file in this case. For example:

```
text2pdf_tcp a.txt t.pdf -send a.txt=c:\myfile.txt
-send b.pdf=c:\bkg.pdf -return c:\out.pdf
```

In this case, `a.txt` is the client file `c:\myfile.txt` and `b.pdf` is `c:\bkg.pdf`. The output as referenced by `t.pdf` on the server will be sent back to the client and saved as `c:\out.pdf`. The file `t.pdf` will not be stored on the server in this case. The data will come back over the same socket connection as binary data if you are writing your own program to communicate with the server. The content length will be passed back first formatted as "Content-Length: n" where n is the number of bytes followed by a blank line and then the data stream. Once the port is closed that is the end of the file.

You may use `-sendcache filename.pdf` to send the file only the first time you call the server program. The `filename.pdf` should be the same path and file

name of one of your input files. The server will cache the file the next time you need it on future calls to the server. Include the `-sendcache` option each time you run the program with the file name even though the file will only need to be transmitted once. This can be useful when you have the same background PDF, for example, you wish to reuse many times.

Do not include interactive options such as `-open` as part of the commands sent to process unless Text2PDF server is running locally. Otherwise, the PDF will open on the remote server which is probably not what you intend.

You may want to create a script on the server which will create the necessary data and input file for use by Text2PDF. In this case, use the command `-exec` to provide what script to run. You'll need to include the `-allowexec` option when you install/start the server to allow `-exec`. For example:

```
text2pdf_tcp -exec myscript.sh "abc" 123
  -text2pdf #file1
  t.pdf -return c:\out.pdf -clean
```

In this case, the script `myscript.sh` will be executed on the server and passed the parameters `"abc"` and `123`. The assumption is this script will create two files for input to Text2PDF. All the business logic and database connectivity can remain in one location on the server and clients will automatically get new reports whenever the server script is updated. The option `-text2pdf` is simply a separator between the script command and the options to send to Text2PDF. You may need to provide the full path name to the script as well, especially if running the Text2PDF server as a Windows service.

The script will need to print or echo the value for `#file1`. Do this by echoing `"#file1=<path-file>"` during the script execution with each file you need to send back on a separate line. These variables may be named anything you wish but they must start with a `#` to be converted. For example, have a line that reads `"@echo #file1=c:\temp\abc5125.txt"` in the script. These will be read after the script finishes and will replace the placeholder `#file1`. You may have more than one placeholder - just be sure your script echoes all of the values. The option `-clean` instructs Text2PDF to clean up or delete the temporary files off of the server.

Here are some entries from the log file. In this case, there are 5 simultaneous processes allowed at any one time. The number in parenthesis such as the (1) and (2) below are the pool ids. For example, pool id 1 is used to start a build. While this build is happening, another request comes in to build a PDF. The second request is set to run in thread 2 while thread 1 continues to build.

```
[2010-07-24 16:00:44] Creating pool of 5 entries
[2010-07-24 16:00:44] Accepting commands on port 7080
[2010-07-24 16:01:59] (1) (127.0.0.1) a.txt a2.pdf
[2010-07-24 16:02:00] (2) (127.0.0.1) b.txt b2.pdf
```

```
[2010-07-24 16:02:03] (2) (127.0.0.1) finished build
[2010-07-24 16:02:03] (1) (127.0.0.1) finished build
[2010-07-24 16:02:11] (1) (127.0.0.1) -quit
```

The following are the options to use when setting up Text2PDF to run as a server. Remember to also include your key name/code combination using `-kn` and `-kc` or your software subscription information with `-licname`, `-licpwd`, and `-licweb`.

<u>Option</u>	<u>Description</u>
<code>-server</code>	Used to specify server processing mode. Must be the first option passed.
<code>-host <i>hostname</i></code>	The host name of the computer. The default is localhost.
<code>-port <i>number</i></code>	The port number to use. The default is 7080. You may want to setup a descriptive name in <code>etc/services</code> to use instead. For example: <code>text2pdf 7080/tcp</code> Then, use <code>-port text2pdf</code> . By adding this entry in the <code>services</code> files on your clients, you can connect in the same manner by using <code>-port text2pdf</code> . The server will not start if the port is already in use.
<code>-pool <i>number</i></code>	Optional. Pass the number of simultaneous builds to allow at a time. You should start with 5 and increase if you find users are waiting on connections. The log file will show the pool id number used for each build. If you see the maximum number of pool entries being used most of the time then you may want to increase. Keep in mind more processor time will be needed to handle more simultaneous requests so you'll need to balance the two.
<code>-log <i>path-file</i></code>	Optional. The path and name of a file to log requests to.
<code>-logmax <i>number</i></code>	Optional. The maximum size in bytes for a logfile. Once the file reaches the specified size it is renamed with the current date/time appended to the end and a new log file is started.
<code>-allowexec</code>	Optional. Set this if you want to allow clients to use the <code>-exec</code> option to run scripts.
<code>-allowsafe <i>text</i></code>	Optional. Set this if you want to allow clients to only run certain scripts. Pass in the starting characters for what the script(s) will be named. For example, to only allow people to run scripts that start with the text "t2pscript" then use that text with this option. This prevents people from passing in commands such as "rm * -r", for example. In addition, any <code>&amp;</code> or <code> </code> characters are removed before processing the command. You can separate different starting names with a comma such as "rpt,t2prpt".



<u>Option</u>	<u>Description</u>
-v	Optional. Echoes requests to the screen. Not used when running Text2PDF Service.

The client programs `tex2pdf_tcp` and `text2pdf_gui_tcp` have the same options as Text2PDF. There are a few additional options you may use shown in the following table. The actual location of the Text2PDF server doesn't matter when using the .NET/COM DLL in Windows. That is, the PDF Meld server itself may reside on a Linux box but you can use the DLL under Windows to call the server. The DLL included the with the executable downloads is `text2pdf_20.dll` and the object to create in your code is `FyTek.Text2PDF`.

<u>Option</u>	<u>Description</u>
-host <i>host</i>	The host Text2PDF server is running on.
-port <i>number</i>	The port number Text2PDF server is listening on. The default is 7080. You may want to setup a descriptive name in <code>etc/services</code> to use instead. For example: <code>text2pdf 7080/tcp</code> Then, use <code>-port text2pdf</code> . By adding this entry in the services files on your clients, you can connect in the same manner by using <code>-port text2pdf</code> .
-clopen	Client open. Opens the output PDF file in reader on the local machine. You may use <code>-open</code> if you are running off of the same box Text2PDF server is running on.
-exec <i>script [options]</i>	Pass in the name of the script the server should execute along with any options. This must be the first option passed when using the executable. To call a script called <code>myscript.sh</code> and pass it options <code>"abc"</code> and <code>123</code> you would send <code>'-exec myscript.sh "abc" 123 -text2pdf [options]'</code> . Anything after <code>-text2pdf</code> is what will be passed to Text2PDF for processing.
-text2pdf	Use this option after the <code>-exec</code> command to let the program know you are finished with the options for <code>-exec</code> and what follows is for Text2PDF to process.
-clprint	Client print. Prints the PDF to the default printer on the local machine. Using <code>-print</code> will print the PDF to the default printer from the machine Text2PDF server is running on.

<b><u>Option</u></b>	<b><u>Description</u></b>
-curdir	Sets the working directory for the server or service to be the current directory. That way your file pathing can be relative to the directory you are currently in and not from where the server or service is running from. This will likely only work if you are running off the same machine the server is running on.
-send <i>name=path-file</i>	Used to send files to the machine Text2PDF server is running on. Set the name to an input file name and path-file to the path and file name you wish to use for that file. For example, "text2pdf_tcp a.txt t.pdf -return t.pdf -send a.txt=c:\temp\afile.txt". The file a.pdf will be taken as c:\temp\afile.txt and sent to the server for processing. For the DLL, you may call this method multiple times if you have more than one file to send.
-sendcache <i>path-file</i>	Used to send a file once to the machine PDF Meld server is running on. Once cached, the server will read from a copy it has kept for itself rather than ask for the file to be transmitted again. Include this option with the same file on each build that you wish to use it. The server cache is cleared once the server program is restarted.
-autosend	Used to send all the input files without using -send or setSend for each one. You only need to set this option once. When the server program looks for a file and this option was used it will send a request back to the client requesting the file. The assumption on the server is none of the files being processed are local files.
-return <i>filename</i>	Used to return to your local machine the output PDF from Text2PDF. The filename specified should be a local file to save the PDF under.  You may use a value of "1" with the DLL to return the binary PDF to the method "runPDF". A value of "2" will return the PDF base64 encoded and a value of "3" will return it in hex format. You may then save the PDF or stream it to a browser in a web application.

<u>Option</u>	<u>Description</u>
-serverstat	<p>Returns a report of the server status. The report contains the following information:</p> <pre>Current date time      : 2010-08-01 15:00:00 Server started        : 2010-08-01 12:00:00 Requests received     : 75 Bytes received        : 4090 Bytes sent            : 0 Pool size             : 5 Available pool threads : 5 Highest pool thread use: 2 Requests that waited  : 0</pre> <p>"Server started" = the date and time the server was started. "Requests received" = the total number of requests the server has received to process. "Bytes received" = the total number of bytes sent into the server for requests. "Bytes sent" = the total number of bytes in returned PDFs sent back to clients. "Pool size" = the total number of pool entries the server was started with. "Available pool threads" = the current number of available threads. "Highest pool thread use" = the most threads that were in use at any one time. "Requests that waited" = the total number of requests that have had to wait for an entry in the pool to become available in order to run.</p>
-wait	<p>Set this option to cause the non-GUI version of text2pdf_tcp to wait until Text2PDF has finished processing before returning. Normally, when you are not receiving back the resulting PDF, text2pdf_tcp will simply send the request to the TCP/IP port and not wait for Text2PDF to perform its processing. This option causes the program to wait until finished so you know the PDF has been built and you can take some further action with it.</p>

## Text2PDF Service

The Text2PDF service is another option when running Text2PDF as a server under Windows. See the [Client-Server](#) section as the details on the various parameters are covered there. The difference with running as a service is the server program is available to anyone with network access to the server. Plus you don't need to manually start up Text2PDF in server mode each time you log in. The service can be set to start whenever the machine is booted so it can be made available without logging in first.

The program `text2pdf_srv.exe` (or `text2pdf_srv64.exe` for 64-bit) is the program for the service. You pass in `-install` as the first option (rather than `-server` like when running `text2pdf.exe`) followed by the normal options (such as `-pool` or `-host`) that you would use to start in server mode. You'll likely need administrative privileges in order to initially setup the service. Select the "Run as Administrator" option for the DOS box when you go to install.

You'll need to allow TCP traffic on the port if you want to make the service available to other computers. Go into your Windows firewall program and create an entry to allow traffic on that port. You can restrict access by computer and/or user if you like.

Note you still need to pass in a key name/code combination using `-kn` and `-kc` or your software subscription information with `-licname`, `-licpwd`, and `-licweb`.

For example:

```
C:\>text2pdf_srv.exe -install auto -pool 5
-log "c:\logs\t2plog.txt" -host "mymachine"
-port 7080 -licname "fytek-inc" -licpwd "abc12345"
-licweb
```

Replace "mymachine" in `-host` with the actual name of your computer or leave out `-host` to use the default of localhost. This should start up the service and you will then be ready to start servicing requests. Other options you can use are:

```
c:\>net start Text2PDFSrv
```

This will start the service if `-install` is used without the "auto" option. For example, you can run `text2pdf_srv.exe -install` to simply install the service without starting it.

To stop the service run:

```
c:\>net stop Text2PDFSrv
```

## Text2PDF Service

This will stop the service.

To remove the service run:

```
C:\>text2pdf_srv.exe -remove -service Text2PDFSrv
```

This will remove or un-install the service.

Be sure to fully qualify your file names as the service is not running out of the directory you are running the program `text2pdf_tcp` from. You can also use the `-cwd` or `-currdir` options to change the working directory. The [Client-Server](#) section also discusses how you can send files from a remote machine to the server running the service. You can use the `-send` and/or `-return` options with `text2pdf_tcp` in order to send and receive your files to and from the server.

The options for startup are the same as those found in the [Client-Server](#) section. The following are additional options for the service.

<b><u>Option</u></b>	<b><u>Description</u></b>
<code>-username <i>username</i></code>	Optional. The username to run the service as.
<code>-password <i>password</i></code>	Optional. The password for the username. You may leave this option off and, if <code>-username</code> is passed, the program will prompt for the password.

## 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:

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

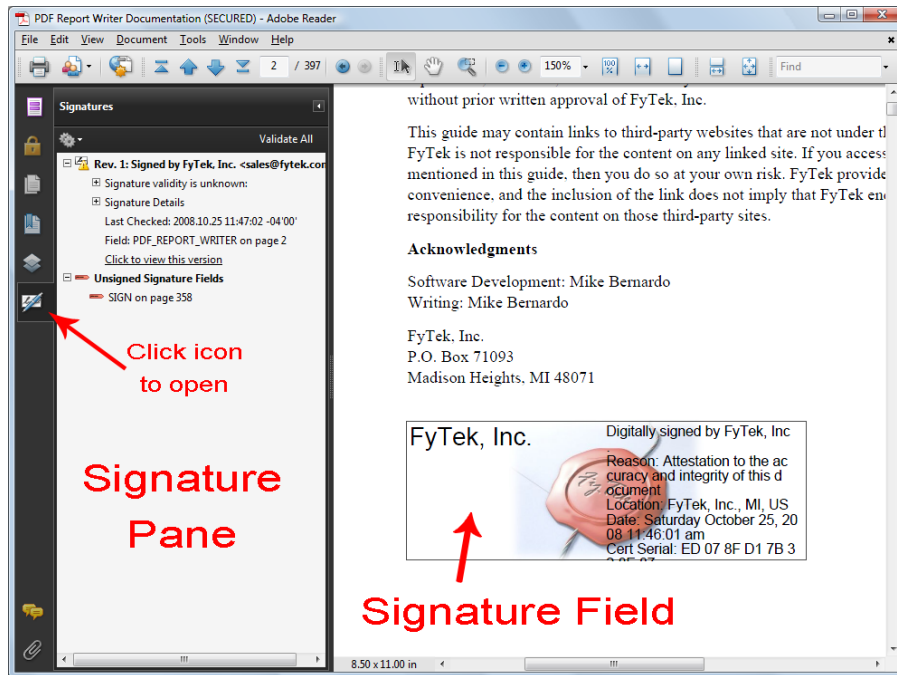
Now that you have the files, use them on the command line or the Sign method in PDF Forms. On the command line, -signpkfile would be set to mykey\_pk.pem and -signderfile would be mykey.der using the example above. You do not have to pass the signing password to the program. If you leave it out you'll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

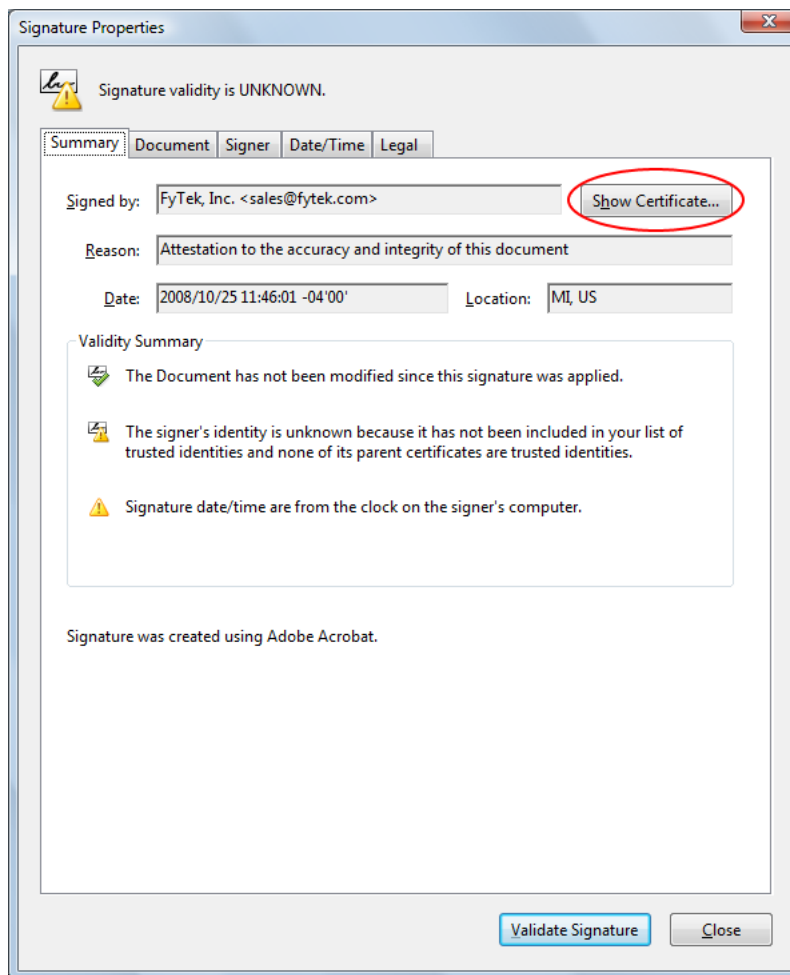
You will see something similar to the following after you sign a document for the first time.



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.

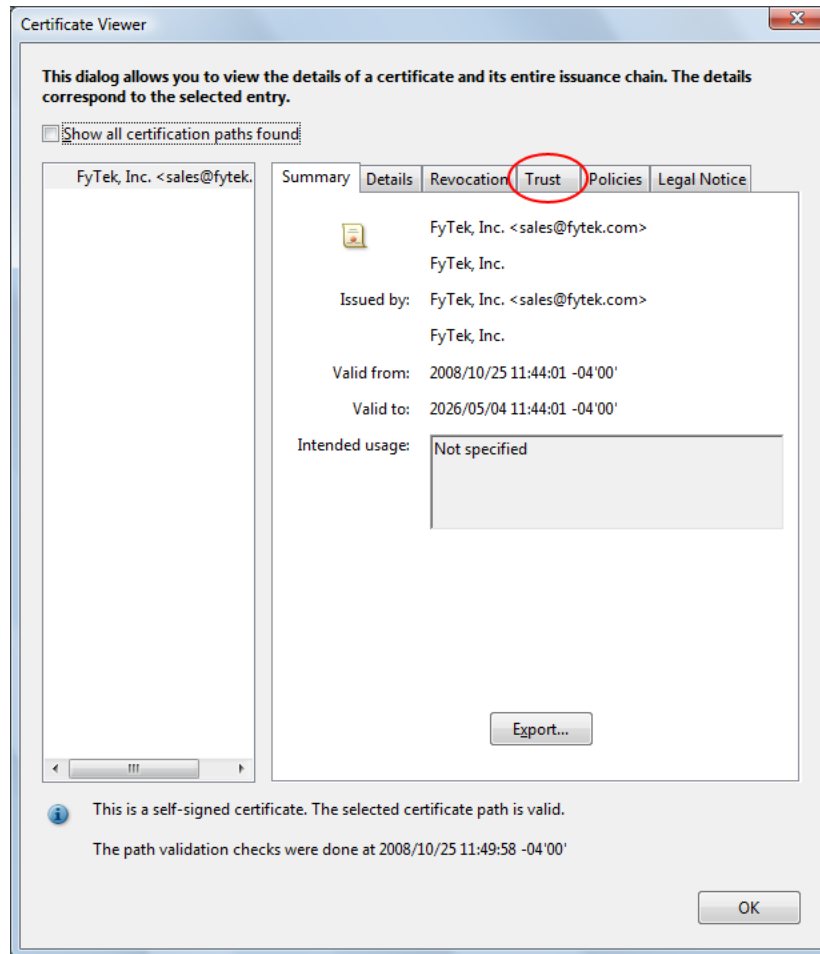


This is the dialog box that appears once you click the signature field. Click the "Signature Properties..." button to continue.

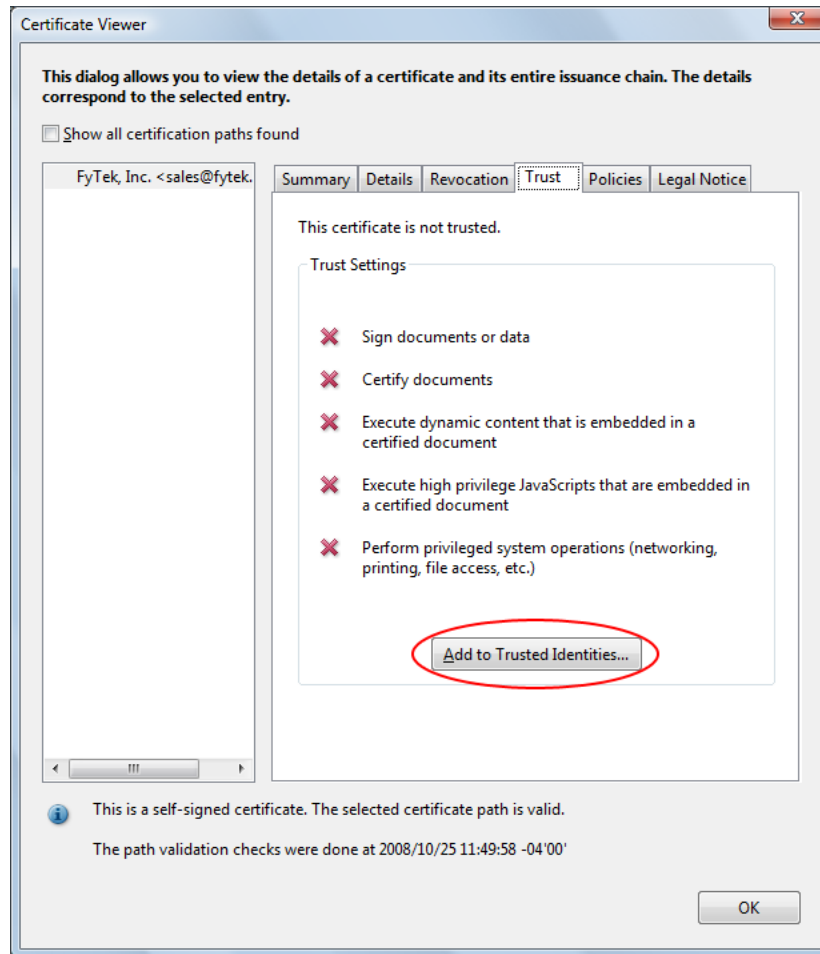


This is the signature property dialog for the certificate. Across the top of the dialog area is a set of tabs you can click on to view various information. For now, click the "Show Certificate..." button to continue.

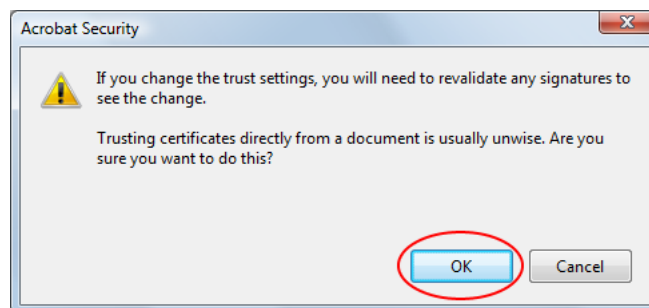
## Digital Signatures



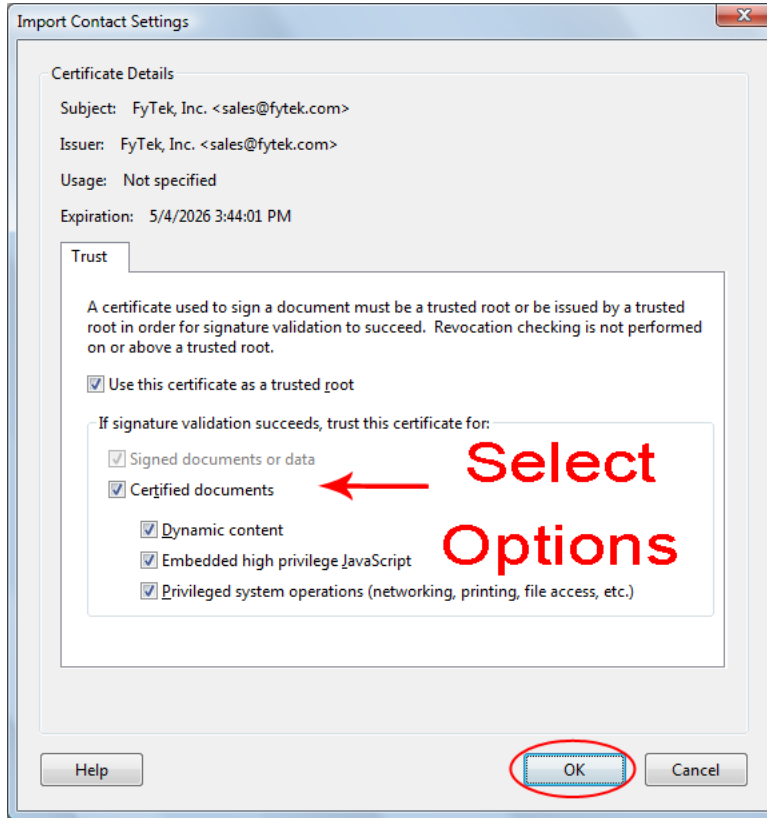
Another dialog box will open containing a set of tabs across the top. Click on the "Trust" tab.



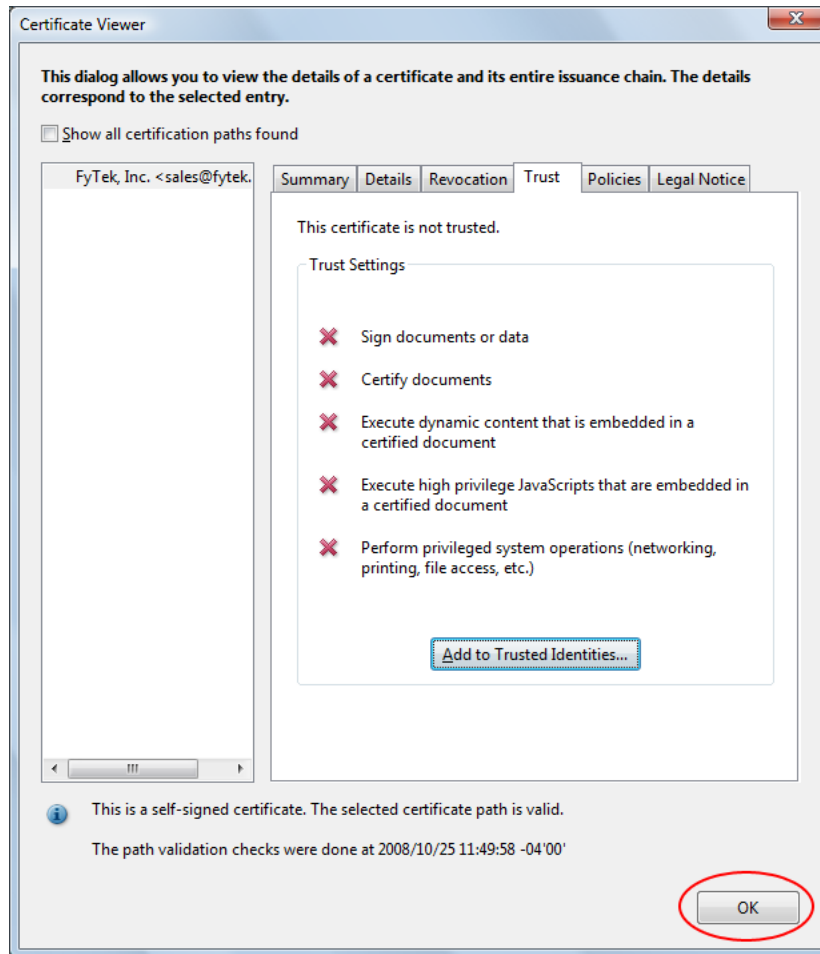
The trust tab shows what trusts you have enabled for the certificate. In this case, no trusts have been established. To trust this certificate, click the "Add to Trusted Identities..." button.



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

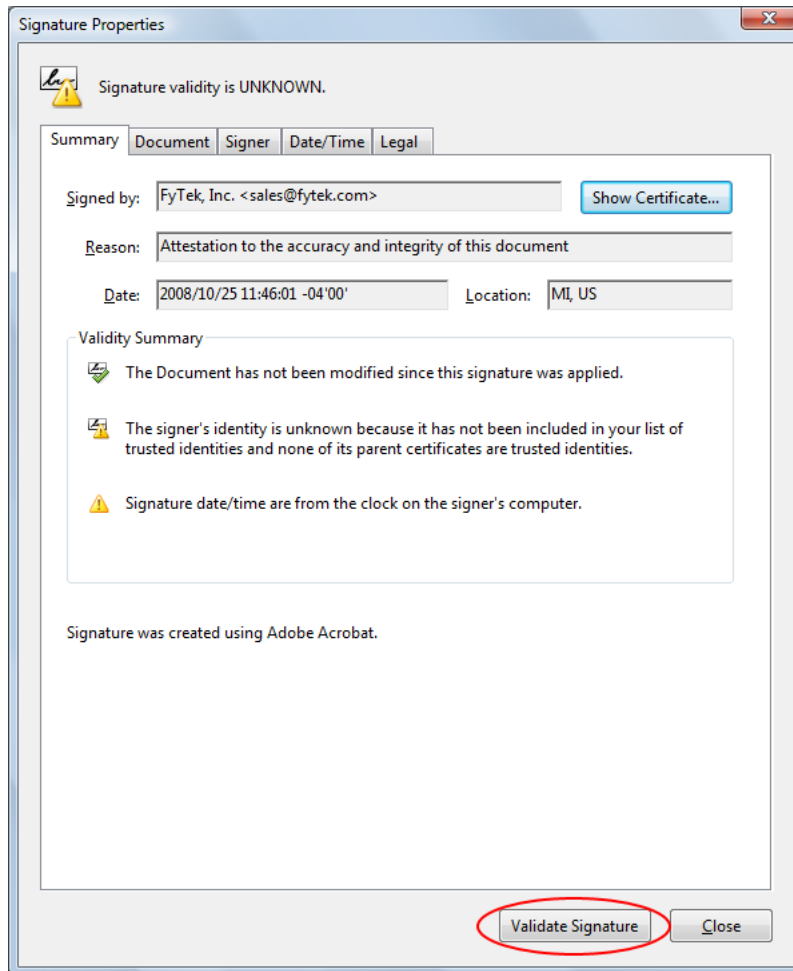


Select what items you want to trust the certificate for by clicking the checkboxes.



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

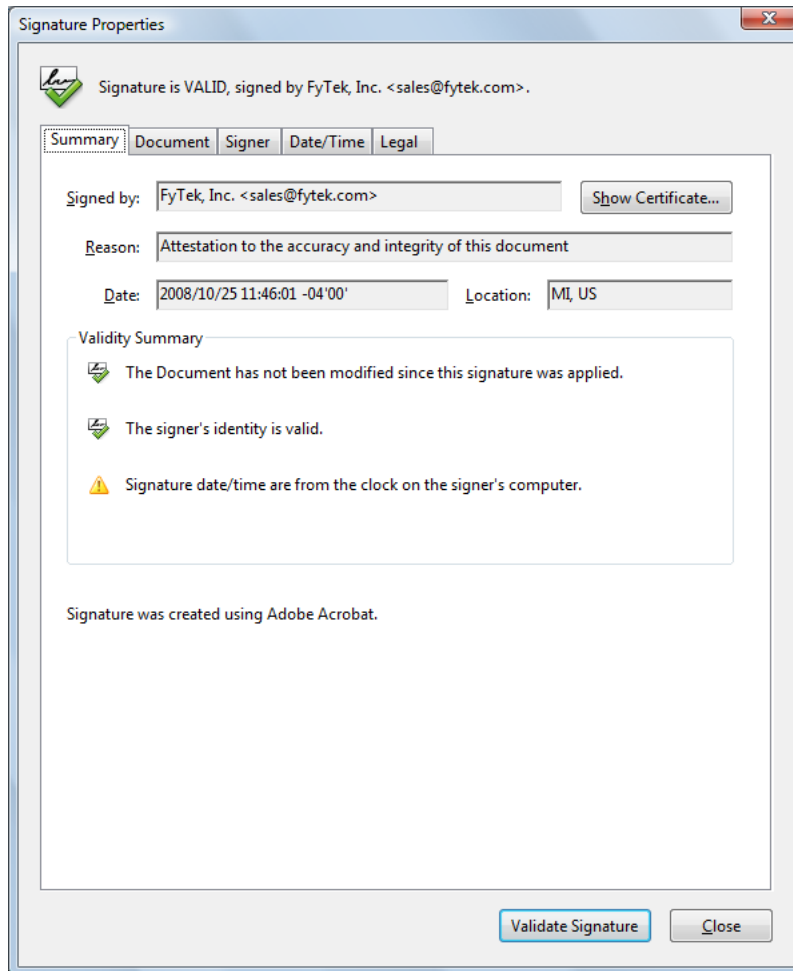
## Digital Signatures



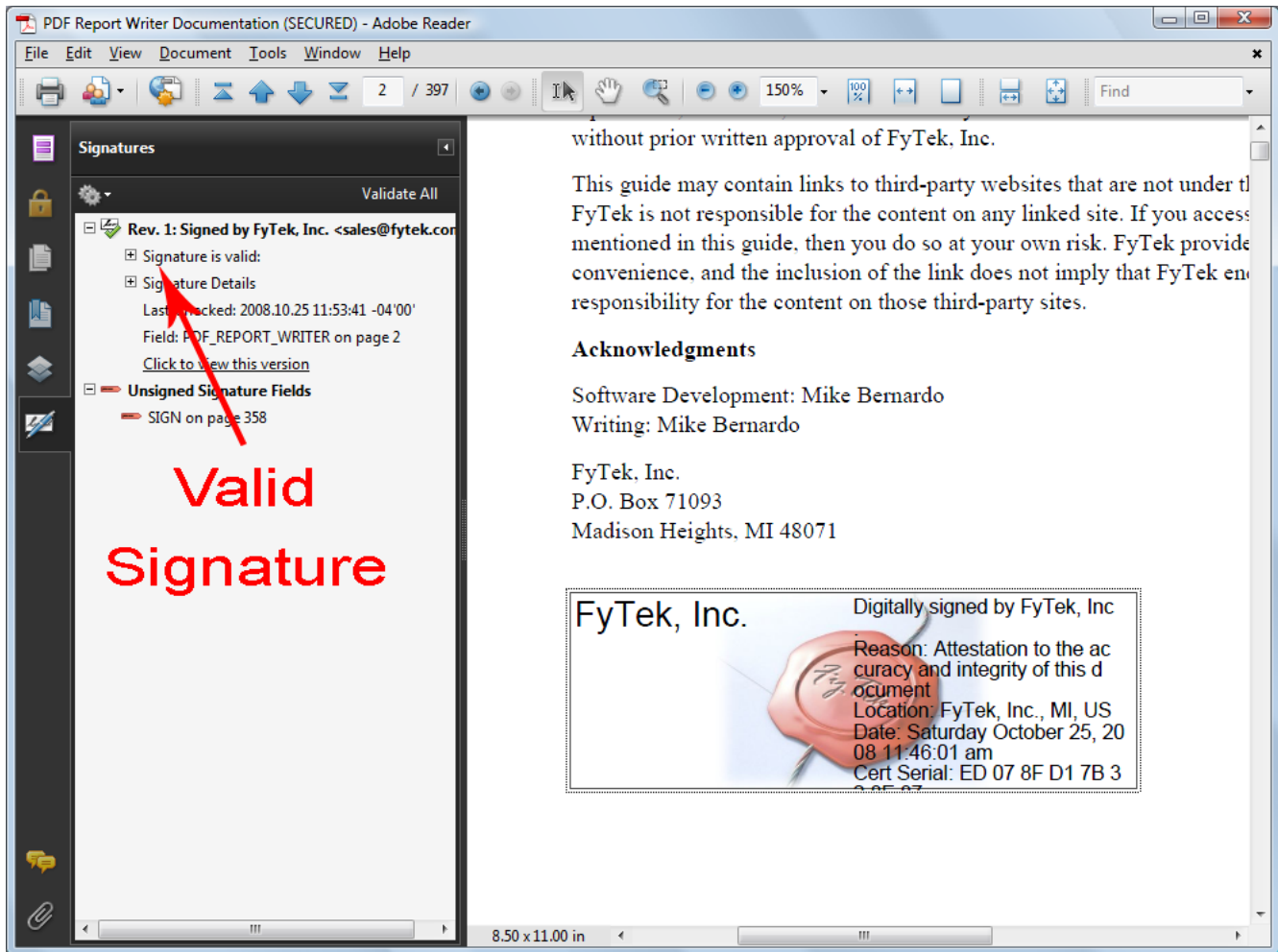
Click the "Validate Signatures" button to validate the signature we just setup the trusts for.



## Digital Signatures

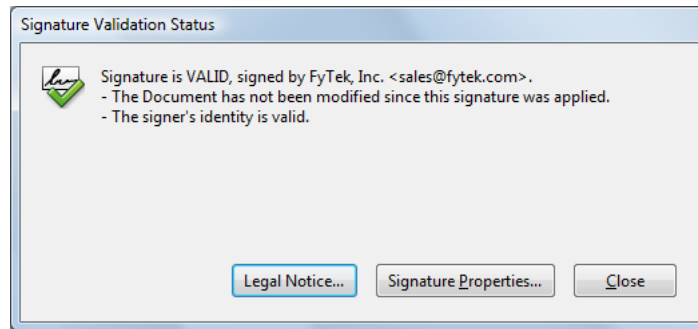


A green check icon now shows in the signature properties dialog.



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 stands for XML Paper Specification and 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.

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 XPSFile method. You may use the -xpsback option or XPSBack method to place a background on your XPS output. This is similar to the -pdf or BkgPDF 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.

## Index of Commands

[A](#)  
[ALIGN](#)  
[AUTHOR](#)

[B](#)  
[BACKGROUND](#)  
[BGCOLOR](#)  
[BOOKMARK](#)  
[BR](#)

[CIRCLE](#)  
[CLEARTABS](#)  
[COLONE](#)  
[COMP](#)  
[CREATOR](#)

[DIV](#)

[FONT](#)

[HTML](#)  
[HR](#)

[I](#)  
[IMG](#)  
[INSERTPDF](#)

[KEYWORDS](#)

[LINE](#)  
[LINESPACE](#)

[MARGINS](#)

[P](#)  
[PAGE](#)  
[PAGENUM](#)  
[PDF](#)  
[PRE](#)

[REPLACE](#)  
[RECT](#)

[SHADING](#)  
[SUB](#)  
[SUBJECT](#)  
[SUP](#)

[TAB](#)  
[TABSTOP](#)  
[TITLE](#)

[U](#)  
[UNITS](#)

[Variables](#)

[X](#)  
[XPS](#)

[Y](#)

[ZOOM](#)