



# Documentation

◆ FyTek Incorporated ◆

web: <http://www.fytek.com>

e-mail: [sales@fytek.com](mailto:sales@fytek.com)

## Executable Version

The PDF File Creator program was designed to help create PDF files quickly and easily. This document was created entirely using the PDF File Creator. What you see here should give you a good idea of what you can do with the program.

## Text Files

You can convert plain text files without modifying the text file. Of course, you won't have as much control over the file as you would using the commands described in the next section. All text will be wrapped based on information provided to the `-f` parameter described next. If you wish to keep the formatting exactly as it is on the file, use the `-nw` (nowrap) option as well.

The `-f` parameter is used to tell the program to convert a plain text file. The following may be entered after the `-f` parameter. Any entries not entered will use the defaults. Separate all entries with a comma.

<u>Field</u>	<u>Default</u>
Page Width	8.5
Page Length	11
<a href="#">Font</a> Number	4
Font Point Size	10
<a href="#">Alignment</a>	L
Left Margin (in inches)	.5
Top Margin (in inches)	.5
Right edge of text (in inches)	8
Bottom edge of text (in inches)	10
Line spacing (in 1/72 of an inch)	2
Number of columns	1
Space between columns (in inches)	1
<a href="#">Compress percent</a>	100

To create a PDF using the defaults, execute the following:

```
pdffile.exe myfile.txt myfile.pdf -f
```

Use the `-p` parameter to place page numbers at the bottom of each page. They will print a half-inch from the bottom so be sure your margins are set appropriately.

Use the `-b` parameter as well to convert any existing line feeds to `<BR>` commands. To create a file with an 11 point Helvetica font using 2 columns, justified, with .25 inches between the columns, run this command:

```
pdffile.exe myfile.txt myfile.pdf -f 8.5,11,2,11,J,,,,,2,.25
```

You can also place [in-line](#) commands in the file if you wish. You can change font, point size and color.

Another option is the nowrap option. Specify using -nw on the command line. This option will read the file without attempting to do any line wrapping. This feature is useful if you want to convert a report that would normally print on a dot matrix printer and you wish to keep all of the spacing. To convert a report using landscape you would run the following: `pdffile.exe myfile.txt myfile.pdf -f 11,8.5 -nw`

You can specify other fonts but the courier default will work best since it is non-proportional.

## Command Files

You will be able to create lines, circles and more by using a command file instead of a plain text file. You will also gain much greater control over where text prints.

Start by entering a few commands in a file to create a PDF. To create a simple PDF that displays "Hello, world", enter the following (be sure there is no space between the command and the = sign) in a file using your favorite text editor:

```
PAGE=8.5,11>>  
BOX=0,0,8.5,11,2,2>>  
ALIGN=C FONT=2,20>>  
X=1 Y=1>>Hello, world
```

Next, run the program.

```
pdffile.exe myfile.txt myfile.pdf
```

That's it. You should have a file called myfile.pdf which can be viewed with Acrobat Reader. Here is what the commands mean.

The PAGE= command is used to define a new page and give the size of the page in inches. In this case, 8.5 inches by 11 inches.

The BOX= command is used to define a virtual grid anywhere on the page. In this case, the entire page was used (from physical position 0,0 - the upper left corner for a distance of 8.5 inches along the X axis and 11 inches down the Y axis). The last two parameters of the BOX command specify the number of logical units in the X and Y direction there are respectively. So, in this example, there are three total positions in the X and Y direction (0, 1 and 2). X=0 is the left margin, X=1 is the center of the page at 4.25 inches

and X=2 is the right margin.  
Any text after the >> is printed.

The ALIGN=C command specifies center alignment of text. FONT=2,20 sets a Helvetica font at a size of 20 points. X=1 Y=1 set the position and everything after the >> is sent as text to display.

Once the concept of the page size and virtual grid placement make sense, it is easy to place anything you want precisely on the page.

Remember that all placement of text, lines, rectangles, circles, etc. are based on the virtual grid you specify.

## DLL Version

The PDF File Creator program was designed to help create PDF files quickly and easily. This document was created entirely using the PDF File Creator. What you see here should give you a good idea of what you can do with the program.

### Register the DLL

You first must register the DLL on your system. Do this by running

```
regsvr32 pdffile.dll
```

You should see a message box that reads:

DllRegisterServer in pdffile.dll succeeded.

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

In your project, go to the Project:References dialog and add the reference to buildPDF.

The methods of build.PDF are:

### setInFile (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 setPDFCmd).

### setOutFile (path-file)

Full path and name of the output file. You may leave this blank and have the output stream returned to your program. The PDF stream is returned on the call to buildPDF.

### setOpen

Opens Acrobat with the newly created PDF.

### setPrint

Prints the newly created PDF to the default printer.

### setPrintDlg

Brings up the Acrobat print dialog box and allows printer selection.

### setPrinter printer [, device, port]

Used to print the PDF to the specified printer. 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:

```
setPrinter "Accounting Printer", "HP LaserJet 5", "lpt1:"
```

or

```
setPrinter "Shipping Printer"
```

**setCopies (number)**

Number of copies to print when using the setPrint or setPrinter methods. May not work on all systems.

**setOwner (password)**

Sets the owner password for the PDF. If not specified but the user password is, this is set to the user password. Also, when not specified, the owner has only the rights granted when the document was created. So for example, if setNoPrint was specified, then it is impossible for the owner to print the document.

**setUser (password)**

Sets the user password for the PDF. The following four options can be used to turn off various features for the user.

**setNoPrint**

Disables the print command from the menu so the user cannot print the document.

**setNoPrint**

Disables the print command from the menu so the user cannot print the document.

**setNoChange**

Does not allow the user to change the document (using the full version of Adobe Acrobat).

**setNoCopy**

Does not allow the user to highlight text or graphics for cutting/pasting.

**setNoAnnote**

Does not allow the user to add/modify annotations (using the full version of Adobe Acrobat).

**setEncrypt128**

Specifies 128-bit encryption should be used (only users of Acrobat or Acrobat Reader 5.0 and up will be able to open documents using 128-bit encryption).

**setPDFCmd (command)**

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.

Leave the input file blank and send commands to setPDFCmd if you are creating them on the fly and just want to pass them to the program.

## **buildPDF**

Call this method to build the PDF. This returns the name of the output file, if set, otherwise returns the PDF stream. Be sure to remove any default header, if applicable, and send out Content-type: application/pdf followed by two line feeds if you are sending the output to a browser over the web. You don't need to do that if you are building the PDF on disk and then redirecting to that file.

The following methods are meant to be called before the call to buildPDF. They may be helpful in determining if a text string will fit in the area you are planning to place it in.

## **getStringWidth (b1, b2, b3, b4, b5, b6, f, p, c, "str")**

Call this method to return the length of a string in X units based on the box setting passed in. See the [BOX](#) command for information on the values used. The variables passed in are:

- b1...b6** - The 6 box values to use
- f** - The font number (1-14 only)
- p** - The point size
- c** - Text compression percentage (100 is default)
- "str"** - String to get the length for

## **getStringHeight (b1, b2, b3, b4, b5, b6, p)**

Call this method to return the height of a string in Y units based on the box setting passed in. It's not necessary to pass in a string as the height is based off of the point size. The variables passed in are:

- b1...b6** - The 6 box values to use
- p** - The point size

If you converting flat files (as opposed to command files discussed next) the following may be used as well. Do not use these for command files or you will simply get the text of your commands converted rather than the execution of commands.

<b><u>Method</u></b>	<b><u>Field</u></b>	<b><u>Default</u></b>
setPageWidth	Page Width	8.5
setPageLength	Page length	11
setFontNum	<a href="#">Font</a> number	4
setPointSize	Font point size	10
setAlignment	<a href="#">Alignment</a>	L
setLeftMargin	Left Margin (in inches)	.5
setTopMargin	Top Margin (in inches)	.5
setRightEdge	Right edge of text (in inches)	8
setBottomEdge	Bottom edge of text (in inches)	10
setLineSpacing	Line spacing (in 1/72 of an inch)	2
setColumns	Number of columns	1
setColSpacing	Space between columns (in inches)	1
setCompress	<a href="#">Compress percent</a>	100
setPageNum	Print page numbers	Off
setKeepBreaks	Keep line breaks	Off
setNoWrap	Do not reformat lines	Off

## Command Files

You will be able to create lines, circles and more by using a command file instead of a plain text file. You will also gain much greater control over where text prints.

Load up the sample VB program called `sampledll.vbp`. The sample program takes commands from the text editor, passes them to the PDF converter and displays a PDF. You can modify the behavior in the sample code but it should give a good idea of how it works. You load in one of the sample reports included or start by entering a few commands in the editor. To create a simple PDF that displays "Hello, world", enter the following (be sure there is no space between the command and the = sign):

```
PAGE=8.5,11>>
BOX=0,0,8.5,11,2,2>>
ALIGN=C FONT=2,20>>
X=1 Y=1>>Hello, world
```

Next, click the create button.



That's it. You should see your PDF displayed in a separate window. Here is what the commands mean.

The PAGE= command is used to define a new page and give the size of the page in inches. In this case, 8.5 inches by 11 inches.

The BOX= command is used to define a virtual grid anywhere on the page. In this case, the entire page was used (from physical position 0,0 - the upper left corner for a distance of 8.5 inches along the X axis and 11 inches down the Y axis). The last two parameters of the BOX command specify the number of logical units in the X and Y direction there are respectively. So, in this example, there are three total positions in the X and Y direction (0, 1 and 2).

X=0 is the left margin, X=1 is the center of the page at 4.25 inches and X=2 is the right margin. Any text after the >> is printed.

The ALIGN=C command specifies center alignment of text. FONT=2,20 sets a Helvetica font at a size of 20 points. X=1 Y=1 set the position and everything after the >> is sent as text to display.

Here is a small example of using the DLL from within VB. For a more complete example, have a look at the source code included for the previewer application.

```
Dim pdfCr
Dim outPdf As String
Set pdfCr = CreateObject("build.PDF")
pdfCr.setOutFile "c:\temp\hello.pdf"
pdfCr.setPDFCmd ("PAGE=8.5,11>>")
pdfCr.setPDFCmd ("BOX=0,0,8.5,11,80,50>>")
pdfCr.setPDFCmd ("FONT=2,12 ALIGN=C>>")
pdfCr.setPDFCmd ("X=40 Y=10>>Hello, world")
pdfCr.buildPDF
```

Here is an example of using the DLL from within PowerBuilder.

```
OLEObject pdfCr
pdfCr = CREATE OLEObject
li_rc = pdfCr.ConnectToNewObject("build.PDF")
ls_outfile = "c:\temp\hello.pdf"
pdfCr.setOutFile(ls_outfile)
pdfCr.setPDFCmd ("PAGE=8.5,11>>")
pdfCr.setPDFCmd ("BOX=0,0,8.5,11,80,50>>")
pdfCr.setPDFCmd ("FONT=2,12 ALIGN=C>>")
pdfCr.setPDFCmd ("X=40 Y=10>>Hello, world")
pdfCr.buildPDF
```

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

```
<%  
Dim pdfCr  
Dim binaryData  
Set obj = Server.CreateObject("fytek.unicode")  
Set pdfCr = Server.CreateObject("build.PDF")  
pdfCr.setPDFCmd ("PAGE=8.5,11>>")  
pdfCr.setPDFCmd ("BOX=0,0,8.5,11,80,50>>")  
pdfCr.setPDFCmd ("FONT=2,12 ALIGN=C>>")  
pdfCr.setPDFCmd ("X=40 Y=10>>Hello, world")  
pdfOut = pdfCr.buildPDF  
binaryData = obj.StrToByte(pdfOut)  
Response.ContentType = "application/pdf"  
Response.binarywrite binaryData  
set pdfCr = nothing  
set pdfOut = nothing  
set binaryData = nothing  
set obj = nothing  
%>
```

Here is another example of using the DLL with ASP. This option creates the PDF on disk then redirects the user.

```
<%  
Dim pdfCr, RndFile  
Set pdfCr = Server.CreateObject("build.PDF")  
Randomize  
RndFile = "output\" & Int(10000000 * Rnd + 1) & ".pdf"  
pdfCr.setOutFile (RndFile)  
pdfCr.setPDFCmd ("PAGE=8.5,11>>")  
pdfCr.setPDFCmd ("BOX=0,0,8.5,11,80,50>>")  
pdfCr.setPDFCmd ("FONT=2,12 ALIGN=C>>")  
pdfCr.setPDFCmd ("X=40 Y=10>>Hello, world")  
pdfOut = pdfCr.buildPDF  
Response.redirect(RndFile)  
%>
```

Here is an example using C.

```
#include <iostream.h>

// The import directive reads the typelib information from the DLL
// and creates pdffile.tlh and pdffile.tli, which are included.
// These define wrappers for each of the pdffile object methods.

#import <pdffile.dll>

// Using VC++ 5.0 Smart Pointers makes this much easier.
// The parameter string for a method is converted to Unicode, allocated
// and passed as a variant. The wrappers call IDispatch::Invoke
// This is all compatible with MFC (use AfxOleInit instead of CoInitialize, etc.).

int main(int argc, char* argv[])
{
    HRESULT      hr;

using namespace buildPDFs_TypeLib;

    hr = CoInitialize (NULL);    // Initialize COM
    if (SUCCEEDED(hr))
    {
        try    // Each of the following lines can throw exceptions
        {
            // Create the instance and get a pointer to the interface
            IbuildPDFsPtr pPDF(__uuidof(FileCreator));

            pPDF->setOutFile (_bstr_t(L"c:\\TestPDF\\new.pdf")); // Set the output file
            pPDF->setPDFCmd (_bstr_t(L"PAGE=8.5,11>>")); // Could do a setInFile instead
            pPDF->setPDFCmd (_bstr_t(L"BOX=0,0,8.5,11,80,50>>"));
            pPDF->setPDFCmd (_bstr_t(L"FONT=2,12 ALIGN=C>>"));
            pPDF->setPDFCmd (_bstr_t(L"X=40 Y=10>>Hello, world"));

            _variant_t outval = pPDF->buildPDF (); // Build the PDF file
        }
        catch (_com_error e)
        {
            cout << e.ErrorMessage() << endl;
        }
    }
    else
        cout << "CoInitialize Failed" << endl;

    CoUninitialize();    // Uninitialize COM

    return 0;
}
```

You can also string your commands together and separate them with a carriage-return line-feed (CHR(13) & CHR(10) in VB). Then just call setPDFCmd once.

Remember that all placement of text, lines, rectangles, circles, etc. are based on the virtual grid you specify.

## Encryption

You can encrypt your documents so only people you give the user or owner password to can open the document. The owner password allows access for printing, copying text, etc. The user password allows only the rights granted that were specified for the user when the document was created. If the owner password is not specified but the user password is, then the owner password is set to the user password. Also, when the owner password is not specified, the owner has only user rights. So for example, if -noprnt was specified, then it is impossible for the owner to print the document.

You specify the owner and/or user passwords with the -o and -u paramters like this:

```
pdffile.exe myfile.rpt myfile.pdf -o abc123 -u xyz
```

This sets two passwords and both the owner and user have all access rights. You can restrict four areas:

-noprnt (No print) - disables the print command from the menu so the user cannot print the document.

-nochange (No change) - does not allow the user to change the document (using the full version of Adobe Acrobat).

-nocopy (No copy) - does not allow the user to highlight text or graphics for cutting/pasting.

-noannotate (No annotation changes) - does not allow the user to add/modify annotations (using the full version of Adobe Acrobat).

To specify that the user cannot print or copy from the document, run the program with the noprnt and nocopy options like this:

```
pdffile.exe myfile.rpt myfile.pdf -o abc123 -u xyz -noprnt -nocopy
```

You can mix and match any set of permissions. The permissions only work if you also set a user password. Opening the document with the user password as shown above will not allow printing or copying. Opening the document with the owner password will allow those functions.

The default is a 40-bit encryption method. This method can be used by users of Acrobat or Acrobat Reader 4.05 and up. You can optionally specify a 128-bit method be used but only users of Acrobat or Acrobat Reader 5.0 and up will be able to open documents with

this method. Specify 128 bit encryption by including the option `-e128` on the command line. Using the above example with 128-bit encryption would be:

```
pdffile.exe myfile.rpt myfile.pdf -o abc123 -u xyz -e128 -noprint
```

The methods for the DLL are `setOwner (password)`, `setUser (password)`, `setNoPrint`, `setNoChange`, `setNoCopy`, and `setNoAnnote`.

## Auto Open/Print/E-Mail Attach

These options are for the command line version or when using the DLL with an installed application (i.e. not over the Web). You can use the `-open` and/or the `-print` options on the command line to automatically open or print the newly created PDF. To create a PDF and open it in Acrobat, run:

```
pdffile.exe myfile.rpt myfile.pdf -open
```

To create a PDF and print it without bringing up Acrobat you can run:

```
pdffile.exe myfile.rpt myfile.pdf -print
```

The methods for the DLL are `setOpen` and `setPrint`.

The `-printdlg` option or `setPrintDlg` method are used to bring up the printer dialog box once the PDF is created. The user may then select the printer and layout options.

The `-printer "printer" ["driver" "port"]` option or `setPrinter Printer[,Driver,Port]` method are used to specify the printer to print the PDF on. The first parameter in both the option and method is required (the printer). The driver and port are optional. There is no printer dialog box in this case. For example:

```
-printer "Accounting Printer" "HP LaserJet 5" "lpt1:"
```

or

```
-printer "Shipping Printer"
```

The `-copies n` option or `setCopies(n)` method allows you to set the number of copies to print. This relies on the registry settings and may not work on all computers. Only one copy of a given PDF may be opened at a time. In order for multiple copies to print, the software must open the PDF, print it and monitor the thread then close/re-open for the remaining copies. You'll probably want to use the `-printdlg` option or `setPrintDlg` method if setting the number of copies doesn't work.

You can use the `-mail` option to open the user's e-mail to a new composition window with the newly created PDF attached. To create a PDF and attach it to a new e-mail message you can run:

```
pdffile.exe myfile.rpt myfile.pdf -mail
```

This option may not work for all e-mail programs. The method for the DLL is `setMail`. Also see the [SENDMAIL](#) command to email the PDF without user interaction.

Commands may be in lower or uppercase. Commands start at the beginning of the line and end at >>. All commands must be entered as COMMAND=X,Y,... with no space between command, equal sign and parameters. The following is a list of commands available and their description.

**PAGE=X,Y** Used to start a new page or form. X = Page width in inches, Y = Page height in inches.

**NEWFORM=X** Used to start a new form. X = Form number. A form is a template that can be placed on a page as a background. The USEFORM command is used to tell a page to include the form or template.

**USEFORM=X** Used to insert a form. X = Form number.

**BOX=A,B,C,D,X,Y**

Used to position a virtual grid on the page. X=0 and Y=0 is the upper left corner. (See the [REV command](#) if you would rather have Y=0 be on the bottom.)  
 A = Number of inches from left edge of page to the left edge of the grid.  
 B = Number of inches from top edge of page to the top edge of the grid.  
 C = Width of grid in inches.  
 D = Height of grid in inches.  
 X = Number of equally spaced units in X direction.  
 Y = Number of equally spaced units in Y direction.

**FONT=X,Y**

Used to specify a font.  
 X = Font number.

1 = Courier (ABC 123)	4 = Courier Bold (ABC 123)
7 = Courier Italics (Sample: ABC 123)	10 = Courier Bold-Italics (Sample: ABC 123)
2 = Helvetica (Sample: ABC 123)	5 = Helvetica Bold (Sample: ABC 123)
8 = Helvetica Italics (Sample: ABC 123)	11 = Helvetica Bold-Italics (Sample: ABC 123)
3 = Times Roman (Sample: ABC 123)	6 = Times Bold (Sample: ABC 123)
9 = Times Italics (Sample: ABC 123)	12 = Times Bold-Italics (Sample: ABC 123)
13 = Symbol (Sample: ABX 123)	14 = Zapf Dingbats (Sample: ☆♣%& / / ✓)

J1 = HeiseiMin-W3-90ms-RKSJ-H (Japanese)  
 J1B = HeiseiMin-W3-90ms-RKSJ-H Bold (Japanese)  
 J1I = HeiseiMin-W3-90ms-RKSJ-H Italics (Japanese)  
 J2BI = HeiseiMin-W3-90ms-RKSJ-H Bold-Italics (Japanese)  
 J2 = HeiseiKakuGo-W5-90ms-RKSJ-H (Japanese)  
 J2B = HeiseiKakuGo-W5-90ms-RKSJ-H Bold (Japanese)  
 J2I = HeiseiKakuGo-W5-90ms-RKSJ-H Italics (Japanese)  
 J1BI = HeiseiKakuGo-W5-90ms-RKSJ-H Bold-Italics (Japanese)  
 C1 = STSong-Light (Simplified Chinese)  
 C1B = STSong-Light Bold (Simplified Chinese)  
 C1I = STSong-Light Italics (Simplified Chinese)  
 C1BI = STSong-Light Bold-Italics (Simplified Chinese)

Y = Point size.

You'll need the Japanese or Chinese font pack from Adobe to view PDFs with Asian fonts. They can be downloaded from <http://www.adobe.com/products/acrobat/acrrasianfontpack.html>

In addition, you may need to install Adobe's ATM font manager. It can be downloaded from <http://www.adobe.com/support/downloads/atmwin.htm>. After you've installed the Chinese Simplified font pack, go to the ATM type manager and add the font. The name of the file is STSongStd-Light-Acro.otf.

## **ADDFONT='font,Font Name,Font Type'**

Used to embed a True Type or Type 1 Font.

font.ttf = Name of the True Type Font file (including ttf extension)

or, for Type 1 fonts, the base font file without extension.

Font Name = The name of the font.

Font Type = TrueType for True Type (default) or Type1 for a Type 1 font.

Any fonts added are numbered starting with 15 in the order the ADDFONT command appears.

For example, use `ADDFONT='c:\windows\fonts\myfont.ttf,myfont'>>` for a True Type and `ADDFONT='c:\windows\font\mytype1,myfont,type1'>>` for a Type 1 font.

You then refer to the newly added fonts by number starting with 15 for the first font You've added. For example, use `FONT=16,12` to specify your second (#16) embedded font with a point size of 12.

*Here is a sample embedded font.*



**FCOLOR=R,G,B** Used to specify a fill color.

**SCOLOR=R,G,B** Used to specify a stroke color.

R = Red component.

G = Green component.

B = Blue component.

Each component is a number between 0 and 1 OR use 0 to 255.

OR - You can also use a hexadecimal value preceded by a # sign. For instance, #CCA59B.

OR - You can use FCOLOR= one of the following colors:

black, silver, gray, white, maroon, red, purple, fuchsia,

green, lime, olive, yellow, navy, blue, teal, aqua

**OFFSET=X,Y** Used to assign a small offset to text so it does not print flush to lines when boxes are drawn around it in the grid.

X = X offset amount in 1/72 of an inch (1 is typically sufficient).

Y = Y offset amount in 1/72 of an inch (1 is typically sufficient).

**COMP=X** Amount to compress (or expand) text font by.

X = Percentage (0-99 compresses, 100 is standard, 101+ expands)

This text is compressed using the command COMP=50

**This text is compressed using the command COMP=150**

**ROTATE=X** Used to rotate text. X = angle from 0 to 360.

**ALIGN=X** Used to specify the current text alignment. X = L (left), C (center), R (right) or J (justify).

Left Align	Centered	Right Align
------------	----------	-------------

**X=A Y=B X2=C (X2 for right column position is only used when ALIGN=J (justify) is used)**

Used to specify the current text placement. All text following the >> symbols is printed.

The A and B values for X and Y are the coordinates in the logical box (from the BOX command).

To print "Hello, world" for instance, use something like:

X=1 Y=5>>Hello, world

**REND=A** Used to specify the text drawing mode.

Values for A:

0 = Fill text (default).

1 = Stroke text (outline).

2 = Fill then stroke.

3 = No fill or stroke (invisible).

**Example of A=1 - Stroke Text**

**Example of A=2 - Fill and Stroke Text**

**TEXTBLOCK=X,Y,X1,Y1,L**

Used to specify a block of text.

X = Grid position of left margin.

Y = Grid position for top line of text.

X1 = The right margin if alignment is Justify (ignored for all others).

Y1 = Grid position for bottom line of text.

L = (optional) Line spacing to use between lines (1 = 1/72 of an inch).

All text following this command is copied as is. If any PDF commands are encountered, they will simply be printed. The ENDTXTBLOCK command signals the end of the block.

If line spacing is not specified, lines will be spaced evenly for the entire block.

This command does not perform any text wrapping. Line breaks occur where they are found in the original text.

**ENDTXTBLOCK**

Used to signal the end of the TEXTBLOCK or WRAPBLOCK command.

**WRAPBLOCK=X,Y,X1,Y1,L,C,CS,F**

Used to specify a block of text to wrap in a rectangle.

X = Grid position of left margin.

Y = Grid position for top line of text.

X1 = Grid position for right margin.

Y1 = Grid position for bottom line of text.

L = (optional) Line spacing to use between lines (1 = 1/72 of an inch, default is 2).

C = (optional) Number of columns to use (default is 1).

CS = (optional) Space between the columns based on current grid (default is 1).

F = (optional) Force text to fit in the area specified (1 or 2 - default is 0).

All text following this command will be copied as is. You may insert in-line commands such as FONT (font/size), A (links/tags) and FCOLOR (text color).

All line breaks read will be ignored. Line breaks will be added where needed. Additionally, you can use the <BR> command to force a line break. The WRAPBLOCK command will also span WRAPBLOCK commands so you can split up your text in a variety of ways.

Use the F option to force the text to fit in the area specified. By setting F to 1 Text will be compressed until it fits. Setting F to 2 will first reduce the font size (down to a minimum of 66% of the original size) then start compressing, if necessary, to force the text to fit.

**Example:**

```
PAGE=8.5,11>>
BOX=.25,.25,8,10.5,100,60>>
ALIGN=J FONT=2,11>>
WRAPBLOCK=1,6,60,18,,2,2>>
```

Here is some text for the WRAPBLOCK command.

The first part contains two columns. The last part is in one column.

Note that line breaks require the <BR> command.

Also, <FONT=3,18>font types<FONT=2,11> can be changed as well as  
<FCOLOR=0,.6,.2>font <FCOLOR=.8,.2,0>color<FCOLOR=0,0,0>.

```
<BR><BR>
```

Once the two columns defined for this command have filled, the remaining text will spill over into the next WRAPBLOCK command.

In order for this to work, you must specify the next WRAPBLOCK command followed immediately by an ENDTXTBLOCK command.

If not, any remaining text will be discarded and the new text will be loaded.

You may issue any other commands including new pages between the first ENDBLOCK and next WRAPBLOCK.

You may also have several WRAPBLOCK commands for the text to flow into.

Note also that the second block is using left justification and a different font.

Each block can be left, right, center or justified across the column.

```
ENDTXTBLOCK>>
ALIGN=L FONT=3,12>>
WRAPBLOCK=1,18,50,30>>
ENDTXTBLOCK>>
```

**Here is the output from the WRAPBLOCK example:**

Here is some text for the [WRAPBLOCK](#) command. The first part contains two columns. The last part is in one column. Note that line breaks require the <BR> command. Also, font types can be changed as well as font color.

this command have filled, the remaining text will spill over into the next **WRAPBLOCK** command. In order for this to work, you must specify the next **WRAPBLOCK** command followed immediately by an ENDTXTBLOCK command. If not, any remaining text will be discarded and the new text will be loaded. You may issue any

other commands including new pages between the first ENDBLOCK and next **WRAPBLOCK**. You may also have several **WRAPBLOCK** commands for the text to flow into. Note also that the second block is using left

Note that page breaks do not automatically occur. If you have a lot of text that you want to wrap to another page, you must start another page using the PAGE= command then issue another WRAPBLOCK command followed immediately by an ENDTXTBLOCK command.

Here are some special in-line commands you can use with the WRAPBLOCK command:

**<X=pct>**

Used to set the X position in the current line.  
 pct = Value from 0 to 100 representing the percentatge of the current line width.  
 This allows you to set column alignments while in the WRAPBLOCK command.  
 Example:  
 Col1<X=40>Col2<X=90><ALIGN=C>Col3<BR>  
 ABC<X=40>123<X=90><ALIGN=C>1<BR>  
 XYZ<X=40>555<X=90><ALIGN=C>500<BR>

<u>Col1</u>	<u>Col2</u>	<u>Col3</u>
ABC	123	1
XYZ	555	500

**<ALIGN=L|R|C|J>**

Used to change the alignment in a line.

**<FILL="str"> or <FILLLINE>**

Used to fill the space between <X=pct> commands with a character or string.  
 str = A character or string enclosed in quotes.  
 To use, pick a point somewhere between the text you want to fill with the character.  
 Next, place the fill command in front to fill the area.

Here is an example:

```
WRAPBLOCK=1,10,60,20>>
Chapter 1 (Intro)<FILL=" ."><X=60><X=95><ALIGN=R>15<BR>
Chapter 2<FILL=" ."><X=60><X=95><ALIGN=R>127<BR>
Appendix<FILLLINE><X=60><X=95><ALIGN=R>127<BR>
ENDTXTBLOCK>>
```

Chapter 1 (Intro) .....	15
Chapter 2 .....	127
Appendix .....	227

The X=60 is used as the pivot point. The software takes that point and works it way backwards to fill space to the left. Next, the software works it way forward from that point to the start of the text on the right. This keeps the dots or whatever string is used aligned from row to row. The FILLLINE command works the same way except it draws a line instead of text.

**TBL=Y1,Y2,R,X1,X2,...,X<sub>n</sub>**

Used to draw a table (gridlines) on the page.

Y1 = The grid position for the top.

Y2 = The grid position for the bottom.

R = The number of rows.

X1...X<sub>n</sub> = The grid positions of the rows (include the left and right edges).

**TBLSET=Y1,Y2,R,X1,X2,...,X<sub>n</sub>**

Used to set the points for the table but not actually draw it.

Y1 = The grid position for the top.

Y2 = The grid position for the bottom.

R = The number of rows.

X1...X<sub>n</sub> = The grid positions of the rows (include the left and right edges).

**TBLALIGN=A1,A2,...,A<sub>n</sub>**

Used to specify the column alignments.

A1...A<sub>n</sub> = L for Left, R for Right, C for Center or J for Justify.

**TBLROWALIGN=X,A1,A2,...,A<sub>n</sub>**

Used to specify the column alignments for row X.

A1...A<sub>n</sub> = L for Left, R for Right, C for Center or J for Justify.

**TBLCOLALIGN=Y,A**

Used to specify the column alignments for column Y.

X = The row in the table (X=1 is the first row).

A = L for Left, R for Right, C for Center or J for Justify.

**TBLCOLOR=R,G,B (Must issue before the TBL command)**

Used to specify the background color for the cells.

R, G and B are the Red, Green and Blue color components. Each is a decimal from 0 to 1.

**TBLROWCOLOR>=X,R,G,B (Must issue before the TBL command)**

Used to specify the background color for row X.

X = The row in the table (X=1 is the first row).

R, G and B are the Red, Green and Blue color components. Each is a decimal from 0 to 1.

**TBLCOLCOLOR>=Y,R,G,B (Must issue before the TBL command)**

Used to specify the background color for col Y.

Y = The column in the table (Y=1 is the first column).

R, G and B are the Red, Green and Blue color components. Each is a decimal from 0 to 1.

**TBLCELLCOLOR>=X,Y,R,G,B (Must issue before the TBL command)**

Used to specify the background color for cell X, Y.

X = The X cell position in the table (X=1, Y=1 is the upper left corner).

Y = The Y cell position in the table.

R, G and B are the Red, Green and Blue color components. Each is a decimal from 0 to 1.

**TBLROW=X** X = The starting row number to place data (X=1 is the first row).

**TBLDATA** Insert the text to print for each row after this command.  
Use a verticle bar '|' to separate columns of data.  
Use a <BR> for a line break.  
Example: TBLDATA>>Data for col 1|Data for col 2|Data for col 3

The following page shows an example of two of tables.  
The BOX command used for both is BOX=.25,1,8,9,20,50>>

**TBLDATA2** Works similar to TBLDATA but does not check if text will fit or needs to wrap. Using this command can increase the build speed by 2X or more over the TBLDATA command. However, this command will not attempt to wrap text or make sure it fits in the cell so you'll need to make sure you're printing only what will fit.

**Example 1 (Grid is set-up for two lines of text per row):**

```
TBLCOLOR=.9,.9,.9>>
TBLROWCOLOR=1,.8,1,.8>>
TBL=12,22,4,4,7,9,12>>
TBLROW=1>>
TBLROWALIGN=1,C,C,C>>
TBLALIGN=L,C,R>>
TBLDATA>><FONT=5,11>Company <BR>Address|Active|Sales<FONT=2,11>
TBLDATA>>ABC Corp<BR>123 Main Street|Yes|500.00
TBLDATA>>Any Company<BR>1000 2nd Street|Yes|25,500.00
TBLDATA>>XYZ Company<BR>555 South Blvd|No|1,200.00
```

Company \ Address	Active	Sales
ABC Corp 123 Main Street	Yes	500.00
Any Company 1000 2nd Street	Yes	25,500.00
XYZ Company 555 South Blvd	No	1,200.00

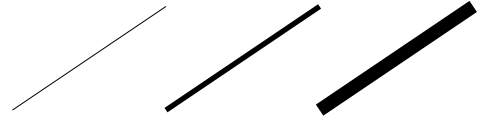
**Example 2:**

```
TBLCOLCOLOR=1,1,.8,.8>>
TBLROWCOLOR=1,1,1,.8>>
TBLCELLCOLOR=1,5,.8,1,.8>>
TBL=40,48,5,2,5,7,9,11,13,15,17>>
TBLROW=1>>
TBLCOLALIGN=1,L>>
TBLROWALIGN=1,C,C,C,C,C,C,C>>
TBLALIGN=L,R,R,R,R,R,R>>
TBLDATA>><FONT=5,11>Company|Jan|Feb|Mar|Apr|May|Jun
TBLDATA>><FONT=5,11>ABC Corp.<FONT=2,11>|200|1,250|600|500|1,000|300
TBLDATA>><FONT=5,11>Any Company<FONT=2,11>|25,000|9,000|6,500|4,500|1,800|3,000
TBLDATA>><FONT=5,11>XYZ, Inc.<FONT=2,11>|5,00|3,000|1,000|5,000|8,250|4,500
TBLDATA>><FONT=5,11>Total<FONT=2,11>|$30,200|$13,250|$8,100|$10,000|$11,050|$7,800
```

Company	Jan	Feb	Mar	Apr	May	Jun
ABC Corp.	200	1,250	600	500	1,000	300
Any Company	25,000	9,000	6,500	4,500	1,800	3,000
XYZ, Inc.	5,00	3,000	1,000	5,000	8,250	4,500
<b>Total</b>	\$30,200	\$13,250	\$8,100	\$10,000	\$11,050	\$7,800

**LINEW=X**

Used to specify the line thickness for drawing.  
X = Decimal value (.1=thin, 1=standard, 3=thick).

**LINEC=X**

Used to specify the line cap  
X Values:  
0 = Butt ends.  
1 = Round ends.  
2 = Projecting Square Caps.

**LINED=A,B**

Used to specify the dash pattern  
A = On pixel length.  
B = Off pixel length.  
Example using 5,4  
Note that the 'Off' color is the color set with the FCOLOR command.  
If both [FCOLOR](#) and [SCOLOR](#) are the same, the line won't appear as dashed.

**LINE=A,B,C,D**

Used to draw a line based on the current grid.  
A = From X grid position.  
B = From Y grid position.  
C = To X grid position.  
D = To Y grid position.

**RECT=A,B,C,D**

Used to draw a rectangle based on the current grid.  
A = From X grid position.  
B = From Y grid position.  
C = To X grid position.  
D = To Y grid position.

**RECTF=A,B,C,D**

Used to draw a rectangle based on the current grid and fill with current fill color.  
A = From X grid position.  
B = From Y grid position.  
C = To X grid position.  
D = To Y grid position.

**CIRCLE=A,B,C[I] (Like RECTF, CIRCLEF draws a filled in circle)**

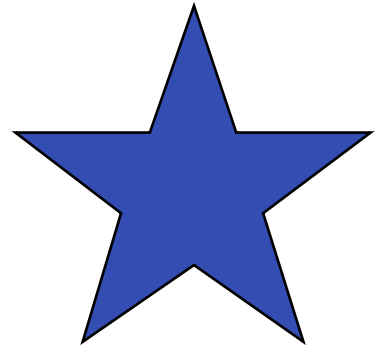
Used to draw a circle based on the current grid.  
A = X grid position.  
B = Y grid position.  
C = C size in grid position or, if followed by letter 'I', then radius in inches.



**POLY=X1,Y1,X2,Y2,...X<sub>n</sub>,Y<sub>n</sub> (like RECTF, POLYF fills in the polygon)**

Used to draw a polygon based on the current grid.

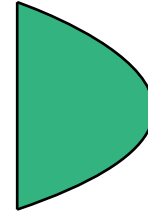
- X1 = X grid position 1.
- Y1 = Y grid position 1.
- X2 = X grid position 2.
- Y2 = Y grid position 2.
- etc...



**CURVE=X1,Y1,X2,X3,Y3 (Like RECTF, CURVEF draws a filled in curve)**

Used to draw a Bezier curve w/ 1 control point based on the current grid.

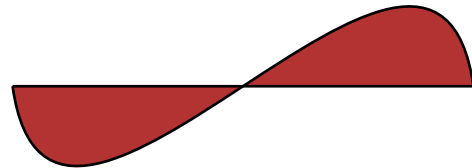
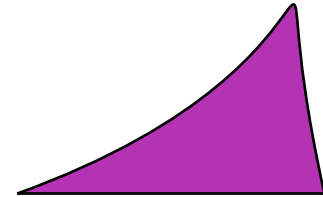
- X1 = X grid position 1.
- Y1 = Y grid position 1.
- X2 = X grid position for control point.
- Y2 = Y grid position for control point.
- X3 = X grid position 2.
- Y3 = Y grid position 2.




**CURVE=X1,Y1,X2,Y2,X3,Y3,X4,Y4 (Like RECTF, CURVEF draws a filled in curve)**

Used to draw a Bezier curve w/ 2 control points based on the current grid.

- X1 = X grid position 1.
- Y1 = Y grid position 1.
- X2 = X grid position for first control point.
- Y2 = Y grid position for first control point.
- X3 = X grid position for second control point.
- Y3 = Y grid position for second control point.
- X4 = X grid position 2.
- Y4 = Y grid position 2.



- OUTLINE='X,Y,C,X1,Y1,C1'** Used to specify the outline entry for the document.  
 X = Outline level (outlines can have suboutlines).  
 Y = Text of the outline.  
 C = Closed (optional - either put a single character C for closed or leave blank).  
 X1 = Outline level (optional - creates a suboutline for this entry).  
 Y1 = Text of the optional suboutline.  
 C1 = Closed (optional - either put a single character C for closed or leave blank).
- IMAGE='A,file[,effects]'** Used to specify an image (jpeg, gif, tif, bmp, png, etc.)  
 A = Image id number.  
 file = File name of image.  
 Effects are combination of BLUR, CHARCOAL, EMBOSS, NEGATE or OILPAINT.  
 For example, IMAGE='1,c:\images\myimage.tif,charcoal,emboss'>>  
 Do not put multiple image commands in the same command line.
- PUTIMG=A,X,Y,X1,Y1** Used to place an image (defined in the IMAGE command) on the grid.  
 The current [alignment](#) will affect the placement of the image.  
 A = Image id number.  
 X = X position in grid for left corner of image.  
 Y = Y position in grid for lower corner of image.  
 X1 = % scaling factor for width (optional, default is 100).  
 Y1 = % scaling factor for height (optional, default is 100).
-  **LINKREF='A,text'** Used to define a link or annotation.  
 A = An ID for the link (letter or word, no spaces).  
 text = Text for annotation - leave out for a link.  
 (if used as a link, place LINKREF tag within the page to link to).
- LINK=A,X1,Y1,X2,Y2,B** Used to place a link or annotation  
 A = An ID for the link (from LINKREF).  
 X1 = Left grid coordinate for rectangle defining link.  
 Y1 = Bottom grid coordinate for rectangle defining link.  
 X2 = Right grid coordinate for rectangle defining link.  
 Y2 = Top grid coordinate for rectangle defining link.  
 B = Border (single character B, optional).  
[Click here to go to the sample report](#)
- REV=Y** Used to mirror the Y coordinate (REV=N turns it back off).  
 Normally, the Y coordinate runs from 0 at the top to whatever value is used in the BOX command for the bottom. This command reverses the coordinate system so 0 is at the bottom. This is added only as a convenience so you can do measurements based on the top or bottom.

<b>TITLE='text'</b>	Used to specify the title of the document.
<b>AUTHOR='text'</b>	Used to specify the author.
<b>SUBJECT='text'</b>	Used to specify the subject.
<b>MENUBAR=N</b>	Used to hide the menubar for the document.
<b>TOOLBAR=N</b>	Used to hide the toolbar for the document.
<b>ZOOM=X</b>	Used to specify the initial zoom factor. X = Value as a percent (ex. 100 means 100%).
<b>STARTPG=N</b>	Used to specify the page number to open the PDF at. N = A page number.
<b>SETPG=N</b>	Will reset the page number to this value (See <a href="#">&amp;page</a> ). N = The next page number to start numbering with. Issue this command just after the PAGE command. If you do it before, the counter will be increased by 1 more than what you set it as.
<b>SENDMAIL SMTP=text FROM=text FAKEFROM=text TO=text FAKETO=text CC=text FAKECC=text BCC=text PRIORITY=text SUBJECT=text BODY=text LOG=text</b>	Place this command and its options on a line by itself. This command will send the PDF via SMTP to the recipients specified. The SMTP, FROM, TO, SUBJECT and BODY are required, rest are optional. SMTP = The SMTP server (i.e. SMTP=mail.yourdomain.com). FROM = The FROM address. Should be a valid account for the SMTP server. (i.e. FROM="myaccount@yourdomain.com") FAKEFROM = The from address to show in the email - default is FROM setting. TO = The address(es) to send to. Separate multiple addresses with a comma. (i.e. TO="bob@yourdomain.com,sally@yourdomain.com") FAKETO = The to address to show in the email - default is TO setting. CC = The address(es) to CC to. Separate multiple addresses with a comma. (i.e. CC="bob@yourdomain.com,sally@yourdomain.com") FAKECC = The CC address to show in the email - default is CC setting. BCC = The address(es) to BCC to. Separate multiple addresses with a comma. (i.e. BCC="bob@yourdomain.com,sally@yourdomain.com") SUBJECT = The subject of the email. Enclose in quotes. BODY = The body of the email. Enclose in quotes. May also be an existing text file. PRIORITY = The priority - set to either "high" or "low". Normal is the default. LOG = An optional file to log the sent emails to.

**In-line commands**

Used for formatting and links

There are several tags that may be used within a line of text to produce special formatting or to link to the web or movie files.

A sample command you would use is:

X=1 Y=2>>Here is a **<SUP>superscript</SUP>** example.

**<FONT=X,Y>**

Used to switch font and/or size.

Example: This text is **Big** and this is small.

**<FCOLOR=R,G,B>**

Used to switch color in a line.

R, G and B are the Red, Green and Blue color components as a decimal from 0 to 1.

Example: Here is some **Color**.

**<BGCOLOR=R,G,B>**

Used to highlight text (Set BGCOLOR to 1,1,1 to turn off).

R, G and B are the Red, Green and Blue color components as a decimal from 0 to 1.

Example: Here is a **highlighted section** of text.

**<U> and <U2>**

Used to underline (or double underline) text (Use </U> and </U2> to turn off).

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

Used to start and stop superscripting.

Example: X<sup>2</sup> Here is a <sup>superscript</sup> example.

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

Used to start and stop subscripting.

Example: X<sub>2</sub> Here is a <sub>subscript</sub> example.

**<A HREF="location"> and </A>**

Used to insert a web link.

The "location" should be an address like <http://www.fytek.com>.

It can also be a mail address like <mailto:sales@fytek.com>.

The command would be written:

X=1 Y=5>>Goto <A HREF="http://www.fytek.com">Fytek, Inc.</A> web site.

Example: Goto [FyTek, Inc.](http://www.fytek.com) web site.

**<MOV HREF="location"> and </MOV>**

Used to insert a movie link.

The "location" should be a file such as <c:\\mov\\mymovie.mov>.

Note that the movie is not embedded in the PDF like JPEG files.

Users will need a copy of the movie and have it in the correct directory.

**<IMG SRC=A [HEIGHT=Y] [WIDTH=Z] [ALIGN=TOP|BOTTOM|MIDDLE]>**

Used to insert an image.


A = The image id number (from the [IMAGE](#) command).

HEIGHT and WIDTH are optional and if used are a percent of the normal image size (50 for 1/2 size, 200 to double the size, etc.)

ALIGN = TOP to align the image with the top of the text,

ALIGN = BOTTOM to align the image to the baseline of the text (default).

ALIGN = MIDDLE to align the image with the middle of the text.

Example: Here is a middle aligned image  in this string.

**&page** Will be replaced by the current page number (See [SETPG](#)).

**&runpage** The running page number for the entire pdf.

**&date** Will be replaced by the current date (Mon DD, YYYY format).  
Current value for &date is Sep 18, 2003.

**&time** Will be replaced by the current time (HH:MM pm format).  
Current value for &time is 7:26 pm.

The following are not in-line commands but do control the display of the in-line link commands.

**LINKCOLOR=R,G,B** Used to set the color of the links (default is blue).

**LINKLINE=X** Used to set the size of the link underlines.  
X = 0 to turn off underline of links else width of line (default is 1).

## Interactive Features

You may include the following interactive functions in your PDF documents. Users of Acrobat Reader will be able to fill in text and select check boxes for printing but will not be able to save what they type in. Users of the full version of Acrobat will be able to save the changes they make. Only one input statement may appear on a line. Most of the parameters are optional. See the sample report included for examples on many of these functions.

### INPUT=text

**TYPE=[TEXT|TEXTAREA|RADIO|BUTTON|SUBMIT|CHECKBOX|SIGNATURE|HIDDEN]**  
**[X1=number] [Y1=number] [X2=number] [Y2=number] [ROWS=number] [COLS=number]**  
**[ALIGN=L|C|R] [SIZE=number] [MAXLENGTH=number] [DESCR="text"]**  
**[WIDTH=number] [HEIGHT=number] [DATE="text"] [CHECKED]**  
**[NUMBER] [NOCOMMAS] [FCOLOR=color] [BGCOLOR=color] [BORDERCOLOR=color]**  
**[DEC=number] [READONLY] [REQUIRED] [ONCLICK="text"] [ONCHANGE="text"]**  
**[URL="text"] [VALUE="text"] [LOCK=["list"]] [LOCKEXCEPT="list"]**

- INPUT=text = The unique name for this widget.
- TYPE=text = Must be one of TEXT, TEXTAREA, RADIO, BUTTON, SUBMIT, CHECKBOX, SIGNATURE or HIDDEN.
- X1, Y1, X2, Y2 = The coordinates of the widget based on the current grid. If X1 or Y1 are not specified, the current X and/or Y coordinates are used. Set X2 and/or Y2 to override the setting that is calculated based on the SIZE option.
- ROWS=number = The number of rows (based on current font) for a TEXTAREA widget.
- COLS=number = The number of characters across (based on current font) for a TEXTAREA widget.
- ALIGN=L|C|R = The alignment of the text within the text box (left, center or right).
- SIZE=number = The number of characters to size a TEXT fill-in to.
- MAXLENGTH=number = The maximum number of characters to allow in a TEXT fill-in.
- DESCR="text" = The description for the widget (used in JavaScript error messages).
- WIDTH=number = The width (based on the current grid) to size the TEXT fill-in to.
- HEIGHT=number = The height (based on the current grid) to size the TEXT fill-in to. The default height is based on the current font size.
- DATE="text" = The date format to use (default is mm/dd/yyyy).
- CHECKED = Sets a CHECKBOX or RADIO button to checked initially.
- NUMBER = Sets the TEXT fill-in to allow numbers only.
- NOCOMMAS = Turns off the display of commas in a TEXT fill-in with the NUMBER option.
- FCOLOR=color = The color of the text (default is black).
- BGCOLOR = The background color.
- BORDERCOLOR = The border color (default is black).
- DEC=number = The number of decimal places for TEXT fill-in with the NUMBER option.
- READONLY = Sets the widget as read only.
- REQUIRED = Forces the user to enter a value before doing a submit.
- ONCLICK="text" = JavaScript to execute when user clicks on a BUTTON.
- ONCHANGE="text" = JavaScript to execute when the value changes (technically, this is executed when an onBlur takes place - works best in Acrobat 5.0 and up).
- URL="text" = The URL to submit to for a SUBMIT button.
- VALUE="text" = The default value for a TEXT fill-in or label for a BUTTON.
- LOCK=["list"] = A comma separated list of widget names to lock down once the signature field is filled in - leave out the ="list" part to lock down all fields.
- LOCKEXCEPT="list" = A comma separated list of widget names to exclude from lock down once the signature field is filled in.

Any text after the >> is the default value for TEXTAREAs or the label for RADIO buttons and checkboxes.

Here is an example:

Type your name:

Company

Individual

New Account

Comments:

Signature:

First #:

Second #:

Result:

**SELECT=text**     **[DESCR="text"] [READONLY]**

SELECT=text = The unique name for this widget.

DESCR="text" = The description for the widget (used in JavaScript error messages).

READONLY = Sets the widget as read only.

**OPTION=text**

OPTION=text - The value for the OPTION.

Any text after the >> is put into the drop down list for the SELECT widget.

The OPTION commands are placed between the SELECT and ENDSELECT commands.

**ENDSELECT**     Ends the SELECT statement

```
SELECT=s1 X1=8>>
OPTION=1>>Checking account
OPTION=2 SELECTED>>Savings account
OPTION=3>>Other
ENDSELECT>>
```

Transfer from



**Here are some samples:**

## Simple "Hello World" example

```
PAGE=8.5,11>>  
BOX=0,0,8.5,11,2,2>>  
FONT=5,36 ALIGN=C X=1 Y=1>>Hello, world
```

## More complex example with multiple pages and outline

```
IMAGE='1,images/test.jpg'>>  
PAGE=8.5,11 NEWFORM=1>>  
TITLE='New Report'>>  
BOX=0,0,8.5,1,2,2>>  
FONT=2,18 ALIGN=C X=1 Y=1>>Report Title  
LINEW=.01>>  
LINE=0,1,2,1>>  
OFFSET=.01,.1>>  
  
PAGE=8.5,11 USEFORM=1 OUTLINE='1,Main Page,C'>>  
BOX=0,1,8.5,10,20,50>>  
FONT=2,11 ALIGN=L>>  
X=1 Y=2>>Here is some text on line 1.  
Y=+>>Here is some text on line 2.  
Y=+>>Here is some text on line 3.  
Y=+>>  
Y=+>>Here is some text on line 5.  
PUTIMG=1,3,30>>  
PAGE=8.5,11 USEFORM=1 OUTLINE='2,Sub-outline to Main'>>  
BOX=0,1,8.5,10,20,50>>  
FONT=2,24 ALIGN=C X=1 Y=25>>Here is some text in the middle  
PAGE=8.5,11 USEFORM=1 OUTLINE='1,Summary'>>  
BOX=0,1,8.5,10,20,50>>  
FONT=2,11 ALIGN=L X=1 Y=35>>Here is text in a box  
RECT=1,35,5,34>>
```

### Symbol (#13)

A =	A	a =	α	0 =	0
B =	B	b =	β	1 =	1
C =	X	c =	χ	2 =	2
D =	Δ	d =	δ	3 =	3
E =	E	e =	ε	4 =	4
F =	Φ	f =	φ	5 =	5
G =	Γ	g =	γ	6 =	6
H =	H	h =	η	7 =	7
I =	I	i =	ι	8 =	8
J =	ϑ	j =	φ	9 =	9
K =	K	k =	κ	.	.
L =	Λ	l =	λ	:	:
M =	M	m =	μ	;	;
N =	N	n =	ν	!	!
O =	O	o =	ο	@ =	≅
P =	Π	p =	π	# =	#
Q =	Θ	q =	θ	\$ =	∃
R =	P	r =	ρ	% =	%
S =	Σ	s =	σ	^ =	⊥
T =	T	t =	τ	& =	&
U =	Υ	u =	υ	* =	*
V =	ς	v =	ϖ	( =	(
W =	Ω	w =	ω	) =	)
X =	Ξ	x =	ξ		
Y =	Ψ	y =	ψ		
Z =	Z	z =	ζ		

### Zapf Dingbats (#14)

A =	☆	a =	✿	0 =	✍
B =	✂	b =	✱	1 =	✍
C =	✂	c =	✿	2 =	✍
D =	✂	d =	✿	3 =	✓
E =	✂	e =	✿	4 =	✓
F =	◆	f =	✿	5 =	X
G =	◇	g =	✿	6 =	X
H =	★	h =	✿	7 =	X
I =	☆	i =	✿	8 =	X
J =	⊕	j =	✿	9 =	+
K =	☆	k =	✱	.	✍
L =	☆	l =	●	:	+
M =	☆	m =	○	;	+
N =	★	n =	■	!	✂
O =	☆	o =	□	@ =	✂
P =	☆	p =	□	# =	✂
Q =	★	q =	□	\$ =	✂
R =	★	r =	□	% =	✂
S =	*	s =	▲	^ =	✂
T =	*	t =	▼	& =	✂
U =	✿	u =	◆	* =	✂
V =	*	v =	◇	( =	✂
W =	*	w =	▷	) =	✂
X =	*	x =			
Y =	*	y =			
Z =	✿	z =	■		

You can use the [ADDFONT](#) command to add your own True Type Fonts.  
They will be numbered starting at 15 based on the order they appear in the command file or array of commands.

Click on a command to jump to the page containing that command

<a href="#">&amp;page</a> (in-line command)	<a href="#">PAGE</a>
<a href="#">&amp;date</a> (in-line command)	<a href="#">POLY</a>
<a href="#">&amp;runpage</a> (in-line command)	<a href="#">PUTIMG</a>
<a href="#">&amp;time</a> (in-line command)	
<a href="#">A</a> (in-line command)	<a href="#">RECT</a>
<a href="#">ADDFONT</a>	<a href="#">RECTF</a>
<a href="#">ALIGN</a> (in-line command for WRAPBLOCK)	<a href="#">REND</a>
<a href="#">ALIGN</a>	<a href="#">REV</a>
<a href="#">AUTHOR</a>	<a href="#">ROTATE</a>
<a href="#">BGCOLOR</a> (in-line command)	<a href="#">SCOLOR</a>
<a href="#">BOX</a>	<a href="#">SELECT</a>
	<a href="#">SETPG</a>
<a href="#">CIRCLE</a>	<a href="#">STARTPG</a>
<a href="#">CIRCLEF</a>	<a href="#">SUB</a> (in-line command)
<a href="#">COMP</a>	<a href="#">SUBJECT</a>
<a href="#">CURVE</a>	<a href="#">SUP</a> (in-line command)
<a href="#">ENDTXTBLOCK</a>	<a href="#">TBL</a>
	<a href="#">TBLALIGN</a>
<a href="#">FCOLOR</a>	<a href="#">TBLCELLCOLOR</a>
<a href="#">FCOLOR</a> (in-line command)	<a href="#">TBLCOLALIGN</a>
<a href="#">FILL</a> (in-line command for WRAPBLOCK)	<a href="#">TBLCOLCOLOR</a>
<a href="#">FILLLINE</a> (in-line command for WRAPBLOCK)	<a href="#">TBLCOLOR</a>
<a href="#">FONT</a>	<a href="#">TBLDATA</a>
<a href="#">FONT</a> (in-line command)	<a href="#">TBLROW</a>
	<a href="#">TBLROWALIGN</a>
<a href="#">IMAGE</a>	<a href="#">TBLROWCOLOR</a>
<a href="#">IMG</a> (in-line command)	<a href="#">TBLSET</a>
<a href="#">INPUT</a>	<a href="#">TEXTBLOCK</a>
	<a href="#">TITLE</a>
<a href="#">LINE</a>	<a href="#">TOOLBAR</a>
<a href="#">LINEC</a>	
<a href="#">LINED</a>	<a href="#">SENDMAIL</a>
<a href="#">LINEW</a>	
<a href="#">LINK</a>	<a href="#">U and U2</a> (in-line command)
<a href="#">LINKCOLOR</a>	<a href="#">USEFORM</a>
<a href="#">LINKLINE</a>	
<a href="#">LINKREF</a>	<a href="#">WRAPBLOCK</a>
<a href="#">MENUBAR</a>	<a href="#">X</a> (in-line command for WRAPBLOCK)
<a href="#">MOV</a> (in-line command)	<a href="#">X</a>
	<a href="#">X2</a>
<a href="#">NEWFORM</a>	
	<a href="#">Y</a>
<a href="#">OFFSET</a>	
<a href="#">OUTLINE</a>	<a href="#">ZOOM</a>