Bluebeam Script Reference
Version 2.0.9

Bluebeam, Inc.
Published February 09, 2018
Applies to Bluebeam Q® and Bluebeam Revu® eXtreme®

Introduction

Bluebeam Scripts can be run from the command with the tool ScriptEngine.exe which can be found as part of the installation. Scripts can either be run directly from the command line, or they may be defined in a script file. The standard extension for a script file is bci.

The Script Engine maintains a stack of PDF files. The topmost file in the stack is considered active. Most operations work off of the active document. Opening a file places it at the top of the stack, closing a file removes it from the top of the stack.

Finally for debugging purposes you may specify a –d parameter which will output verbose logs to the console.

Interactive Mode

The Script Engine can be run in two modes, normal, and interactive. Historically the scripting engine in the normal mode would exit immediately after running a script. In order to accomplish tasks where return values need to be consumed before execution could continue, the engine would need to be run several times, opening and closing the file repeatedly which is inefficient. Also, the few commands that did return values were not consistent, and it was difficult to handle errors on a per command basis.

The interactive mode solves these problems. In the interactive mode, the scripting engine does not immediately exit. It waits for commands on standard in, and then writes a response on standard out consistently and reliably. After executing a command, the engine will continue to listen on standard in until it receives 'exit'.

The Script Engine can be started in the interactive mode by supplying a -i command line argument. The response always starts with a numeric code, and may have additional data based on that code.

When the return code is 0 or greater it indicates both success and the count of return values. The return values are written one per line as strings on standard out. If the code is 0, indicating a count of zero, no additional lines are written on standard out.

If the code is less than 0 then an error occurred. The code is followed by a colon and then a description of the error. The possible codes are as follows:

-1 = Parse Error
-2 = Command Failed
-3 = Exception
-4 = Software not Registered

For example: -2:Invalid Page style

This error indicates that the command failed, and then the description indicates exact problem that was encountered.

Throughout the documentation, commands may refer to having return values. Those that do will return their values as described above when in interactive mode. When not in interactive mode newer commands will still follow the same return pattern, however, for backwards compatibility some older commands may return the values differently.

Syntax

Bluebeam Scripts are a series of commands that are single word identifiers followed by a comma delimited list of parameters enclosed in paranthesis.

For example:

ScriptEngine.exe Script('myscript.bci')

In this example 'Script' is the Script command which would then run the commands listed in the file 'myscript.bci', which is a parameter.

There are several different types in Bluebeam Script:

Bool: Boolean values (true or false)
Number: Either an Integer or Real (1, 3.5, 0.2 ...)
String: Quoted list of characters ("document.pdf" ...) (the escape character is |)
Name: Unquoted list of characters containing lettings and numbers only. (Print, View, Flatten ...)
Color: A special string that represents a color that is either a name such as "Black" or "Red", or a hex string such as "#FF0000" which indicates Red, or an integer that contains the RGB values as packed bytes where B is the lowest byte.
Dictionary: A special string that represents a set of key/value pairs. The syntax is as follows: {"Key1":"Value1","Key2":"Value2"}
Date: A special string that represents an ISO 8601 compliant date in UTC. The format is YYYY-MM-DDThh:mm:ss

Additionally comments may be added by using the '%' character. Any characters following the comment character will be ignored until a new line character is encountered.

Special Note about Strings
When commands are run directly from the command line, not from within a script file, the single quote is recommended to encapsulate strings. Otherwise the quote

characters need to be escaped. Additionally, the entire script should be quoted.

For example:
ScriptEngine.exe "Open('c:\Test Folder\test.pdf') Flatten() Save('c:\Test Folder\flattened.pdf') Close()"

Markups

There are commands that return and accept string dictionaries containing key/value pairs of markup properties. The following keys are supported by those commands:

type = The type of markup such as square or polygon
page = The page index that the markup occurs on
author = The author of the markup
subject = The subject of the markup
comment = The comment of the markup
color = The color of the markup
colorfill = The fill color of the markup
colortext = The text color of the markup
opacity = The opacity of the markup from 0 to 1
opacityfill = The fill opacity of the markup from 0 to 1
rotation = The rotation of the markup in degrees from 0 to 360
parent = The markup id of the markup's parent. Needed to understand the parent/child relationship of grouped markups
grouped = Boolean value indicating if the markup is grouped
status = The status of the markup, valid states are "Accepted", "Rejected", "Cancelled", "Completed", and "None"
checked = Boolean value indicating that the markup is checked or unchecked
locked = Boolean value indicating that the markup is locked or unlocked
datecreated = The creation date of the markup
datemodified = The modified date of the markup
linewidth = The width of the line in points where 72 points equals 1 inch. For most markups the range is 0 to 12.
linestyle = The style of the line, valid styles are "solid", "dashed1", "dashed2", "dashed3", "dashed4", "dashed5", "dashed6", "cloudy1", and "cloudy2"
x = The x coordinate of the markup in points where 72 points equals 1 inch
y = The y coordinate of the markup in points where 72 points equals 1 inch
width = The width of the markup in points where 72 points equals 1 inch
height = The height of the markup in points where 72 points equals 1 inch
space = The space defined in the PDF that the markup resides in (read-only)
layer = The layer that the markup is assigned to (read-only)
captureCount = The number of capture images and video attached to the markup (read-only)

Example

This following example will run a script named myscript.bci, which will open a PDF file, import markups, flatten the markups, and then save and close the file.

ScriptEngine.exe Script('myscript.bci')

Where myscript.bci contains:

```
Open("c:\source.pdf")
Import("c:\user.bax")
Flatten()
Save("c:\output.pdf")
Close()"
```

Commands

BalancePages

Description
Inserts blank pages into active document to balance the total number of pages to an odd, even, specific count, or specific page division

Parameters
pType [String]: Specifies how blank pages will be inserted at the end of the pdf file as follows:

even = Inserts one page if needed to make count even
odd = Inserts one page if needed to make count odd
n = Inserts pages to make page count divisible by n, n is a number
-n = Inserts pages to make page count at least n pages, n is a number

pWidth [String, Optional]: Width of page in inches, last means width of last page
pHeight [String, Optional]: Height of page in inches, last means height of last page
pStyle [Number, Optional]: Page Style as follows:

0 = Blank
1 = Notebook
2 = 1/8" Grid
3 = ¼" Grid
4 = Engineering Grid
5 = 0.5 cm Grid
6 = 1 cm Grid
7 = ½" Isometric Grid
8 = 0.5 cm Isometric Grid

Example

```
BalancePages("even")
BalancePages(4, 8.5, 11, 1)
```

Batch

Description
Runs a script file on each specified PDF file. Any set of either PDF files or folders containing PDF files may be passed in as arguments. The filename of each PDF file will be passed in as arg0 to the script. See the Script command for more information about arg0.

Parameters
pScriptPath [String]: Filename of the script file to run
pIncludeSubFolders [Bool]: If true sub-folders will be processed recursively
pPath [String]: File or directory of the PDF files to loop over
pPathN [String, Optional, ...]: File or directory of additional PDF files to loop over

Example
```
Batch("script1.bci", "c:\Directory")
Batch("script1.bci", false, "c:\\Directory\\file1.pdf", "c:\\Directory\\file2.pdf", ...)
```

Where script1.bci contains:

```
Open(arg0)
ColorProcess("black", "white") % Convert file to grayscale
Close(true) % True specifies that the document should be saved before closing
```

Close

Description
Closes the active document removing it from the top of the stack.

Parameters
pSave [Bool, Optional]: Boolean value specifying whether to save the document before closing
pSaveMode [Number, Optional]: Save Mode as Follows:

0 = Incremental Updates
1 = Publish
2 = Publish Compressed

Example
```
Close()
Close(true)
Close(true, 2)
```

ColorProcess

Description
Converts page content colors to a color or gray scale.

Parameters
pStartColor [Color]: Start color to convert source colors to, usually darker color
pEndColor [Color]: End color to convert source colors to, usually lighter color
pScale [Bool, Optional]: Indicates that colors should be scaled from start to end
pProcessImages [Bool, Optional]: Images should be converted to new colors
pPageRange [String, Optional]: List or range of pages to be processed, -1 will process all pages, exp: 1,2,10-20

Example
ColorProcess("black", "white") % Convert to grayscale
ColorProcess("Red", "white", true, true, "10-20")

ColumnDataGet

Description
Retrieves the Custom Column data associated with a particular markup and returns the data for more than one column as a string dictionary, or a single column as a string.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pColumn [String, Optional, ...]: Column name for which the data is associated

Example
ColumnDataGet(0, "NDFJKXLKJKLDFY")
ColumnDataGet(0, "NDFJKXLKJKLDFY", "Material", "Subtotal")

ColumnDataGetDict

Description
Retrieves the Custom Column data associated with a particular markup and returns the data as a string dictionary.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pColumn [String, Optional, ...]: Column name for which the data is associated

Example
ColumnDataGetDict(0, "NDFJKXLKJKLDFY")
ColumnDataGetDict(0, "NDFJKXLKJKLDFY", "Material", "Subtotal")

ColumnDataSet

Description
Sets Custom Column data for a particular markup.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pData [String]: Custom Column data as a string dictionary

Example
ColumnDataSet(0, "NDFJKXLKJKLDFY", "{'Material':'Glass'}")

ColumnsExport

Description
Exports the Custom Column definition of the active document to an .xml file.

Parameters
pFileName [String]: Filename to export the columns into

Example
ColumnsExport('columns.xml')

ColumnsImport

Description
Imports a Custom Column definition .xml file into the active document overwriting any existing Custom Columns. An .xml file can be generated by either the command ColumnsExport, or from within Bluebeam Revu.

Parameters
pFileName [String]: Filename of the Custom Column definition .xml file to import into the active document

Example
ColumnsImport("columns.xml")

Combine

Description
Takes each file specified as a parameter and combines them into a new PDF file that becomes the active document. The save command as seen in the example would save the newly combined PDF file.

Parameters
pFile1 [String]: Filename of the first pdf file to combine
pFile2 [String]: Filename of the second pdf file to combine
pFileN [String, Optional, ...]: Filename of additional pdf files to combine.

Example
Combine("document1.pdf", "document2.pdf", "document3.pdf" ...)
Save("output.pdf")

CreatePDFAReport

Description
Generate a text report on whether a PDF file is PDF/A-1b compliant. Results will always be appended to the report file specified by the pFileName parameter.

Parameters
pFileName [String]: Absolute full path to the report file to be created or updated.

Example
CreatePDFAReport("C:\pdfa\report.txt")

DeleteFile

Description
Deletes a file from specified location.

Parameters
pFileName [String]: Filename to delete from the file system

Example
DeleteFile("c:\Directory\Filename.pdf")

EmbedJavaScript

Description
Embeds the sepecified JavaScript file as a document level script in the active document.

Parameters
pName [String]: Name of JavaScript Code
pFile [String]: JavaScript file to embed

Example
EmbedJavaScript("File.js")

Export

Description
Exports the markups in the active document to the specified output file optionally using a User ID to filter on.

Parameters
pOuputBAX [String, Optional]: Filename to export the markups into
pUserID [String, Optional]: User ID as used in bFX File Exchange to filter on when exporting markups

Example
Export("output.bax")
Export("output.bax", "12345")

FilePropertyGet

Description
Returns the value of a file property that corresponds to the key passed in as a parameter. When not running in the Interactive Mode, this command will output the result straight to the console without first outputting a count for backwards compatibility.

Parameters
pKey [String]: Key of file property to retrieve

Example

FilePropertyGet("Author")

FilePropertyList

Description
Returns the keys of all file properties in the active document as a list of strings.

Parameters
None

Example
FilePropertyList()

FilePropertySet

Description
Sets a file property in the active document based on the specified key and value.

Parameters
pKey [String]: Key of file property to set
pValue [String]: Desired value of file property

Example
FilePropertySet("Author", "Homer J. Simpson")

Flatten

Description
Takes the active document and flattens all markups to be part of the page content.

Parameters
pRecoverable [Bool, Optional]: Specifies whether or not the flatten process is reversible
pFlags [Number, Optional]: Specifies what type of markups to flatten

Default = 8191
Image = 1
Ellipse = 2
Stamp = 4
Snapshot = 8
Text and Callout = 16
Ink and Highlighter = 32
Line and Dimension = 64
Measure Area = 128
Polyline = 256
Polygon and Cloud = 512
Rectangle = 1024
Text Markups = 2048
Group = 4096
File Attachment = 8192
Flags = 16384

Notes = 32768
Form Fields = 65536

Add together all values that should be flattened

pPageRange [String, Optional]: List or range of pages to be flattened, -1 will flatten all pages, exp: 1,2,10-20
pLayerName [String, Optional]: Layer Name to flatten markups to

Example
Flatten()
Flatten(true)
Flatten(true, 9) % Flattens Images (1) and Snapshots (8)

FormExport

Description
Exports the form data in the active document to a .xml, .csv, or .fdf file.

Parameters
pFileName [String]: Filename (.xml, .csv, or .fdf) to export the form data into

Example
FormExport("formdata.fdf")

FormImport

Description
Imports an FDF file containing form data into the active document.

Parameters
pFileName [String]: Filename of FDF file to import into the active document

Example
FormImport("formdata.fdf")

FormMerge

Description
Merges the form data from a set of PDF files into one output file, either an .xml or .csv file.

Parameters
pFileName [String]: Filename (.xml or .csv) to merge the form data into
pIncludeSubFolders [Bool]: If true sub-folders will be processed recursively
pPath [String]: File or directory of the PDF files to process for merging
pPathN [String, Optional, ...]: File or directory of additional set of PDF files to process for merging

HeaderAndFooter

Description

Applies headers and footers to the active document.There are many codes that can be passed in as part of the header or footer text that will be dynamically substituted when the text is applied to the document.

Page Index Codes
<<1>>, <<1 of n>>, <<1/n>>, <<Page 1>>, <<Page 1 of n>>

Date Codes
<<M/d>>, <<M/d/yy>>, <<M/d/yyyy>>, <<MM/dd/yy>>, <<MM/dd/yyyy>>, <<d/M/yy>>, <<d/M/yyyy>>, <<dd/MM/yy>>, <<dd/MM/yyyy>>, <<MM/yy>>, <<MM/yyyy>>, <<ddd MMM d, yyyy>>, <<dddd MMMM d, yyyy>>, <<MM/dd/yyyy h:mm tt>>, <<dd/MM/yyyy HH:mm>>

Bates Numbering
<<Bates Number#Digits#Start#Prefix#Suffix>>

Examples:
<<Bates Number#6#1#_Prefix#_Suffix>>, <<Bates Number#6#1>>

File Properties
Headers and Footers also support pulling file property data from the PDF, any file property key can be used such as:
<<Title>>, <<Author>>, <<Client>> ...

These are additional special codes:
<<FileName>>, <<Path>>, <<PageLabel>>

Parameters

pTopLeft [String]: Header text for top left of page
pTopCenter [String]: Header text for top center of page
pTopRight [String]: Header text for top right of page
pBottomLeft [String]: Footer text for bottom left of page
pBottomCenter [String]: Footer text for bottom center of page
pBottomRight [String]: Footer text for bottom right of page.
pMarginLeft [Number, Optional]: Left margin in points (72 points per inch)
pMarginTop [Number, Optional]: Top margin in points (72 points per inch)
pMarginRight [Number, Optional]: Right margin in points (72 points per inch)
pMarginBottom [Number, Optional]: Bottom margin in points (72 points per inch)
pFont [String, Optional]: Name of font to use with header and footer
pSize [Number, Optional]: Size of font
pBold [Bool, Optional]: Emboldens font
pItalic [Bool, Optional]: Italicizes font
pUnderline [Bool, Optional]: Underlines text
pColor [Color, Optional]: Font color
pFitToContent [Bool, Optional]: Make content of page fit inside margins
pBatesOffset [Number, Optional]: The offset of the bates numbering
pBatesKey [String, Optional]: The unique key used to persistantly store the last used Bates offset. Use this key to ensure that
every bates number will be uniqiue across documents.
pPageRange [String, Optional]: List or range of pages to apply the header and footer to, -1 will apply to all pages, exp: 1,2,10-20

Example

```
HeaderAndFooter("", "<<dddd MMMM d, yyyy>>","<<h:mm ss tt>>","<<Author>>","","<<Page 1 of n>>",
108, 28.8, 108, 48, "Blackadder ITC", 10.0, false, false, false, "Red"
```

,false, 93, "1,3,5,10-20")

Import

Description
Imports the markups from list of files specified as parameters into the active document.

Parameters
pBAXorPDF [String, ...]: Filename of a bax or pdf file to import into the active document

Example
Import("markups1.bax", "markups2.bax" ...)
Import("revA.pdf" ...)
Import("markups1.bax", "revB.pdf" ...)

InsertBlankPages

Description
Inserts new blank pages into the active document using the specified parameters for width, height, count and style. The default count is 1 and the default style is blank.

Parameters
pIndex [Number]: Page Index in the active document to insert pages after, 0 is before first page.
pWidth [Number]: Width of page in inches
pHeight [Number]: Height of page in inches
pCount [Number, Optional]: Number of pages to insert
pStyle [Number, Optional]: Page Style as follows:

0 = Blank
1 = Notebook
2 = 1/8" Grid
3 = ¼" Grid
4 = Engineering Grid
5 = 0.5 cm Grid
6 = 1 cm Grid
7 = ½" Isometric Grid
8 = 0.5 cm Isometric Grid

Example
InsertBlankPages(0, 8.5, 11)
InsertBlankPages(2, 8.5, 11, 10, 3)

InsertPages

Description
Inserts a PDF file into the active document using the specified parameters to control what additional data to be additionally imported such as bookmarks, file attachments, and file properties

Parameters
pIndex [Number]: Page Index in the active document to insert pages after, 0 is before first page.

pFileName [String]: Filename of document to insert
pBookmarks [Bool, Optional]: Insert bookmarks from inserted file, default is false
pAttachments [Bool, Optional]: Insert file attachments from inserted file, default is false
pProperties [Bool, Optional]: Merge document properties from inserted file, default is false
pLayers [Bool, Optional]: Merge document layers from inserted file, default is false
pUseFileName [Bool, Optional]: User file name as Page Label, default is false

Example
InsertPages(0, "Document.pdf")
InsertPages(0, "Document.pdf", true, true, true, true)

MarkupCaptureExport

Description
Exports the photos and videos attached to the markup.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pFolder [String, Optional]: The folder to extract the attachments to. If the folder does not exist it will be created. By default the folder will be the markup ID

MarkupCopy

Description
Returns an xml string that contains raw markup data that can be passed into MarkupPaste to be placed at a new location. If the markup is the parent of a group, then the whole group will be copied.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup

Example
MarkupCopy(1, "YIBKQIOZSROMNDGD")

MarkupDelete

Description
Deletes a particular markup from the active document.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup

Example
MarkupDelete(1, "YIBKQIOZSROMNDGD")

MarkupGet

Description
Retrieves the properties associated with a particular markup that returns multiple properties as a string dictionary, or a single property as a string. Refer to the Markups section for description of the available properties.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pProperty [String, Optional, ...]: Particular markup property to retrieve

Example
MarkupGet(1, "YIBKQIOZSROMNDGD")
MarkupGet(1, "YIBKQIOZSROMNDGD", "subject")
MarkupGet(1, "YIBKQIOZSROMNDGD", "type", "comment")

MarkupGetEx

Description
Retrieves the properties, including the custom ones, associated with a particular markup and returns those properties as a string dictionary, or a single property as a string. Refer to the Markups section for description of the available properties.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pProperty [String, Optional, ...]: Particular markup property to retrieve

Example
MarkupGetEx(1, "YIBKQIOZSROMNDGD")
MarkupGetEx(1, "YIBKQIOZSROMNDGD", "subject")
MarkupGetEx(1, "YIBKQIOZSROMNDGD", "type", "comment")

MarkupGetExList

Description
Retrieves all of the properties, including the custom ones,for all of the markups on a given page as a string dictionary,or a single property as a string. Refer to the Markups section for description of the available properties.

Parameters
pPageIndex [Number]: Page Index

Example
MarkupGetExList(1)

MarkupList

Description
Retrieves the list of markup IDs associated with a particular page.

Parameters
pPageIndex [Number]: Page Index

Example
MarkupList(1)

MarkupPaste

Description
Pastes a markup passed in as raw XML at the coordinates provided. The raw XML would have been returned from a call to MarkupCopy. Returns a list of markup IDs of the pasted markups.

Parameters
pPageIndex [Number]: Page Index of paste destination
pXML [String]: XML string containing raw markup data
pX [Number]: X coordinate of paste location in points (72 points per inch)
pY [Number]: Y coordinate of paste location in points (72 points per inch)

Example
MarkupPaste(1, "< ... Raw XML returned from MarkupCopy(...) ...>", 144, 72)

MarkupSet

Description
Sets properties for a particular markup. The data is passed in as a string dictionary containing key/value pairs. Refer to the Markups section for description of the available properties.

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pData [String]: Markup properties as a string dictionary

Example
MarkupSet(1, "YIBKQIOZSROMNDGD", "{'comment':'The color is red','color':'#FF0000'}")

MarkupThumbnail

Description
Generates a thumbnail of a markup. If the markup is the parent of a group, then the whole group will be rendered. Can have an extension of most common image formats including (.bmp, .png, .jpg ...).

Parameters
pPageIndex [Number]: Page Index of the markup
pMarkupID [String]: ID associated with the markup
pWidth [Number]: Desired width in pixels of output thumbnail image
pHeight [Number]: Desired height in pixels of output thumbnail image
pPercentage [Number]: Desired percentage of the thumbnail that the markup should cover
pIncludePageContent [Bool]: Boolean value specifying if the thumbnail should include the background page content
pFilename [String]: Filename of desired output thumbnail image

pIncludeAllMarkups [Bool, Optional]: Boolean value specifying if all markups on the page should be included in the thumbnail

Example
MarkupThumbnail(1, "YIBKQIOZSROMNDGD", 256, 256, 0.5, true, "thumb.png", false)

New

Description
Creates a new blank PDF file using the specified parameters for width, height, count and style. The default size is 8.5x11", the default count is 1, and the default style is blank.

Parameters
pWidth [Number, Optional]: Width of page in inches
pHeight [Number, Optional]: Height of page in inches (required if width specified)
pCount [Number, Optional]: Number of pages to create on new document
pStyle [Number, Optional]: Page Style as follows:

0 = Blank
1 = Notebook
2 = 1/8" Grid
3 = ¼" Grid
4 = Engineering Grid
5 = 0.5 cm Grid
6 = 1 cm Grid
7 = ½" Isometric Grid
8 = 0.5 cm Isometric Grid

Example
New()
New(8.5, 11)
New(8.5, 11, 10, 3)

Open

Description
Opens the specified PDF file and pushes it to the top of the document stack thus making it active. If a password is required to open the PDF file, the password can be passed as the second parameter.

Parameters
pFilename [String]: Filename of PDF file to open
pPassword [String, Optional]: Password to open PDF file

Example
Open("document.pdf")
Open("document.pdf", "abacadabra")

OpenImage

Description

Converts and combines image files to PDF and pushes it to the top of the document stack thus making it active.

Parameters
pImage1 [String]: Filename of image to open
pImageN [String, Optional, ...]: Filename of additional images to open

Example
OpenImage("Picture1.jpg")
OpenImage("drawing.tiff", "scan.png", "photo.jpg")

PageCount

Description
Returns the number of pages in the active document. When not running in the interactive mode, this command will output the result straight to the console without first outputting a count for backwards compatibility.

Parameters
None

Example
PageCount()

PageDelete

Description
Deletes pages from the current document.

Parameters
pPageRange [String]: List or range of pages to delete. Cannot delete all pages. exp: 1,2,10-20

Example
PageDelete("1,2,10-20")

PageExtract

Description
Extracts pages from the currently active pdf document.

Parameters
pPageRange [String]: List or range of pages to Extract, -1 will extract all pages, exp: 1,2,10-20
pFileNameOrDirectory [String]: Filename or directory to save the extracted pages to
pPrefix [String, Optional]: A prefix that can be appended to the filename
pSuffix [String, Optional]: A suffix that can be appended to the filename

Example
PageExtract("1-3", "c:\Directory\file.pdf")
PageExtract("1,5,10-20", "c:\Directory")

PageExtract("1,5,10-20", "filename.pdf")
PageExtract("1,5,10-20", "", "prefix_", "_suffix")

PageRotate

Description
Rotates the active document pages by 90 degree increments.

Parameters
pRotations [Number]: Degrees to rotate pages by, must be multiple of 90
pPortrait [Bool, Optional]: Include portrait pages, default is true
pLandscape [Bool, Optional]: Include landscape pages, default is true
pPageRange [String, Optional]: List or range of pages to be Rotated, -1 will rotate all pages, exp: 1,2,10-20

Example
PageRotate(90)
PageRotate(-90, false, true, "10-20")

PageRotateGet

Description
Returns the page rotation of the active document corresponding to the index passed in as an integer in degrees.

Parameters
pIndex [Number]: Page index to get the page rotation from

Example
PageRotateGet(1)

PageSize

Description
Returns the page size of the active document corresponding to the index passed in as a parameter as a string list containing numbers formatted as strings. The first string is the page width, the second string is the page height. When not running in the interactive mode, this command will output the result straight to the console without first outputting a count for backwards compatibility.

Parameters
pIndex [Number]: Page index to get page size from

Example
PageSize(1)

Print

Description
Prints the active document to a physical printer. There are only 3 syntaxes available for this function, see

examples below. If advanced printing options are required, all 9 parameters must be specified.

Parameters
pPrinter [String, Optional]: Name of Printer
pPageSize [String, Optional]: Page size as it appears on Printer
pLandscape [Bool, Optional]: Whether to print landscape(true) or portrait(false)
pPageRange [String, Optional]: List or range of pages to be printed, -1 will print all pages, exp: 1,2,10-20
pAutoRotateAndCenter [Number, Optional]: Automatically rotated and center page content on paper.

-1 : Autorotate and center -90
0 : No autorotate and center
1 : Autorotate and center 90

pScaleType [Number, Optional]: Specifies how to scale when printing according to the following:

0 = None
1 = Fit to Paper
2 = Shrink large Images
3 = Custom

pCustomScale [Number, Optional]: If scale type is set to custom, this is the custom scale value (e.g. 0.5 would be 50%)
pDim [Bool, Optional]: Specifies whether to dim the content when printing
pCopies [Number, Optional]: Number of copies to print

Example
```
Print()
Print("HP Laserjet")
Print("HP Laserjet", "letter", false, "1-3", true, 1, 1, false, 1)
```

PrintToFile

Description
Prints the active document to a file. There are only 3 syntaxes available for this function, see examples below. If advanced printing options are required, all 10 parameters must be specified.

Parameters
pFileName [String]: File to print output to
pPrinter [String, Optional]: Name of Printer
pPageSize [String, Optional]: Page size as it appears on Printer
pLandscape [Bool, Optional]: Whether to print landscape(true) or portrait(false)
pPageRange [String, Optional]: List or range of pages to be printed, -1 will print all pages, exp: 1,2,10-20
pAutoRotateAndCenter [Number, Optional]: Automatically rotated and center page content on paper.

-1 : Autorotate and center -90
0 : No autorotate and center
1 : Autorotate and center 90

pScaleType [Number, Optional]: Specifies how to scale when printing according to the following:

0 = None
1 = Fit to Paper
2 = Shrink large Images

3 = Custom

pCustomScale [Number, Optional]: If scale type is set to custom, this is the custom scale value (e.g. 0.5 would be 50%)
pDim [Bool, Optional]: Specifies whether to dim the content when printing
pNumberOfCopies [Number, Optional]: Number of copies to print

Example
PrintToFile("out.prn")
PrintToFile("out.prn", "HP Laserjet")
PrintToFile("out.prn", "HP Laserjet", "letter", false, "1-3", true, 1, 1, false, 1)

ReduceFileSize

Description
Takes the active document and reduces file size.

Parameters
pReduceImageLevel [String, Optional]: Specifies the compression ratio. Possible values are Low, Medium and High, the default is Low.
pImageDPI [Number, Optional]: Specifies the Image's maximum DPI, the default is 150.
pDropFonts [Bool, Optional]: Specifies if Fonts will be dropped or not, the default is true.
pDropMiscellaneous [Bool, Optional]: Specifies if Metadata, Thumbnails, Private Data, Unused Resources will be dropped or not, the default is true.
pCompressAllStreams [Bool, Optional]: Compress All Streams, the default is true.

Example
ReduceFileSize()
ReduceFileSize("High", 150, true, true, true)

Repair

Description
Runs a repair process on the active document using the specified options.

Parameters
pFixStripedImages [Bool]: Groups neighboring image stripes into a single image
pCombineStripedImages [Bool]: Attemps to merge groups of thin adjacent images into one image
pOptimizeSolidColorImages [Bool]: Converts single color images into vector rectangles
pProcessMasks [Bool]: Fixes AutoCAD files with Blend Modes and Masks
pRemoveTextClipping [Bool]: Fixes AutoCAD files with text clipping problems
pSimplifyClippingPaths [Bool]: Fixes Revit files with text clipping problems
pRepairFontIssues [Bool]: Fixes issues with fonts

Example
Repair(true, true, true, true, true, true, true)

ReplacePages

Description

Replaces pages in the current document with pages from the source document.

Parameters
pSourceFileName [String]: PDF document to get pages from
pSourcePages [String]: List or range of all source pages to use, -1 will use all pages, exp: 1,2,10-20
pPagesToReplace [String]: List or range of pages to replace, -1 will replace all pages, exp: 1,2,10-20
pContentOnly [Bool, Optional]: If true only the page content witll be replaced leaving markups and hyperlinks

Example
```
ReplacePages("c:\Directory\test.pdf", "1", "1")
ReplacePages("c:\Directory\test.pdf", "3,6", "4,7", true)
```

ReversePages

Description
Reverses all pages in the document.

Parameters
None

Save

Description
If no parameters are specified it will save the file over itself. Otherwise it will save the file to the specified file location without changing the source file.

Parameters
pFileName [String, Optional]: Filename to save the file or directory to save file to using same filename.
pSaveMode [Number, Optional]: Save Mode as Follows:

0 = Incremental Updates
1 = Publish
2 = Publish Compressed

Example
```
Save()
Save("output.pdf")
Save("output.pdf", 1)
```

SaveAsPDFA

Description
Converts the current PDF document into a PDF/A-1b document.

Parameters
pFileName [String, Optional]: Full path or file name to the PDF/A-1b document being saved.

Example

SaveAsPDFA()
SaveAsPDFA("C:\pdfa\output.pdf")

Script

Description
Runs the script file specified as a parameter. Be careful to avoid infinite looping. Inside of a script file, arguments starting with arg0 ... can be used istead of fixed values. At run time, the arguments will be substitued with the passed in values. The Batch command relies on the arguments in order to dynamically run a script on a set of files. In older versions of the Script Engine, this command supported passing multiple scripts as parameters, that functionality no longer works in order to support the new argument functionality.

Parameters
pScriptPath [String]: Filename of script to run
Arg0 [String, Optional]: Argument parameter to pass to script, sets key arg0
ArgN [String, Optional, ...]: Argument parameter to pass to script, sets key argN

Example
Script("script1.bci")
Script("script1.bci", "arg0", "arg1", ...)

SetOpenPassword

Description
Sets open password on active document.

Parameters
pOpenPassword [String]: The open password need to open PDF
pEncryptionLevel [String, Optional]: Encryption Level to use. Values can be RC4, AES128 or AES256.

Example
SetOpenPassword("abacadabra")

SetPDFSecurity

Description
Applies security permissions to the active document.

Parameters
pPermissionPassword [String]: Password to lock pdf permissions
pFlags [Number]: Specifies what permission are allowed

Print = 1
PrintLowOnly = 2
FillForms = 4
EditMarkups = 8
EditDocument = 16
PageManipulation = 32
CopyContent = 64

Accessibility = 128

Add together values to set permissions

pOpenPassword [String, Optional]: Password used to open the pdf file
pEncryptionLevel [String, Optional]: Encryption Level to use. Values can be RC4, AES128 or AES256.

Example
SetPDFSecurity("master", 1)
SetPDFSecurity("master", 13, "open")

SplitPages

Description
Extracts all pages in page range to individual files.

Parameters
pPageRange [String]: List or range of pages to Extract, -1 will extract all pages, exp: 1,2,10-20
pDirectory [String]: Directory to save the extracted pages to
pUsePageLabels [Bool, Optional]: Use page labels to name extracted pages as pdf files.
pPageFormat [String, Optional]: Format to number files names for multiple pages, if pUsePageLables is true then this parameter will be ignored

Example
SplitPages("-1", "c:\Directory")
SplitPages("1,5,10", "c:\Directory", true)
SplitPages("1,5,10-20", "c:\Directory", false, " Page 001")

Stamp

Description
Places a stamp on the active document using the specified parameters.

Parameters
pFileName [String]: Filename of Stamp
pOrigin [String]: Origin of where to place the stamp as follows:

"upperleft"
"upperright"
"lowerleft"
"lowerright"
"center"
"uppercenter"
"lowercenter"

pXOffset [Number]: X Offset from origin in inches
pYOffset [Number]: Y Offset from origin in inches
pRotation [Number, Optional]: Rotation in Degrees
pScale [Number, Optional]: Scale (e.g. 0.5 would be 50%)
pOpacity [Number, Optional]: Opacity (0.4 would be 40% opacity)
pBlendMode [String, Optional]: Blend Mode as follows:

"normal"
"multiply"
"screen"
"overlay"
"darken"
"lighten"
"colordodge"
"colorburn"
"hardlight"
"softlight"
"difference"
"exclusion"
"luminosity"
"hue"
"saturation"
"color"

pPageRange [String, Optional]: List or range of pages to be stamped, -1 will stamp all pages, exp: 1,2,10-20
pLocked [Bool, Optional]: Specifies whether or not the stamp should be locked

Example
Stamp("mystamp.brx", "lowerright", 0.5, 1.0, 0, 1, 1, "normal")

Thumbnail

Description
Creates a thumbnail of given width and height and saves it to the specified filename. Can have an extension of most common image formats including (.bmp, .png, .jpg ...)

Parameters
pWidth [Number]: Desired width in pixels of output thumbnail image
pHeight [Number]: Desired height in pixels of output thumbnail image
pFileName [String]: Filename of desired output thumbnail image.
pPageFormat [String, Optional]: Suffix used when generatating thumbnails for multiple pages. " Page 001" would cause the resulting files to be named "File Page 001.png", "File Page 002.png" ...
pPageRange [String, Optional]: List or range of pages to have thumbnails generated for, -1 will generate thumbnails for all pages, exp: 1,2,10-20
pShowPopups [Bool, Optional]: Indicates that popups should be included

Example
thumbnail(320, 200, "thumbnail.png")

Unflatten

Description
Reverses the flattening process on the active document.

Parameters
pPageRange [String, Optional]: List or range of pages to unflatten, -1 will unflatten all pages, exp: 1,2,10-20

Example
Unflatten()

UnlockPDFA

Description
Unlocks the current PDF/A-1b document for editing.

Parameters
None

Example
UnlockPDFA()

UserNameGet

Description
Returns the user name to use when adding or modifying markups

Parameters
None

Example
UserNameGet()

UserNameSet

Description
Sets the user name to use when adding or modifying markups

Parameters
pUserName [String]: User name

Example
UserNameSet("Homer J. Simpson")

View

Description
Launches a file to be opened in the default viewing application. With no parameters specified the active document will be viewed, dirty document must be saved before calling view. Note that this is not limited to PDF files.

Parameters
pFileName [String, Optional]: Filename of file to view

Example
View()

View("document.pdf")