Technologies of Ohio, Inc.
Home of filePro * Development Software

432 W. Gypsy Lane Road
Bowling Green, OH 43402
Tel. (800) 847-4740
Fax (877) 606-6853
sales@fptech.com

Version 6.1.01 USP New Features

JSON Import and Export

filePro now has the ability to import and export JSON files.

Export:

```
JSON [id] :CR fname          - Creates a JSON file. The id is optional and
                               defaults to "0" if only one file is open at
                               a time. If two or more are open, the id
                               must be supplied ("0"-"99")
JSON [id] :CR-|:CL           - Closes an open JSON file.
JSON [id] :OB [name]         - Starts an object in a JSON file.
JSON [id] :OB-               - Closes an object.
JSON [id] :AR [name]         - Starts an array in a JSON file.
JSON [id] :AR-               - Closes an array in a JSON file.
JSON [id] :IT name [value]   - Adds an item to a JSON file, if a value is
                               not supplied, the resulting value will be
                               null.
JSON [id] :NO name [value]   - Adds a number to a JSON file, if a value is
                               not supplied, the resulting value will be
                               null.
JSON [id] :BL name [value]   - Adds a boolean value to a JSON file, if a
                               value is not supplied, the resulting value
                               will be null.
```

Note: Names will be ignored when adding an item, number, or boolean directly to an array.

Example:

```
JSON :CR "/tmp/myfile.json"
JSON :OB
JSON :OB "name"
JSON :IT "first" "Tom"
JSON :IT "last" "Anderson"
JSON :OB-
JSON :NO "age" "37"
JSON :AR "children"
JSON :IT "" "Sara"
JSON :IT "" "Alex"
JSON :IT "" "Jack"
JSON :AR-
JSON :IT "fav.movie" "Deer Hunter"
JSON :OB-
JSON :CL
```

Output:

```
    {
      "name":    {
        "first": "Tom",
        "last":    "Anderson"
      },
      "age": 37,
      "children": ["Sara", "Alex", "Jack"],
      "fav.movie": "Deer Hunter"
    }
```

Import:

JSON [id] :RO fname - Opens a JSON file for reading. The id is optional and defaults to "0" if only one file is open at a time. If two or more are open, the id must be supplied ("0"-"99")

value = JSON [id] :GV key - Get a value from a JSON file using a path to a key.

Keys are a way to reference part of a JSON document using dot syntax. An example of dot syntax would be a key, such as "name.first" or "age". There are reserved symbols used in key syntax that can be used to retrieve certain values from the JSON:

'#' is used to get the number of elements inside of an object or array.
'@' is used to specify a literal, or if at the end of the path, get the name of the current object.

Index positions can also be used to reference specific elements by numeric position inside of an object or an array. Indexes in Key Syntax start at position 1.

x = JSON :GV "fruits.10" will attempt to find the tenth (10) item inside a fruits object or array.

x = JSON :GV "fruits.@10" will attempt to find a key named "10" inside a fruits object and return its value.

Example:

Given the following JSON, here are example commands and what they return.

```
{
  "name":    {
    "first": "Tom",
    "last":    "Anderson"
  },
  "age": 37,
  "children": ["Sara", "Alex", "Jack"],
  "fav.movie": "Deer Hunter"
}
```

```
Then: JSON :RO "/tmp/myfile.json" ' open the JSON file for reading
Then: x=JSON :GV "name.first"     ' x contains "Tom"
Then: x=JSON :GV "name.1.@"       ' x contains "first"
Then: x=JSON :GV "age"            ' x contains "37"
Then: x=JSON :GV "children.#"     ' x contains "3"
Then: x=JSON :GV "children.1"     ' x contains "Sara"
Then: x=JSON :GV "fav\.movie"     ' x contains "Deer Hunter"
Then: JSON :CL                    ' close the JSON file
```

Fill-In-The-Blank PDFs

filePro now has the ability to place fill-in-the-blank PDF objects on output formats and also retrieve values from PDF documents that have fill-in-the-blank fields to be used in Processing.

There are four types of PDF Form Objects that can be used:

- **Textbox**
- **Dropdown**
- **Checkbox**
- **Radio**

When a PDF output is generated, placed objects will be interactive in any supporting PDF viewer/editor. These PDF files can be saved after filling in fields, and processing can be written to retrieve values from these fields.

Note: Using the new generation features in a report can lead to unintended results. Fields are shared across records and pages. Updating one field updates all matching instances of that field throughout the document. It is recommended to use output forms over output report

Please See Fill In PDFs in the manual for more information on document creation.

Manual Link

If the PDF was created with filePro, field names will be either the real-field or dummy field used to create the PDF object.
e.g. "1", "42", "aa", "ab".

Use these commands to read filled-in PDF documents:

handle = PDF_OPEN(pdf_path)
Returns a handle value (10,.0) that points to a PDF document with pdf_path as the filename. Returns a negative value on error.

error_value = PDF_CLOSE(handle)
Frees all values and memory associated with a PDF handle and closes the document. Returns a non-zero number on error.

num_fields = PDF_GETNUMFIELDS(handle)
Returns the number of fields in the PDF document.

name = PDF_GETFIELDNAME(handle, index)
Returns the full name of a field in a PDF document, given its index. The index is a number between "1" and the num_fields value returned by **PDF_GETNUMFIELDS**.

type = PDF_FIELDTYPE(handle, fieldname)
Returns the field type name of the specified field fieldname, which is one of:

- **NONE**
- **BUTTON**
- **RADIO**
- **CHECKBOX**
- **TEXT**
- **RICHTEXT**
- **CHOICE**
- **UNKNOWN**

type = PDF_FIELDTYPE2(handle, index)
Returns the field type name of the specified field index, which is one of:

- **NONE**
- **BUTTON**
- **RADIO**
- **CHECKBOX**
- **TEXT**
- **RICHTEXT**
- **CHOICE**
- **UNKNOWN**

The index is a number between "1" and the num_fields value returned by **PDF_GETNUMFIELDS.**

value = PDF_GETVALUE(handle, fieldname [, richtext])
Returns the field value, e.g. the text in the field, checkbox status, combo box index, etc. for the given field name fieldname. Optionally, richtext can be set to "1" to return rich text data if it exists.

value = PDF_GETVALUE2(handle, index [, richtext])
Returns the field value, e.g. the text in the field, checkbox status, combo box index, etc. for the given field index index. Optionally, richtext can be set to "1" to return rich text data if it exists. The index is a number between "1" and the num_fields value returned by **PDF_GETNUMFIELDS.**

QRCODE Command

ret = QRCODE(str, dest [, size [, logo [, fg [, bg]]]])
Create a QR Code from a text string.

str is the text to store in the QR code.

dest is the full name and path to the QR code to be generated.

size is the size of the QR code to be generated in pixels. Must be large enough to store the full QR code.

logo is an optional logo to place in the center of the QR code.

fg is the foreground color of the QR code in hexadecimal.

bg is the background color of the QR code in hexadecimal.

Returns the size of the generated QR code, or -1 on error.

Example:
Then: ret=QRCODE("fptech.com","/tmp/website.png")

QRCODE print code

<QRCODE TEXT="qr text" [SIZE="size"] [COLOR="color"] [FILL="bg color"] [X="x-pos"] [Y="y-pos"]>

Adds a QR code with the specified text to the PDF document.

All attributes, except for **"TEXT"**, are optional.

TEXT is the text to add to the QR code when generating the image.

SIZE is the width and height of the QR code, must be large enough to fit the entire generated image.

COLOR is the foreground color of the QR code (in hexadecimal).

FILL is the background color of the QR code (in hexadecimal).

X X position. (Default: current X position.)

Y Y position. (Default: current Y position.)

FPML Print Code Enhancements

FPML print codes can now use field names for any attribute.

Any attribute inside of an FPML print code can now reference a real field or variable inside of processing. Use "@" to reference a field.

e.g.

```
<IMAGE FILE="@1">            ' reference a real field
<IMAGE FILE="@im">           ' reference a dummy field
<IMAGE FILE="@image_path">   ' reference a long name variable
```

Note: Print codes can also be stored in a print code table and do not need to be placed directly on the output to work.

Array Commands and Enhancements

Added initial support for multi-dimensional arrays.

DIM array[n1,n2,...,n8](l,e)
Multi-Dimensional array of fields with length "l" & edit "e". Array edit is optional.

Example:

```
dim array(2,2)
array["1","1"]="John"
array["1","2"]="Smith"
array["2","1"]="Sarah"
array["2","2"]="Jane"
```

Existing array functions can also use multi-dimensional arrays by referencing one of an array's sub arrays.

Example:

```
CLEAR array["1"]
```

subscript = INDEXOF(array, value)
Find the subscript of some value in an array.

Example:

```
array["1"]="cat"
array["2"]="dog"
array["3"]="bird"

subscript = INDEXOF(array, "dog") ' subscript will contain "2"
```

value = A_MAX(array [, array2 [, array3 [, ... [, arrayN]]]])
Find the maximum value between the passed in arrays.

Example:

```
array1["1"]="5"
array1["2"]="7"
array2["1"]="30"
value = A_MAX(array1, array2) ' value will contain "30"
```

Note: This method supports multi-dimensional arrays.

value = A_MIN(array [, array2 [, array3 [, ... [, arrayN]]]])
Find the minimum value between the passed in arrays.

Example:

```
array1["1"]="5"
array1["2"]="7"
array2["1"]="30"
value = A_MIN(array1, array2) ' value will contain "5"
```

Note: This method supports multi-dimensional arrays.

value = A_TOT(array [, array2 [, array3 [, ... [, arrayN]]]])>
Total all of the values in the passed in arrays.

Example:

```
array1["1"]="5"
array1["2"]="7"
array2["1"]="30"
value = A_TOT(array1, array2) ' value will contain "42"
```

Note: This method supports multi-dimensional arrays.

value = A_AVG(array [, array2 [, array3 [, ... [, arrayN]]]])
Find the avereage of all of the values in the passed in arrays.

Example:

```
array1["1"]="5"
array1["2"]="7"
array2["1"]="30"
value = A_AVG(array1, array2) ' value will contain "14"
```

Note: This method supports multi-dimensional arrays.

DECLARE Enhancement
Added the ability to assign directly to a longvar when creating it.

Example:

```
DECLARE my_var(32,*)="Hello, World!"
```

Runtime Engine

Reworked tokenization engine to no longer require setting PFTOKSIZE or related variables. Variable will now be silently ignored.

Define Processing

Added a new F5 shortcut in Define Processing for calls. F5 will now open a call for editing, or, will prompt you to create the call if it does not exist.

Debugging

New stacktrace option.

Added a new option **T** to the debugger to display a stacktrace.