Release notes - filePro Plus 6.1 - 05/17/2024
    fP 6.1.XX.05

PLEASE NOTE the support and fax phone numbers listed in this
readme file. Open new support incidents on our website.

****

WWW   http://www.fptech.com
Support   support@fptech.com
Sales   sales@fptech.com
Management filepro@fptech.com

****

To submit bug reports
---
1. Login to your account portal on our website
   http://www.fptech.com/fptech/login.php and then
   go to the Support Incident Menu and submit an
   incident request.
2. EMail them to support@fptech.com including the text
   "Bug Report" with the version # and your filePro
   License # in the subject line
3. FAX them to (813) 354-2722 clearly marking them as bug reports
   and be sure to reference your filPro License #
4. Call the customer support number (800) 847-4740

****

A special thank you to Jim Asman for his contribution
to the functionality of our printer tables.   Jim was
a good friend to filePro and is dearly missed.

****

Contact Information

Surface Mail
    fP Technologies, Inc.
    432 W. Gypsy Lane Road
    Bowling Green, OH 43402

Phones
    Support   (800) 847-4740
    Sales   (800) 847-4740
    Fax   (813) 354-2722

Email
    Support   support@fptech.com
    Sales   sales@fptech.com
    Management filepro@fptech.com

It's important that you clearly describe a suspected bug and
include the filePro version number. If the programmer has trouble
figuring out what you meant, you might as well not have reported
the bug. Be very specific. For example, if you are reporting a
bug concerning a Browse, identify if it is a lookup browse or
browse created by using the [F6] key. A screen shot is very
helpful and sometimes better than more than 1000 words.

Describe exactly how to duplicate the bug. Although it's
sometimes difficult to create a working sample to demonstrate the
problem, make every effort to trim down your code and provide a
working sample application with test data. You may even discover
that what you thought to be a bug is due to a coding error or the
bug may only occur with lots of data or large processing tables.

Take good notes as to any error messages and under what
circumstances the error message is presented. It never hurts to

provide more information rather than not enough. This is
particularly true when the programmer asks for additional
information. Rather than responding with a single sentence, be
verbose since this may shed some light on the bug or what you may
be doing wrong in your code.

Read what you wrote. Closely read your bug report before submitting
to make sure it's clear and complete. If you have listed steps for
duplicating the bug in a sample, exercise the sample with the
listed steps to make sure you haven't missed a step.

****
filePro and filePro Plus are registered
  trademarks of fP Technologies, Inc.
****
==========
Bug fixes are below the New Items.
==========
==========
Version 6.1.01.RR New USP Only Items
==========

Added JSON import and export code.
    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


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

    name = 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.


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")


Added QRCODE FPML 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 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.


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.


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"


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"]


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.
==================================
END OF NEW USP ITEMS
==================================
==================================
6.1.XX.04 NEW ITEMS
==================================
Added PFOLDCHAIN to allow CHAIN to return to the top of processing when a record
    is saved and the chain was performed inside of an event.

Added basic reconnect functionality into ODBC mirroring upon communications link
    failure.

Added the ability to directly assign to a longvar when declaring it.
    e.g.
    declare myvar = "Hello!"

Updated Fuzzy search screen in clerk to be larger and show correct button
    prompts.

n = STACKTRACE(array)
    Fill an array with a processing trace, listing the current and past
    processing tables and their line numbers to the current line being executed.
    This will show lines "jumped" from gosubs and follow calls and functions.

    Returns the number of elements that could fit into the array.
```

Added new debugger option "T" to show the current stacktrace while debugging.

==========
6.1.00.03 NEW ITEMS
==========
Updated all programs to no longer require unixODBC by default. unixODBC will now
    only be required when an ODBC related function is used. If unixODBC is not
    found when an ODBC function is required, a filePro error will be returned.

Added the ability to assign directly to a longvar when creating it.
    e.g.
    declare myvariable(32,*)="Hello, World!"

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

Added PFPDFAUTOBREAK=ON (default OFF) to allow PDFs to automatically break pages
    based off of selected paper type.

Added menu letter to menu script editor.

==========
6.1.00.00 NEW ITEMS
==========
You can now use: @wlf<letter>*
    ex. @wlfT*
    This will apply to any dummy/associated field that begins with 'T'
    Overrides any other @wlf*

Added logging to ddefine.
    ddefine can now optionally track changes made to filePro file
    layouts. This includes the name of the file, who changed it,
    and what fields were changed. Requires a logging configuration
    file to be added under the ./fp/logs directory named 'ddefine.cfg'.
    Format of the config file is the same as the servlog.cfg file that
    comes shipped with filePro.
    Example ddefine.cfg:
        ROLLING,DEBUG,ddefine.log,60000

xx=FORMERROR
    syntax: xx=FORMERROR()
    returns: errno from last FORM or FORMM command.
    e.g. 2=file not found, 13=permission error

Validate menu script before prompting for removal

Added new option 'C' to F8 Extended Functions for dmoedef
    to show a list of all print codes on an output format. Selecting
    an item from the list will jump the editor to it.

TRIM command to remove spaces
    aa=ltrim(fld)
        left trim
    aa=rtrim(fld)
        right trim
    aa=trim(fld)
        trim both left and right

PFIXGT can now be set in dxmaint F8 options.
    This is backwards compatible, so if PFIXGT is still set in config,
    then it is honored by clerk *if true*. If false, the index header
    is checked for the flag.

Windows fPTransfer now will accept wildcards.

A compress-filePro file routine
    fppack

    Function:
     Remove deleted records from a filePro file, and then
     (optionally) rebuild all automatic indexes.

     Syntax:
      fppack [filename | -] [-H heading] [-E] [-R] [-X] [-EX] [-C]

```
      [ -M name | -MD | -MQ mesg | -MA ] [ -BG ] [ -BS ]

       -H "heading" custom title to display in box.
       -E           don't actually pack the records, just
                       give statistics.
       -R           rebuild the automatic indexes even if
                       no records were deleted.
       -EX          skip statistics
       -C           skip continue and finished prompts

       -X           skip rebuilding the auto indexes.
       -M name      qualifier file name to use.
       -MD          ask for qualifier with default prompt.
       -MQ "mesg"   ask for qualifier with "mesg" as the prompt.
       -MA          use all qualified files & main file.
      UNIX/XENIX only:
       -BG          work in the background.
       -BS          suppress "completed in background" message.

Added various enhancements to PDF engine.
    See on-line or ~/fp/docs PDF documentation.

Added optional error message suppression and basic password
    auditing to filePro.

        PFERRSUPPRESS=ON, default OFF
        PFPWAUDIT=ON, default OFF

    Password auditing also requires a ./fp/logs/pwaudit.cfg file. Same
    structure as servlog.cfg.
    Any error that would be sent to mail will still be mailed on
    unix/linux based systems.
    Errors reported in the background will still be suppressed.
    Including the program name.
    Invalid password and license errors will still be reported. Password errors
    omit the filename.
        dcabe and rcabe are exempt from the error suppression.

These functions lock or unlock bytes of the file specified by handle.

    x=lock(handle,how[,nbyte])
        handle - an open handle to a file
        how    - U|0 : unlock bytes
                 L|1 : lock bytes
                 N|2 : lock bytes non-blocking
        nbyte  - How many bytes in the file to lock, if omitted, lock
                 the billionth byte in the file (file does not have to be
                 that large)

    x=unlock(handle[,nbyte])
        handle - an open handle to a file
        nbyte  - How many bytes in the file to unlock, if omitted,
                 unlock the billionth byte in the file (file does not
                 have to be that large)

    (returns "1" on success and returns negated system error on error)


ddefine will now create new screens the same as dscreen does instead of just
    mono.

NEW command OPENDIR2 to handle long-named files and paths.
    Syntax:
        N = OPENDIR2(mask, path, fmt_sz, ext_sz, nam_sz)
        All arguments are optional.
            Format Length
            Extension Length
            Fullname Length

*cabe lookup wizard will now honor PFQUAL and show qualified indexes

Added new FPML commands to control the appearance of underlines. (See PDF Docs)

New RINSTR, and INSTR now allows negative positions for working backwards.
```

New GIadmin that will count GUI (GI or Web) sessions, ease of system
    and user configuration files and additional security.

Added PDF syntax as an option for printer maintenance (pmaint): Windows only

Lookup Wizard in cabe now allows long vars as key.

Added alias and arrays to F6-D-L display in *cabe.

Updated color with new routines and corrected the shell escape codes.

Automated processing table backups.
    CABEBACKUP ON|OFF (on by default)
    CABEBACKUPMINS n (minutes between backups)
    CABEBACKUPCT n (backup files per process)

Menu maintenance (makemenu) now asks if you wish to remove
    an unused menu script if the menu item is not used.

*report now allows one to use .outs from a pathed directory library

SCREEN command can switch fields in a POPUP UPDATE -, provided no screen name
    is passed to the SCREEN command.

MEMO EDIT now accept maxsize to limit the number of
    characters that can be intered into a memo field.
    memo NNN edit (row,col,lines,width,startLine,startcol,maxSize)
    (Text mode only)

Added option 7 to dxmaint to clear qualifier

New -SE *report flag to allow report to edit/save a selection set.

Added @EXIT label to *clerk processing. This is executed whenever
    a record is exited or broken out of. Events that trigger this are
    'X' while not in update mode, 'BRKY' while not in update mode, and
    'exit' in processing. It is the opposite of @entsel, and is the last
    thing executed when leaving a record. Assignment of real fields is
    not allowed, this is similar to @once in that the processing that is
    executed is NOT sitting on a record, but rather record '0'.

Partial lookup flag added to *cabe lookup wizard.
    -O on an exact lookup now does partial key matching. This kills a
    lookup once the begining of the key value no longer matches the lookup
    key value.

BUSYBOX
        BUSYBOX "my message"
        BUSYBOX("10","10")
        BUSYBOX("10","10") "my message"

Added PFPPFULLPATH as an ehancement to PDFPOSTPRINT
    and added an PFNEWPOSTPRINT alias to name to PDFPOSTPRINT
    Added PFPPFULLPATH to augment the filename passed to the post print
    handler, default ON, this causes the filename passed to the postprint
    script to contain the full path to the file, not just the file name.
    Set to OFF to revert to old behaviour. PFPOSTPRINTnnn will now work
    with normal file destinations. Same rules as the old global PFPOSTPRINT
    but also supports PDF files.

Clerk will now allow a full path to a form when using
    the FORM and FORMM command in processing.

User defined functions
    Forward declare functions to be used:
    (function|func) [file.]name([dim|var] var1, [dim|var] var2, ...)

    e.g.
         function fplib.showlock(var pid)
         function fplib.log(file, line, what)
         function somefunc(dim myarray)

    Call a function:
         [x=][file.]name(var1, var2, ...)

```
    Return a value from a function:
         return(value)

    Can pass fields: real, dummy, longvar
    Can pass arrays: Alias and system arrays are copied to a non-aliased
    array. Non-aliased arrays are passed by reference.

    Function names must be at least 3 characters in length.
    Functions cannot modify values outside of its scope.
    Functions do not call automatic processing.
    Functions cannot modify real fields.
    Functions cannot be called unless it they are declared.
    Functions can pass values by reference (changes made to the value will
    carry back out of the function, only to arrays).
    Functions can optionally return a value.

    Parameter names must be at least 3 characters in length.
    Parameters will be passed to the function using the name they were defined
    with in the declaration statement.

    Environment variables:
        PFFUNCDBG=(ON|OFF), default OFF.
            If ON the debugger will be allowed to continue into the function
            call. If OFF the debugger will skip over user defined functions.
            NOTE: Debug statements inside of functions will still be able to
            be activated. If debug is set inside of a function, it will
            continue even after the function is left.

    Example:
    Processing table for fibonacci:
        If:                     ' Declare for future use
        Then: function fibonacci(nval)
          If:                     ' Get the parameter
        Then: declare extern nval
          If: nval le "1"         ' Return the result
        Then: return(nval)
          If:                     ' Return the result
        Then: return(fibonacci(nval-"1")+fibonacci(nval-"2"))
    Usage:
        If:                     ' Declare for future use
        Then: function fibonacci(nval)
        If:                     ' Call the function
        Then: n=fibonacci("9")
        If:                     ' Display the result
        Then: msgbox ""{n         ' Prints "34"

EXTERN and GLOBAL arrays
    DIM GLOBAL name(size)
    DIM EXTERN name

    Only non-aliased arrays can be declared GLOBAL/EXTERN.
    Functions similar to GLOBAL/EXTERN longvars.

New compare condition for Associated Fields
    Added new selection set relational operators:
        AEQ - Associated field, all equal
        ANE - Associated field, all not equal
        ACO - Associated field, all contain
    These require ALL components of an associated field to match the
    comparison being done, rather than just one of its component fields.

New functions for creating XLSX documents from filePro.
e = XL_OPEN(file [, name])
    Start building an XLSX output file.

    Parameters -
        file : Path to the file to create. If no full path is given the
               generated file will be placed in the PFTMP or equivalent
               directory.
        name : The name for the default sheet that will be created. Defaults to
               Sheet1.

    If the filename does not end in ".xlsx" it will be added on creation.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
```

```
    the last error.

    Note: Only one XLSX file can be created at a time.


e = XL_SAVE([password])
    Save the current XLSX file.

    Parameters -
        password : If specified, encrypt the XLSX output file using Agile
                   encryption (AES128).

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.

    Note: Encrypted XLSX files cannot be opened with most third party programs
    such as LibreOffice and OpenOffice. They are fully supported by Excel
    however. The documents are saved in an encrypted CFB file.


handle = XL_ADDSHEET([name])
    Add a new sheet to the XLSX document.

    Parameters -
        name : The name for the sheet to be created. Defaults to auto naming the
               sheet based on the Sheet1, Sheet2, ..., SheetN template.

    Returns a handle to a new sheet object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.


e = XL_ADDCELL([data [, style [, sheet [, row [, col]]]]])
    Add a new cell to the XLSX document.

    Parameters -
        data  : Data to be inserted into the document. A cell starting with '='
                will be treated as a formula.
        style : Handle to style to be used for this cell. Use blank to use the
                default style.
        sheet : Handle to sheet to insert the cell on. Use blank, "0", or "-1"
                to use the default sheet.
        row   : Row to place the cell (0 indexed).
        col   : Column to place the cell (0 indexed).


    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.

    Note: Using an empty or negative row/column value will cause the cell to be
    added using the auto counter in the sheet, incrementing the column value
    after the cell is added. Specifying a location will reposition the auto
    counter. Formulas can be used as part of the data as well by prefixing the
    string with '='.


e = XL_ADDCELL2([data [, style [, sheet [, cell]]]])
    Add a new cell to the XLSX document.

    Parameters -
        data  : Data to be inserted into the document. A cell starting with '='
                will be treated as a formula.
        style : Handle to style to be used for this cell. Use blank to use the
                default style.
        sheet : Handle to sheet to insert the cell on. Use blank, "0", or "-1"
                to use the default sheet.
        cell  : The Excel style cell to insert the cell. e.g. "A1" "D6" "F6".

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.

    Note: Using an empty cell number will cause the cell to be added using the
    auto counter in the sheet, incrementing the column value after the cell is
    added. Specifying a location will reposition the auto counter. Formulas can
    be used as part of the data as well by prefixing the string with '='.
```

```
handle = XL_FORMAT(format)
    Create a new format to use with the XLSX document.

    Parameters -
        format : Excel format string to use to format the a style. e.g.
                 "$ #,###,nnn.nn"
                 "% ##n.n"
                 "m/d/yyyy"

    Returns a handle to a new format object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.


e = XL_COLWIDTH(width, firstcol, lastcol [, sheet])
    Change the default column width for a sheet between a range.

    Parameters -
        width    : Width of the column(s). e.g. "24" "12.5", "11"
        firstcol : Zero based column index or column letter to set from.
        lastcol  : Zero based column index or column letter to set to.
        sheet    : Handle to sheet to change the cell widths.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


handle = XL_FONT(font, [size [, attr [, color]]])
    Create a new font to use with the XLSX document.

    Parameters -
        font  : Name of the font to use.
        size  : Point size of the font. e.g. "11" "8.42" "12", default "11.0"
        attr  : List of attributes to apply to this font, separated by commas.
                e.g. "bold,italic"
                Values:
                    "bold"
                    "italic"
                    "underline"
                    "strike"
                    "unlocked"
                    "hidden"
                    "wrap"
                    "shrink"
                    "fill"
                    "left"
                    "center"
                    "right"
                    "justify"
                    "top"
                    "bottom"
                    "vjustify"
                    "vcenter"
        color : The RGB Hex value to set the font color.
                e.g. "000000" "ADD8E6"

    Returns a handle to a new font object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.


handle = XL_BORDER(borderstyle [, color])
    Create a new border to use with the XLSX document.

    Parameters -
        borderstyle : The style to use with this border. Must be one of the
                      following values:
                      "thin"
                      "medium"
                      "dashed"
                      "dotted"
                      "thick"
                      "hair"
                      "medium_dashed"
                      "dash_dot"
                      "medium_dash_dot"
```

```
                      "dash_dot_dot"
                      "medium_dash_dot_dot"
                      "slant_dash_dot"
        color       : The RGB Hex value to set the border color.
                      e.g. "000000" "ADD8E6"

    Returns a handle to a new border object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.


handle = XL_FILL(bg [, fg [, fill]])
    Create a new fill to use with the XLSX document.

    Parameters -
        bg   : The RGB Hex value to set the background fill color.
               e.g. "000000" "ADD8E6"
        fg   : The RGB Hex value to set the foreground fill color.
               e.g. "000000" "ADD8E6"
        fill : The fill pattern to use, defaults to "solid" fill. Value must be
               one of the following.
               "solid"
               "medium_gray"
               "dark_gray"
               "light_gray"
               "dark_horizontal"
               "dark_vertical"
               "dark_down"
               "dark_up"
               "dark_grid"
               "dark_trellis"
               "light_horizontal"
               "light_vertical"
               "light_down"
               "light_up"
               "light_grid"
               "light_trellis"
               "gray_125"
               "gray_0625"

    Returns a handle to a new fill object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.


e = XL_ADD_DT(date, time [, style [, sheet [, row [, col]]]])
    Combine two fields into a single spreadsheet datetime field and insert it as
    a new cell in the XLSX document.

    Parameters -
        date  : filePro date field.
        time  : filePro time field.
        style : Handle to style to be used for this cell. Use blank to use the
                default style.
        sheet : Handle to sheet to insert the cell on. Use blank, "0", or "-1"
                to use the default sheet.
        row   : Row to place the cell (0 indexed).
        col   : Column to place the cell (0 indexed).

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_ADD_DT2(date, time [, style [, sheet [, cell]]])
    Combine two fields into a single spreadsheet datetime field and insert it as
    a new cell in the XLSX document.

    Parameters -
        date  : filePro date field.
        time  : filePro time field.
        style : Handle to style to be used for this cell. Use blank to use the
                default style.
        sheet : Handle to sheet to insert the cell on. Use blank, "0", or "-1"
                to use the default sheet.
        cell  : The Excel style cell to insert the cell. e.g. "A1" "D6" "F6".

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
```

```
    the last error.


handle = XL_CHART(type [, title [, xname [, yname [, row [, col [, stylenum
                 [, sheet [, xoff [, yoff [, xscale [, yscale]]]]]]]]]])
    Add a new chart to the XLSX document.

    Parameters -
        type     : Type of chart to create. Must be one of the following values.
                   "area"
                   "area_stacked"
                   "area_stacked_percent"
                   "bar"
                   "bar_stacked"
                   "bar_stacked_percent"
                   "column"
                   "column_stacked"
                   "column_stacked_percent"
                   "doughnut"
                   "line"
                   "line_stacked"
                   "line_stacked_percent"
                   "pie"
                   "scatter"
                   "scatter_straight"
                   "scatter_stright_markers"
                   "scatter_smooth"
                   "scatter_smooth_markers"
                   "radar"
                   "radar_with_markers"
                   "radar_filled"
        title    : The title for this chart.
        xname    : The title for the x-axis.
        yname    : The title for the y-axis.
        row      : Row to place the cell (0 indexed).
        col      : Column to place the cell (0 indexed).
        stylenum : Number of the built in Excel style to use. Must be between
                   "1" and "48". The default style is 2. The value is one of
                   the 48 built-in styles available on the "Design" tab in
                   Excel 2007.
        sheet    : Handle to sheet to insert the chart on. Use blank, "0", or
                   "-1" to use the default sheet.
        xoff     : X axis offset to place the chart, in pixels.
        yoff     : Y axis offset to place the chart, in pixesl.
        xscale   : Scale the chart along the x axis. e.g. "1", "0.5" "2". Value
                   cannot be negative.
        yscale   : Scale the chart along the x axis. e.g. "1", "0.5" "2". Value
                   cannot be negative.

    Returns a handle to a new chart object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.

    Note: The chart functions do not use the auto counter found in the sheets
    and instead will default to "0", "0" or "A1" when used for insertion.


handle = XL_CHART2(type [, title [, xname [, yname [, cell [, stylenum [, sheet
                  [, xoff [, yoff [, xscale [, yscale]]]]]]]]]])
    Add a new chart to the XLSX document.

    Parameters -
        type     : Type of chart to create. Must be one of the following values.
                   "area"
                   "area_stacked"
                   "area_stacked_percent"
                   "bar"
                   "bar_stacked"
                   "bar_stacked_percent"
                   "column"
                   "column_stacked"
                   "column_stacked_percent"
                   "doughnut"
                   "line"
                   "line_stacked"
                   "line_stacked_percent"
```

```
                    "pie"
                    "scatter"
                    "scatter_straight"
                    "scatter_stright_markers"
                    "scatter_smooth"
                    "scatter_smooth_markers"
                    "radar"
                    "radar_with_markers"
                    "radar_filled"
        title    : The title for this chart.
        xname    : The title for the x-axis.
        yname    : The title for the y-axis.
        cell     : The Excel style cell to insert the cell. e.g. "A1" "D6" "F6".
        stylenum : Number of the built in Excel style to use. Must be between
                   "1" and "48". The default style is 2. The value is one of
                   the 48 built-in styles available on the "Design" tab in
                   Excel 2007.
        sheet    : Handle to sheet to insert the chart on. Use blank, "0", or
                   "-1" to use the default sheet.
        xoff     : X axis offset to place the chart, in pixels.
        yoff     : Y axis offset to place the chart, in pixesl.
        xscale   : Scale the chart along the x axis. e.g. "1", "0.5" "2". Value
                   cannot be negative.
        yscale   : Scale the chart along the x axis. e.g. "1", "0.5" "2". Value
                   cannot be negative.

    Returns a handle to a new chart object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.

    Note: The chart functions do not use the auto counter found in the sheets
    and instead will default to "0", "0" or "A1" when used for insertion.


handle = XL_CHARTSHEET(type [, title [, xname [, yname [, stylenum]]]])
    Add a new chartsheet to the XLSX document. A chartsheet is a full chart that
    occupies it's own sheet and cannot contain any cells.

    Parameters -
        type     : Type of chart to create. Must be one of the following values.
                   "area"
                   "area_stacked"
                   "area_stacked_percent"
                   "bar"
                   "bar_stacked"
                   "bar_stacked_percent"
                   "column"
                   "column_stacked"
                   "column_stacked_percent"
                   "doughnut"
                   "line"
                   "line_stacked"
                   "line_stacked_percent"
                   "pie"
                   "scatter"
                   "scatter_straight"
                   "scatter_stright_markers"
                   "scatter_smooth"
                   "scatter_smooth_markers"
                   "radar"
                   "radar_with_markers"
                   "radar_filled"
        title    : The title for this chart.
        xname    : The title for the x-axis.
        yname    : The title for the y-axis.
        stylenum : Number of the built in Excel style to use. Must be between
                   "1" and "48". The default style is 2. The value is one of
                   the 48 built-in styles available on the "Design" tab in
                   Excel 2007.

    Returns a handle to a new chartsheet object on successs and "-1" on
    error. XL_ERROR() can be called to return the last error.


e = XL_SERIES(chartnum, sheet, namerow, namecol, cfirstrow, cfirstcol, clastrow,
              clastcol, vfirstrow, vfirstcol, vlastrow, vlastcol)
```

```
    Add a series to a chart or chartsheet.

    Parameters -
        chartnum  : Handle to a chart or chartsheet to add series.
        sheet     : Handle to sheet to get values from. Use blank, "0", or "-1"
                    to use the default sheet.
        namerow   : Series name row (0 indexed).
        namecol   : Series name column (0 indexed).
        cfirstrow : Categories first row (0 indexed).
        cfirstcol : Categories first column (0 indexed).
        clastrow  : Categories last row (0 indexed).
        clastcol  : Categories last column (0 indexed).
        vfirstrow : Values first row (0 indexed).
        vfirstcol : Values first column (0 indexed).
        vlastrow  : Values last row (0 indexed).
        vlastcol  : Values last column (0 indexed).

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_SERIES2(chartnum, sheet, namecell, cfirst, clast, vfirst, vlast)
    Add a series to a chart or chartsheet.

    Parameters -
        chartnum : Handle to a chart or chartsheet to add series.
        sheet    : Handle to sheet to get values from. Use blank, "0", or "-1"
                   to use the default sheet.
        namecell : Series name Excel style cell. e.g. "A1" "D6" "F6".
        cfirst   : Categories first Excel style cell. e.g. "A1" "D6" "F6".
        clast    : Categories last Excel style cell. e.g. "A1" "D6" "F6".
        vfirst   : Values first Excel style cell. e.g. "A1" "D6" "F6".
        vlast    : Values last Excel style cell. e.g. "A1" "D6" "F6".

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_PROTECTSHEET(sheet, password)
    Add a password to restrict editing of a sheet.

    Parameters -
        sheet    : Handle to sheet to protect. Use blank, "0", or "-1" to use
                   the default sheet.
        password : Password to use to protect this sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_PROTECTCHARTSHEET(cs, password)
    Add a password to restrict editing of a chartsheet.

    Parameters -
        cs       : Handle to chartsheet protect.
        password : Password to use to protect this sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_ERROR()
    Return the last error generated by the XLSX set of functions.

    Returns the last error string generated by the XLSX engine.


e = XL_SETPOS(row [, col [, sheet]])
    Set the auto counter position for a sheet.

    Parameters -
        row   : Row to move auto counter to (0 indexed).
        col   : Column to move auto counter to (0 indexed).
        sheet : Handle of sheet to set. Use blank, "0", or "-1" to use the
                default sheet.
```

```
    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_SETPOS2(cell [, sheet])
    Set the auto counter position for a sheet.

    Parameters -
        cell  : Excel style cell to set the auto counter to. e.g. "A1" "D6".
        sheet : Handle of sheet to set. Use blank, "0", or "-1" to use the
                default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_NEXTROW([sheet])
    Move the auto counter down a row for a sheet.

    Parameters -
        sheet : Handle of sheet to set. Use blank, "0", or "-1" to use the
                default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_NEXTCOL([sheet])
    Move the auto counter one column for a sheet.

    Parameters -
        sheet : Handle of sheet to set. Use blank, "0", or "-1" to use the
                default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


handle = XL_STYLE([font [, fill [, fmt [, btop [, bbot [, bleft
                  [, bright]]]]]]])
    Add a new style to the XLSX document.

    Parameters -
        font   : Handle to font object to use.
        fill   : Handle to fill object to use.
        fmt    : Handle to format object to use.
        btop   : Handle to border object to use for top border.
        bbot   : Handle to border object to use for bottom border.
        bleft  : Handle to border object to use for left border.
        bright : Handle to border object to use for right border.

    Returns a handle to a new style object on successs and "-1" on error.
    XL_ERROR() can be called to return the last error.


e = XL_IMAGE(img [, row [, col [, sheet [, xoff [, yoff [, scalex [, scaley
           [, flag]]]]]]]])
    Add a new image to the XLSX document.

    Parameters -
        img    : Path to image file to use.
        row    : Row to insert the image on (0 indexed).
        col    : Column to insert the image on (0 indexed).
        sheet  : Handle of sheet to insert image. Use blank, "0", or "-1" to use
                 the default sheet.
        xoff   : X-axis offset for the image, in pixels.
        yoff   : Y-axis offset for the image, in pixels.
        scalex : Scale the image along the x-axis. e.g. "1", "0.5" "2". Value
                 cannot be negative.
        scaley : Scale the image along the y-axis. e.g. "1", "0.5" "2". Value
                 cannot be negative.
        flag   : Option of how to position image.
                 "0" - Default positioning.
                 "1" - Move and size image with the cells.
```

```
                "2" - Move but don't size image with the cells.
                "3" - Don't move or size the image with the cells.
                "4" - Same as "1" but wait to apply hidden cells.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.

    Note: The image functions only support PNG, JPEG, and BMP files.


e = XL_IMAGE2(img [, cell [, sheet [, xoff [, yoff [, scalex [, scaley
            [, flag]]]]]]]);
    Add a new image to the XLSX document.

    Parameters -
        img    : Path to image file to use.
        cell   : Excel style cell to insert the image. e.g. "A1" "D6" "F6".
        sheet  : Handle of sheet to insert image. Use blank, "0", or "-1" to use
                 the default sheet.
        xoff   : X-axis offset for the image, in pixels.
        yoff   : Y-axis offset for the image, in pixels.
        scalex : Scale the image along the x-axis. e.g. "1", "0.5" "2". Value
                 cannot be negative.
        scaley : Scale the image along the y-axis. e.g. "1", "0.5" "2". Value
                 cannot be negative.
        flag   : Option of how to position image.
                 "0" - Default positioning.
                 "1" - Move and size image with the cells.
                 "2" - Move but don't size image with the cells.
                 "3" - Don't move or size the image with the cells.
                 "4" - Same as "1" but wait to apply hidden cells.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.

    Note: The image functions only support PNG, JPEG, and BMP files.

e = XL_LASTCMD()
    Get debug information about the last XLSX call.

    Returns the last evaluated command parse string.


e = XL_MARGINS([left, [right, [top, [bottom, [sheet]]]]])
    Set the worksheet print margins.

    Parameters -
        left   : Left margin in inches, e.g. "0.5", "1", "0.75". A blank or
                 negative value will use the default of "0.7".
        right  : Right margin in inches, e.g. "0.5", "1", "0.75". A blank or
                 negative value will use the default of "0.7".
        top    : Top margin in inches, e.g. "0.5", "1", "0.75". A blank or
                 negative value will use the default of "0.75".
        bottom : Bottom margin in inches, e.g. "0.5", "1", "0.75". A blank or
                 negative value will use the default of "0.75".
        sheet  : Handle of sheet to set the margins. Use blank, "0", or "-1" to
                 use the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_LANDSCAPE([sheet])
    Set the worksheet to print in landscape mode.

    Parameters -
        sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_PORTRAIT([sheet])
    Set the worksheet to print in portrait mode.
```

```
    Parameters -
        sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_GRIDLINES(option [, sheet])
    Set if the worksheet should display gridlines when printed.

    Parameters -
        option : Which Gridlines to print. Cannot be blank. Must be one of the
                 following values.
                 "hide_all"
                 "show_all"
                 "show_screen"
                 "show_print"
        sheet  : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                 the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_FITPAGES([height, [width, [sheet]]])
    Fit the printed area to a specific number of pages both vertically and
    horizontally.

    Parameters -
        height : Number of pages vertically. A value of "0" or blank will set
                 the height as necessary.
        width  : Number of pages horizontally. A value of "0" or blank will set
                 the height as necessary.
        sheet  : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                 the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_PAPERTYPE(type [, sheet])
    Set the paper format for the printed output of a worksheet.

    Parameters -
        type  : The paper format to use with a printed worksheet. Must be one of
                the following values.
                "default"
                "letter"
                "tabloid"
                "ledger"
                "legal"
                "statement"
                "executive"
                "a3"
                "a4"
                "a5"
                "b4"
                "b5"
                "folio"
                "quarto"
                "10x14"
                "11x17"
                "note"
                "envelope"
                "envelope_9"
                "envelope_10"
                "envelope_11"
                "envelope_12"
                "envelope_14"
                "c"
                "d"
                "e"
                "envelope_dl"
```

```
                "envelope_c3"
                "envelope_c4"
                "envelope_c5"
                "envelope_c6"
                "envelope_c65"
                "envelope_b4"
                "envelope_b5"
                "envelope_b6"
                "monarch"
                "fanfold"
                "german_std_fanfold"
                "german_legal_fanfold"
        sheet : Handle of sheet to change type. Use blank, "0", or "-1" to use
                the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_CENTERH([sheet])
    Center the worksheet data horizontally between the margins on the printed
    page.

    Parameters -
        sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_CENTERV([sheet])
    Center the worksheet data vertically between the margins on the printed
    page.

    Parameters -
        sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_PRINTACROSS([sheet])
    Change the default print direction to across then down.

    Parameters -
        sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use
                the default sheet.

    Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
    the last error.


e = XL_SETHEADER(string [, margin, [limage, [cimage, [rimage, [sheet]]]]])
    Set the printed page header.

e = XL_SETFOOTER(string [, margin, [limage, [cimage, [rimage, [sheet]]]]])
    Set the printed page footer.

    Parameters -
        string : The header/footer definition string. See below for format
                 options. Cannot be blank.
        margin : The margin in inches to use for the header/footer. A blank,
                 "0", or negative value will use the default margin of "0.3".
        limage : Full path to an image to use in place of the left image
                 placeholder.
        cimage : Full path to an image to use in place of the center image
                 placeholder.
        rimage : Full path to an image to use in place of the right image
                 placeholder.
        sheet  : Handle of sheet to set header/footer. Use blank, "0", or "-1"
                 to use the default sheet.

    Format Options -
```

Control	Category	Description
&L	Justification	Left
&C		Center
&R		Right
&P	Information	Page number
&N		Total number of pages
&D		Date
&T		Time
&F		File name
&A		Worksheet name
&Z		Workbook path
&fontsize	Font	Font size
&"font,style"		Font name and style
&U		Single underline
&E		Double underline
&S		Strikethrough
&X		Superscript
&Y		Subscript
&[Picture]	Images	Image placeholder
&G		Same as &[Picture]
&&	Miscellaneous	Literal ampersand &

Text in headers and footers can be justified to the left, center and right by prefixing the text with the control characters &L, &C and &R. For example, "&LHello, World!", "&CHello, World!", "&RHello, World!"

For simple text, if the justification is not specified the text will be center aligned. However, you must prefix the text with &C if you use any other formatting.

You are limited to 3 images in a header/footer.

Returns "1" on success and "-1" on error. XL_ERROR() can be called to return the last error.

Note: The image types supported are PNG, JPEG, and BMP files. There is a hard limit of 255 characters in a header/footer string, including control characters. Strings longer than this will not be written to the document.

e = XL_SETBACKGROUND(image [, sheet])
Set the background image for a worksheet.

Parameters -
image : Full path to an image to use as the sheet background.
sheet : Handle of sheet to set background image. Use blank, "0", or "-1" to use the default sheet.

Returns "1" on success and "-1" on error. XL_ERROR() can be called to return the last error.

Note: The image types supported are PNG, JPEG, and BMP files.

e = XL_HIDEZEROS([sheet])
Hide zero values in worksheet cells.

Parameters -
sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use the default sheet.

Returns "1" on success and "-1" on error. XL_ERROR() can be called to return the last error.

e = XL_SHOWROWCOL([sheet])
Show row and column headers on the printed page.

Parameters -
sheet : Handle of sheet to change mode. Use blank, "0", or "-1" to use
the default sheet.

Returns "1" on success and "-1" on error. XL_ERROR() can be called to return
the last error.

Rebuild All Indexes on a file. item '8' on the dialog. Note: this
is in the "extended" dialog which shows when a filename is not specified
from the command line. Indexes can be selected individually,
or all (with F7). Press SAVE, and rebuild begins

Ability to SPLIT data into array
Usage:
sz=SPLIT(array, string, delimiter)
array is the array that the data will be placed into
string is the data to split
delimiter is the sequence of characters to split on

NOTE: The array being used must have the size defined for its elements and
cannot be an alias.

Added the ability to show record locks from *clerk. Can also be
used to terminate sessions directly. New option !L added to *clerk. Using
!L will activate the new locked records list. Enter on a selected entry
will give additional options to the user, including the ability to Kill
or Terminate a locked process without having to go to the command line.
Note: This option is only available on Unix/Linux/BSD

Added UID mapping to filePro, ddir/dprodir option F5.
This allows for UIDs (User IDs) to be aliased to specific
usernames. In the event that a login account is removed from
your system, this can be used to maintain the link between the
removed login's UID and those stored in filePro, effectivly
allowing system variables such as @CB and @UB to be mainained.

Windows Only:
This also has the added benefit of allowing @CB and @UB to
function on Windows by linking a "pseudo" UID to a given
username. These UIDs are automatically generated but can
also be manually added. When a user opens filePro and their
username does not exist in the UID map file, a UID will
be generated for that user. filePro will find the next
available UID in the list, starting from 2000, and assign
it to that username.

On all platforms, UIDs stored in this program must be unique
and in the range 0-65535. Usernames can be duplicated on Unix
and Linux platforms, but must be unique on Windows.

Usernames are case-sensitive on Unix and Linux platforms and
are case-insensitive on Windows platforms.

Environmental Variables:
PFUIDMAP = /path
Alternate filePro UID map file. (Use full path)
Note: Must be set in the environment.

PFUSEUIDMAP = ON
Allows filePro to do UID mapping. Also expands the maximum
username length returned by @CB, @UB, and @ID to 32.
Default: ON

String Functions
All "is" functions return "1" for true and "0" for false.

x=isalpha(fld [, pos])
Is the character at the position given a letter?

x=isdigit(fld [, pos])
Is the character at the position given a number?

x=isalnum(fld [, pos])
Is the character at the position given a letter or number?

```
    x=isspace(fld [, pos])
        Is the character at the position given a whitespace character?
        ' ', '\t', '\n', '\r', '\v', '\f'

    x=islower(fld [, pos])
        Is the character at the position given lowercase?

    x=isupper(fld [, pos])
        Is the character at the position given uppercase?

    x=isxdigit(fld [, pos])
        Is the character at the position given a hexadecimal character?
        '0'-'9', 'A'-'F'

    x=iscntrl(fld [, pos])
        Is the character at the position given a control character?
        ASCII codes 0x00 (nul) - 0x1f (US), and 0x7f (del)

    x=isprint(fld [, pos])
        Is the character at the position given a printable character?
        ASCII codes greater than 0x1f (US) not including 0x7f (del)

    x=ispunct(fld [, pos])
        Is the character at the position given a punctuation character?

    x=isgraph(fld [, pos])
        Is the character at the position given a character with a
        graphical representation?  The characters with graphical
        representation are all those characters than can be printed
        (as determined by isprint) except for space.

    x=tolower(fld [, pos])
        Return the character at the position given as a lowercase
        character.

    x=toupper(fld [, pos])
        Return the character at the position given as an uppercase
        character.

    str=strtolower(fld)
        Return the entire string converted to lowercase.

    str=strtoupper(fld)
        Return the entire string converted to uppercase.

Added new array size function to get the size of an array. Can be
    used with GLOBAL, EXTERN, LOCAL, and SYSTEM arrays.

x=ARRAYSIZE(array)
    Where array is the name of the array.
    Where x is the returned size of the passed array.

Added new DECLARED function to check if an array or longvar is
    defined, meaning it is either declared LOCAL or GLOBAL or is declared
    EXTERN but has a matcing GLOBAL definition.

    x=DECLARED(var)
        Where var is either a longvar or an array.
        Where x is the return value.
    Returns 0 if the variable is not fully defined.
    Returns 1 if the variable is fully defined.

Increased ACTION length in debugger from 60 characters to full 128.
    Should now be the same as *cabe.

Added new flag -DM to [dr]clerk to disable the Index Mode prompt
    from @ENTSEL.  Only works when not in update mode.

Added flag -RH to report to disable the automatic record number
    reporting in the middle of the screen.  This enables placing text on the
    center of the screen without it being overwritten when the display updates.

x=@GUI.PAUSE()
    Pauses automatic screen updating while in GI/Web.
```

x=@GUI.RESUME()
    Resumes automatic screen updating while in GI/Web.

REPLACE() enhancement - allow null characters
    Enhanced REPLACE() to accept null characters

FORM WITHPROC
    FORM WITHPROC "formname"
    FORMM WITHPROC "formname"

    Added additional command switch to FORM and FORMM commands to allow
    the associated processing table to run while in input processing.

    Note: You cannot call the WITHPROC variant from within another form
    UNLESS the calling form is a processing only form.

Addqual Program
    Addqual allows you to easily add qualifiers to your files either
    interactively or through the command line.

    This runs interactively:
    addqual [filename]

    This runs automatically:
    addqual filename -q <qualname>
    as does this:
    addqual filename -q <qualname> -x <qual-to-copy-from>

    The automatic commands will display graphics on errors. You can keep
    graphics off with "-s" and errors will be printed on the command line
    if they occur.
    example:
    addqual filename -q <qualname> -s

    List of switches:
    -q qualifier to create
    -x qualifier to copy indexes from
    -s silent, no graphics
    -h --help syntax help

XFER - encrypted transfers server-peer

CABE F6 list files from F8 L-Load

==================================
Version 6.1.XX.05 bug fixes
==================================
Corrected an issue when using the Rebuild Indexes option in dxmaint where
    options were not toggling correctly.

Fixed a regression where scrollable fields in a popup weren't displaying
    correctly.

Fixed message boxes to better handle filePro escape codes.

Fixed -pv flag and print to screen to no longer corrupt the output.

Fixed alternate automatic processing loading in cabe, preventing variables from
    resolving correctly during syntax check.

Fixed a too many open files bug in fpcopy when working on a file with many
    qualifiers and indexes.

Corrected fppack to correctly handle encrypted files.

Changed index rebuild message location on the screen to no longer be hidden
    behind the progress updates.

Corrected an issue preventing GI/fileProWeb from loading [dr]report and
    [dr]clerk on Windows.

Fixed a crash with the PDF import code.

Corrected a crash when opening more than one JSON file at a time.

```
==================================
Version 6.1.XX.04 bug fixes
==================================
Fixed an issue where libodbc would not correctly be found when initializing
    features that use ODBC.

Corrected an issue with RINSTR() where the starting position wasn't honored
    correctly.

Option 'C' to clear selection set in [dr]clerk will no longer cause an infinite
    loop when going back into index selection.

Updated PNG support for PDF outputs. Previously, some PNG files would appear
    corrupted when imported.

Corrected a potential crash when moving/reordering blob fields inside of
    dmoedef.

Added PFOLDCHAIN to allow CHAIN to return to the top of processing when a record
    is saved and the chain was performed inside of an event.

Updated listbox and selectbox code to no longer go outside of screen bounds.

Fixed date handling in XLSX generation when not using the datetime functions.

Fixed an issue where blobs/memos could become corrupted if assigning to the
    field more than once without writing the record.

==================================
Version 6.1.XX.03 bug fixes
==================================
Updated tokenization engine to increase parsing speed.

Corrected Memory fault in FPSQL

Corrected licinfo to read license fallback file.

Corrected memory leaks in [dr]clerk and [dr]report.

Corrected issue where a select or list box would not clear correctly from the
    screen.

Fixed positioning and moving objects (memo) on a form.

Corrected button text in F6 cabe.

Fixed an early error exit condition in ddir to report an error rather than
    exiting.

Corrected "stair step" issue in cabe when using the -C flag on Linux/Unix.

Corrected a crash in clerk when using F5 to duplicate fields between records.

Updated F5 duplicate key in clerk to work with scrolling fields.

Added PFREUSEADDR=ON (default ON) to enable a port to be rebound more quickly
    when using sockets.

Added code to prevent a dummy field from being used as a foreign key when
    performing a fuzzy search.

Corrected and reverted wildcard behaviour during selection in clerk.

Corrected type checking for associated fields in selection sets.

Added buttons to clerk fuzzy search for scrolling the file map.

Increased the number of fields shown in fuzzy search in clerk.

Fixed some button shifting for F6 key in cabe.

Corrected ALL operator in short selection to properly update the selection popup.

==================================
```

Version 6.1.XX.02 bug fixes
==========
Task #1948 Autosave not honoring config flags
    Corrected an issue where Autosave was not correctly reading
    config variables. Addded initial change backup.

Task #1950 Scrolling fields in popups break placement
    Corrected an issue when drawing a popup that contains a scrolling field.

Task #1951 Enhanced runtime format for WHEN flags
    Enhanced runtime format to support extended WHEN flags.

    Added support for @WUKx* @WHPx* and @WBLx*. New WHEN values will be ignored
    in older versions of filePro.

==========
Version 6.1.XX.01 bug fixes
==========
Task #1945 ALL fields search in selection broken
    Corrected ALL field search code for selections.

Task #1947 Short selection prompting twice
    Corrected an issue where short selection was displaying the old selection
    screen.

Task #1949 Enable REVERT command
    Correctly enabled the REVERT command for release.

==========
End End End End End End End End End
==========