

Data encryption supports various encryption standards.

Blowfish, Twofish, AES, Rijandel, DES, Safer+, and Rc2
result = ENCRYPT/DECRYPT(data,method,key [,nonce])

64-bit file I/O breaks the 2 GB file size limit

Nested Calls

'Calls' within processing can now be nested, limited only by system resources

Windows UNC Support

Store your data at \\machine\sharename

SPELLCHECK command

*Check your text memos or fields
for spelling accuracy
Custom dictionaries by user login*

26 Automatic Indexes

Now you can have 26 automatic indexes A-Z

Socket Access commands

(Requires additional purchase & annual license fee)
*Open and monitor TCP/IP ports
Pass information back and forth through the ports*

License Protection

*System calls are excluded from the license user count.
"Free" evaluation software available with expiration dates.*

Increased the number of extended key/data segments from 3 to 6

**Some Operating Systems will not support 64-bit I/O
Verify your system capabilities prior to ordering filePro 5.6*

RESET @PN

This resets @PN to "1", allowing multi-pass printouts to start over on page 1.

GETPID()/GETPPID()

GETPID() returns the process ID of the current process.

GETPPID() returns the process ID of the parent process under Unix/Linux. On Windows, this information is not available, and an empty string is returned.

GET16/GET32/PUT16/PUT32

```
value = GET16( buffer [ ,offset [ ,byteorder ] ] )  
value = GET32( buffer [ ,offset [ ,byteorder ] ] )  
binval = PUT16( value [ ,byteorder ] )  
binval = PUT32( value [ ,byteorder ] )
```

The GETnn functions retrieve a binary value from a buffer, and the PUTnn functions convert the value into binary. The offset parameter specifies the zero-relative offset within the buffer where the value resides. The byteorder parameter specifies the byte order of the binary value, with "L" meaning little-endian, "B" meaning big-endian, and the default being the native order of the current system.

For example, to get the 32-bit little-endian value at offset 8 within the MyBuffer variable, you would use:

```
GET32(MyBuffer,"8","L")
```

To convert the RouterHandle variable into a 16-bit little-endian value, you would use:

```
PUT16(RouterHandle,"L")
```

Note that 8-bit values can already be read/written using the existing ASC and CHR functions.

FIELDNUM()

FIELDNUM(lookupname,fieldname) returns the field number of the given field name. If no such field exists, a null string is returned.

WORDWRAP()/@WORDWRAP[]

Used to manually do word-wrapping of fields.

```
xx = WRAPINFO(value,width[,options])
```

Where "options" is:

- 0 - Word-wrapped text is returned as-is, without hard returns.
- 1 - Text is returned with trailing spaces stripped.
- 2 - Text is returned as-is, including hard returns.

The default is "0". Other values give "undefined results"[1].

```
xx = @WORDWRAP[linenum]
```

The WORDWRAP() function generates wrap text for the given field, as wrapped to the given width. (The field need not be a memo field.)

The @WORDWRAP[] array returns information about the most recent WORDWRAP() call. It is an array of zero-relative values containing the text for each line. Note that some values will be negative, meaning that the line break was caused by a hard return in the buffer just before it. (Also note that the hard return is still in the buffer and would need to be excluded from any printout.)

@SBRKn event

"Start of break" event.

This event is triggered on the first record of a subtotal break.

Unlike @WBRKn, this processing is run prior to the normal processing, and if multiple @SBRKn's are triggered at the same time, they are executed from the outermost level inward. (ie: the opposite order of @WBRKn processing.)

DOKEY

The new DOKEY function allows @key to trap a keystroke and then tell filePro to act on that keystroke as if there were no @key trap. Its syntax is:

```
DOKEY expr
```

where "expr" is the keystroke to perform. Note that only the first character is used, and it need not necessarily be the same as the current @key event.

Note that DOKEY does an implicit END.

For example:

```
@keyX
  Then: msgbox "Are you sure you want to exit?","", "YN"
      If: @bk = "Y"
  Then: dokey "X"
  Then: msgbox "Exit cancelled"
  Then: end
```

```
@keyF
  If: UserCanPrint ne "Y"
  Then: errorbox "You are not allowed to print forms!" ; end
  Then: dokey "F"
```

Also note that DOKEY does not interfere with PUSHKEY, and they can be used in conjunction. For example, to trap "B" and then load the "mybrowse" browse format:

```
@keyB
  Then: pushkey "flmybrowse[ENTR][ENTR]" ; dokey "B"
```

Remember that the DOKEY must come after the PUSHKEY, as it does an implicit END.

SYSTEM() function

```
xx = SYSTEM(command [,noredrawflag] )
```

Equivalent to SYSTEM and SYSTEM NOREDRAW, except that the exit value is returned.

If "noredrawflag" is zero or not supplied, the screen is redrawn after executing the command. If "noredrawflag" is "1", the screen is not redrawn (the equivalent of a SYSTEM NOREDRAW command). Other values are not currently defined.

@LICENSE[]

Contains license information.

Subscripts:

- 1 = A comma-separated list of licensed features.
- 2 = Name
- 3 = Serial number
- 4 = Platform
- 5 = Product name
- 6 = Version
- 7 = User count