FOXTOOLS.FLL

RegFn and CallFn

These notes describe a FOXPRO API library that allows FoxPro programs to call any Windows DLL functions that meet the following requirements:

1) Take the following arguments: integer, long, float, double, string/buffer. These may be passed by reference or by value.

2) The following return types are supported: integer, long, float, double, string/buffer. These are returned by value only.

A surprising number of Windows API functions meet these criteria.

Two FLL functions are used provide this functionality:

RegFn - registers a function and the arguments it takes
CallFn - calls a registered function

RegFn(FunctionName, ArgTypes, ReturnType, DLLName) returns FnHandle

If successful, RegFn returns a number which can be used to reference the function in future calls to CallFn.

Returns -1 if the library couldn't be opened, and will additionally bring up a message box. These cannot be disabled.

The first argument is the name or the ordinal value of the function to be registered.

The second argument is a string of characters which describe what arguments the function will accept. The following values are allowed:

I - Integer
L - Long
C - String of characters
F - Floating point number
D - Double precision floating point number

Each argument type may be preceded by an @ to indicate that the argument is passed by reference. The default is to pass arguments by value.

The third argument is the return type of the function. The values allowed are the same as given above for the argument except that return by reference is not allowed.

The fourth argument gives the name of the DLL library that contains the function. This is optional if the function name (and not an ordinal number) is supplied in the first argument. In this case to all previously loaded libraries are searched in the reverse order that they were loaded. If the function is not found then the standard Windows libraries are searched (i.e. user.exe, krnl386.exe, and gdi.exe). If the DLL is in one of the usual places, then it doesn't have to contain the path. See the Windows SDK function LoadLibrary for exact details about where it looks.

The same function can be registered more than once. This allows functions that take different arguments to be used by declaring them again with different argument types.

Many functions that accept string parameters assign a special meaning to a NULL string. You can pass a NULL pointer to a function expecting a string by passing a 0 in place of the string.

CallFN(FnNum, Arg1, Arg2, ...) returns value from FnNum

Takes at least one argument, a function handle from a previous call to RegFn. Must pass as many arguments as were declared when the function was declared or an error occurs.

All arguments must match their declared type as follows:

F, D - must be a floating point number
I, L - must be an integral number
C - must be a string passed by value, or 0 (zero). If 0, a null pointer is passed.

Returns the value that the function returned, using the type declared by ReturnType in RegFn call.

For More Information about Windows DLLs

The functionality of this FLL is limited mostly by your imagination and your knowledge of the Windows API. Despite the seemingly stringent restrictions, a surprising amount can be done just using the Windows API alone. Look in the directory \foxprow\goodies\foxtools for some example programs that use these functions.

The Windows DLL functions are documented in the books that come with the Windows Software Development Kit. The Windows SDK is also included with Microsoft C/C++ 7.0 and Visual C++. These books are also available separately at many technical bookstores.

Unsupported Functions

The following functions provided in FOXTOOLS.FLL *are not supported by Microsoft Product Support Services (PSS) either electronically or via telephone. They were developed for internal use by some of the FoxPro tools, and although these functions have been around for a while, none of them have been through a true test cycle to test their use in other programs. However, we understand that as developers it is important to get tools as soon as possible, even if they are not fully supported, so we have included the following documentation. Note that these functions may not be supported in future versions of FoxPro.*

ADDBS(<ExpC1>)

Returns:	Character
Description:	Returns <ExpC1> with a backslash appended if needed.
Example:	? ADDBS("C:") (returns: C:\)
	? ADDBS("C:\") (returns: C:\)

CLEANPATH(<ExpC>)

Returns:	Character
Description:	Will return a corrected filename (best guess) for an invalid filename. Removes spaces, invalid characters, duplicate backslashes, etc.
Example:	? CLEANPATH("THISISALONGFILE") (returns: THISISAL) <--- Eight Characters
	? CLEANPATH("C:D:\\THISISALONGFILE") (returns: D:\THISISAL)

DRIVETYPE(<ExpC>)

Returns:	Numeric
Description:	Returns the type of drive passed in <ExpC>
	0 - No type
	2 - Floppy Disk
	3 - Hard Disk

4 - Removable / Network Drive
Example: ? DRIVETYPE("C:") (returns 3)

DEFAULTEXT(<ExpC1>, <ExpC2>)
Returns: Character
Description: Returns <ExpC1> with new extension, if it didn't already have one.
<ExpC1> - File name (With or without a path or extension)
<ExpC2> - Extension (Without a period)
Example: ? DEFAULTEXT("C:\DOS\SETVER", "FLL") (returns: C:\DOS\SETVER.FLL)
? DEFAULTEXT("C:\DOS\SETVER.TXT", "FLL") (returns: C:\DOS\SETVER.TXT)

FORCEEXT(<ExpC1>, <ExpC2>)
Returns: Character
Description: Returns string with old extension replaced with new extension
<ExpC1> - File name (With or without a path or extension)
<ExpC2> - Extension (Without a period)
Example: ? FORCEEXT("C:\FOXPROW\FOXTOOLS.FLL", "DLL")
(returns: C:\FOXPROW\FOXTOOLS.DLL)

FORCEPATH(<ExpC1>, <ExpC2>)
Returns: Character
Description: Returns <ExpC1> with its path changed to <ExpC2>
<ExpC1> - File Name (With or without a path)
<ExpC2> - Path
Example: ? FORCEPATH("C:\DOS\SETVER.EXE", "C:\FOXPROW")
(returns: C:\FOXPROW\SETVER.EXE)

FOXTOOLVER()
Returns: Character
Description: Returns the version number of the FOXTOOLS.FLL library
Example: ? FOXTOOLVER()

JUSTDRIVE(<ExpC>)
Returns: Character
Description: Returns the drive letter from a complete path.
Example: ? JUSTDRIVE("C:\FOXPROW\FOXPROW.EXE")
(returns: C:)

JUSTEXT(<ExpC>)
Returns: Character
Description: Returns the three letter extension from a complete path.
Example: ? JUSTEXT("C:\FOXPROW\FOXPROW.EXE")
(returns: EXE)

JUSTFNAME(<ExpC>)
Returns: Character
Description: Returns the file name from a complete path.
Example: ? JUSTFNAME("C:\FOXPROW\FOXPROW.EXE")
(returns: FOXPROW.EXE)

JUSTPATH(<ExpC>)
Returns: Character
Description: Returns the pathname from a complete path.
Example: ? JUSTPATH("C:\FOXPROW\FOXPROW.EXE") (returns: C:\FOXPROW)

JUSTSTEM(<ExpC>)

Returns: Character
Description: Returns the stem name (first eight characters of file name) from a complete path.
Example: ? JUSTSTEM("C:\FOXPROW\FOXTOOLS.FLL") (returns: FOXTOOLS)

MAINHWND()
Returns: Numeric
Description: Returns a Windows handle (HWND) to the main FoxPro window
Example: ? MAINHWND()

MKDIR(<ExpC>)
Returns: Numeric
Description: Creates a directory specified by <ExpC>. This function will not check for valid length and format of directory string. <see example below>.
The return value is 0 for successful, 1 for not-successful and 6 for "directory already exists". This function is only available in FoxTools version 1.01 or later.
Example:
```
? MKDIR("C:\TEST")
(returns: 0 - Creates directory C:\TEST)
? MKDIR("C:\THISISTOOLONG")
(returns: 0 - Creates directory C:\THISISTO)
? MKDIR("C:\THIS IS WRONG")
(returns: 0 - Creates directory C:\THIS IS - NOTE THE INVALID
 SPACE IS ALLOWED!)
? MKDIR("C:\   THISIS WRONG")
(returns: 1 - Invalid beginning space for directory name)
```

MSGBOX(<ExpC1>, <ExpC2>, <ExpN>)
Returns: Numeric
One Of The Following:

idok	1
idcancel	2
idabort	3
idretry	4
idignore	5
idyes	6
idno	7

Description: Displays a modal dialog box centered in Windows.
<ExpC1> - Contents in dialog box
<ExpC2> - Title of dialog box window
<ExpN> - Type of dialog box as follows:

MB_OK	0
MB_OKCANCEL	1
MB_ABORTRETRYIGNORE	2
MB_YESNOCANCEL	3
MB_YESNO	4
MB_RETRYCANCEL	5
MB_ICONHAND	16
MB_ICONQUESTION	32
MB_ICONEXCLAMATION	48
MB_ICONASTERISK	64
MB_ICONINFORMATION	MB_ICONASTERISK
MB_ICONSTOP	MB_ICONHAND

Example: test = MSGBOX("This is the contents of the window." ;

"This is the title", ;
MB_ICONQUESTION+MB_YESNO)
(This example assumes your program contains #define statements to associate MB_ICONQUESTION and MB_YESNO with their appropriate values shown above).

NEXTWORD(<ExpC1>, <ExpN>[,<ExpC2>])
Returns: Character
Description: Returns the next word in <ExpC1> beginning at character <ExpN> and ending just before any character in optional <ExpC2> or the end of the <ExpC1> string.
Default for <ExpC2> is space, tab and carriage return.
Example: ? NEXTWORD("This is a test of FoxTools", 12) (returns: est)
? NEXTWORD("One,Two,Three,Four,Five", 12,",") (returns: ee)

REDUCE(<ExpC1>, <ExpC2>)
Returns: Character
Description: Removes repetitive values in a string. Usually used to remove groups of spaces in a string and replace it with one space.
<ExpC1> - Character string to change
<ExpC2> - Characters to search for
Example: ? REDUCE("This is   lots of spaces", " ")
(returns: This is lots of spaces)
? REDUCE("Repeat characters AAAA and delete", "A")
(returns: Repe t ch r cters   nd delete)

RMDIR(<ExpC>)
Returns: Numeric
Description: Deletes a directory specified by <ExpC>. Does not check for valid directory name. (See Example). Returns 0 if successful and 1 if not-successful. This function is only available in FoxTools version 1.01 or later.
Example: ? RMDIR("C:\THIS IS WRONG") (returns: 0 - If directory was removed)

STRFILTER(<ExpC1>, <ExpC2>)
Returns: Character
Description: Returns only characters specified in <ExpC2>. This function is case sensitive.
<ExpC1> - Character string to search
<ExpC2> - Characters to search for
Example: ? STRFILTER("This is FoxTools", "T") (returns: TT)

VALIDPATH(<ExpC1>)
Returns: Logical
Description: Checks for a valid DOS filename and/or path syntax.
This function is not foolproof and can sometimes think a file has a valid name when it doesn't. However, it will not reject as invalid a valid name.
NOTE: Does not check to see if it exists
Example: ? VALIDPATH("C:\FOO") (returns: .T.)
? VALIDPATH("C:\THISISVERYLONG") (returns: .F.)

WORDNUM(<ExpC1>, <ExpN>[,<ExpC2>])
Returns: Character
Description: Returns the <ExpN> word in <ExpC> delimited by any character in optional <ExpC2>.
The default for <ExpC2> is a space, tab and carriage return.
Example: ? WORDNUM("This is a test of FoxTools", 4) (returns: test)
? WORDNUM("One,Two:Three,Four,Five", 4, ":,") (returns: Four)

WORDS(<ExpC1>[,<ExpC2>])

Returns: Numeric
Description: Returns the number of words in <ExpC> delimited by any character in optional <ExpC2>. The default for <ExpC2> is a space, tab and carriage return.
Example: ? WORDS("This is a test of FoxTools") (returns: 6)

Windows Clipboard Functions

These functions map to the Windows SDK functions that are similarly named. We strongly recommend that you don't use these unless you know exactly what you're doing! The FoxPro system variable _CLIPTEXT is the recommended way of accessing the Windows clipboard.

CLOSECLIP()
Returns: Logical
Description: Closes the clipboard opened previously with OPENCLIP. Returns .T. if successful.
Example: ? CLOSECLIP() (returns: .T.)

COUNTCLIPF()
Returns: Numeric
Description: The COUNTCLIP function retrieves the number of different data formats currently in the clipboard.
Example: ? COUNTCLIP() (returns: 0)

EMPTYCLIP()
Returns: Logical
Description: The EMPTYCLIP function empties the clipboard and frees handles to data in the clipboard. It then assigns ownership of the clipboard to the window that currently has the clipboard open.
Example: ? EMPTYCLIP()(returns: .F.)

ENUMCLIPFM(<ExpN>)
Returns: Numeric
Description: The ENUMCLIPFM function enumerates the formats found in a list of available formats that belong to the clipboard. Each call to this function specifies a known available format; the function returns the format that appears next in the list.

GETCLIPDAT(<ExpN>)
Returns: Logical
Description: Unknown function... best explanation as follows:
The GetClipDat function retrieves a handle of the current clipboard data having a specified format in <ExpN>.
<ExpN> contains the following defines:

cf_Text	=	1
cf_Bitmap	=	2
cf_MetaFilePict	=	3
cf_SYLK	=	4
cf_DIF	=	5
cf_TIFF	=	6
cf_OEMText	=	7
cf_DIB	=	8
cf_Palette	=	9

GETCLIPFMT(<ExpN>)
Returns: Unknown (Character?)
Description: Unknown function... best explanation as follows:

The GetClipFmt function retrieves the name of a registered clipboard format.

ISCLIPFMT(<ExpN>)

Returns: Logical

Description: Returns the format of the available contents of the clipboard. <ExpN> is as follows:

cf_Text	=	1
cf_Bitmap	=	2
cf_MetaFilePict	=	3
cf_SYLK	=	4
cf_DIF	=	5
cf_TIFF	=	6
cf_OEMText	=	7
cf_DIB	=	8
cf_Palette	=	9

Example: Copy text to clipboard
? ISCLIPFMT(cf_Text) (returns: .T.)

OPENCLIP(<ExpN>)

Returns: Logical

Description: The OPENCLIP function opens the clipboard. Other applications will not be able to modify the clipboard until the CLOSECLIP function is called. <ExpN> is the handle of the window ... 0 is acceptable
<<< ALMOST ALL CLIP FUNCTIONS RELY ON OPENING THE CLIPBOARD PRIOR TO USE >>>

Example: ? OPENCLIP(0)
(returns: .T.)

REGCLIPFMT(<ExpC>)

Returns: Numeric
The return value indicates the newly registered format. If the identical format name has been registered before, even by a different application, the format's reference count is incremented (increased by one) and the same value is returned as when the format was originally registered. The return value is zero if the format cannot be registered.

Description: The REGCLIPFMT function registers a new clipboard format. The registered format can be used in subsequent clipboard functions as a valid format in which to render data, and it will appear in the clipboard's list of formats. <ExpC> is a string that names the new format.

Example: ? REGCLIPFMT("MyRegClip")

SETCLIPDAT(<ExpN>, <ExpC>)

Returns: Logical

Description: The SETCLIPDAT function sets the data in the clipboard.

Example: ? SETCLIPDAT(1, "Test") (returns: .T.)

Editor and Window Functions

A group of functions starting with **_Ed...** and **_W...** map to editor and window functions that are described in the FoxPro 2.5 Library Construction Kit (a separate product that allows you to extend FoxPro using the C programming language). See the documentation in that product for details of these functions.