Advanced File Operation API

by Alyce Watson http://alycesrestaurant.com/

Table of Contents

This tutorial assumes that the reader has a good knowledge of making API calls in Liberty BASIC. The ABCs of APIs series gives basic information on the use of CALLDLL. You can find it here:
Advanced Tutorials

SHFileOperationA from Shell32.DLL

Liberty BASIC has many simple file operation commands available. We can perform more complex file operations with API functions.

Liberty BASIC has a command to delete a file called **KILL**. This command deletes the file forever. If instead we wish to send it to the recycle bin so that our users can recover their files, we can use SHFileOperationA.

The format of the API call is:

```
calldll #shell32, "SHFileOperationA",_ 'Function Name
LPSHFILEOPSTRUCT,_
'structure to give and receive info from function
result as long         'zero = success
```

The only member passed into this function is a pointer to an SHFILEOPSTRUCT structure that contains

information this function needs to carry out the specified operation. According to MSDN, "This parameter must contain a valid value that is not NULL. You are responsible for validating the value. If you do not validate it, you will experience unexpected results."

The function returns zero if successful; otherwise nonzero. Applications normally should simply check for zero or nonzero.

According to MSDN, "It is good practice to examine the value of the fAnyOperationsAborted member of the SHFILEOPSTRUCT. SHFileOperation can return 0 for success if the user cancels the operation. If you do not check fAnyOperationsAborted as well as the return value, you cannot know that the function accomplished the full task you asked of it and you might proceed under incorrect assumptions."

We will not address the fAnyOperationsAborted flag in our demo, below.

LPSHFILEOPSTRUCT

We use a struct for this API call. We fill some members of the struct before making the API call. The function fills some members of the struct when it performs the requested action. We can access the information afterwards, if we want to do so.

According to MSDN:

Important You must ensure that the source and destination paths are double-null terminated. A normal string ends in just a single null character. If you pass that value in either the source or destination members, the function will not realize when it has reached the end of the string and will continue to read on in memory until it comes to a random double null value. This can at least lead to a buffer overrun, and possibly the unintended deletion of unrelated data.

The struct is as follows. We are calling it SHFILEOPSTRUCT.

```
struct SHFILEOPSTRUCT,_
hWnd as uLong,_
wFunc as Long,_
pFrom as ptr,_
pTo as ptr,_
fFlags as long,_
fAnyOperationsAborted as long,_
hNameMappings as Long,_
lpszProgressTitle as ptr
```

The following information is from MSDN and it explains the struct members.

hwnd
HWND
A window handle to the dialog box to display information about the status of the file operation.

wFunc
UINT
A value that indicates which operation to perform. One of the following values:

FO_COPY
Copy the files specified in the pFrom member to the location specified in the pTo member.

FO_DELETE
Delete the files specified in pFrom. **File names must be specified as fully qualified paths and filenames, or they will not be moved to the recycle bin.**

FO_MOVE
Move the files specified in pFrom to the location specified in pTo.

FO_RENAME
Rename the file specified in pFrom. You cannot use this flag to rename multiple files with a single function call. Use FO_MOVE instead.

pFrom
LPCTSTR
Note This string must be double-null terminated.

A pointer to one or more source file names. These names should be fully-qualified paths to prevent unexpected results.

Standard MS-DOS wildcard characters, such as "*", are permitted only in the file-name position. Using a wildcard character elsewhere in the string will lead to unpredictable results.

Although this member is declared as a single null-terminated string, it is actually a buffer that can hold multiple null-delimited file names. Each file name is terminated by a single NULL character. The last file name is terminated with a double NULL character ("\0\0") to indicate the end of the buffer.

pTo
LPCTSTR
Note This string must be double-null terminated.

A pointer to the destination file or directory name. This parameter must be set to NULL if it is not used. Wildcard characters are not allowed. Their use will lead to unpredictable results.

Like pFrom, the pTo member is also a double-null terminated string and is handled in much the same way. However, pTo must meet the following specifications:

Wildcard characters are not supported.
Copy and Move operations can specify destination directories that do not exist. In those cases, the system attempts to create them and normally displays a dialog box to ask the user if they want to create the new directory. To suppress this dialog box and have the directories created silently, set the FOF_NOCONFIRMMKDIR flag in fFlags.
For Copy and Move operations, the buffer can contain multiple destination file names if the fFlags member specifies FOF_MULTIDESTFILES.
Pack multiple names into the pTo string in the same way as for pFrom.
Use fully-qualified paths. Using relative paths is not prohibited, but can have unpredictable results.

fFlags
FILEOP_FLAGS
Flags that control the file operation. This member can take a combination of the following flags.

fAnyOperationsAborted
BOOL
When the function returns, this member contains TRUE if any file operations were aborted before they were completed; otherwise, FALSE. An operation can be manually aborted by the user through UI or it can be silently aborted by the system if the FOF_NOERRORUI or FOF_NOCONFIRMATION flags were set.

hNameMappings
LPVOID
When the function returns, this member contains a handle to a name mapping object that contains the old and new names of the renamed files. This member is used only if the fFlags member includes the FOF_WANTMAPPINGHANDLE flag. See Remarks for more details.

lpszProgressTitle
LPCTSTR
A pointer to the title of a progress dialog box. This is a null-terminated string. This member is used only if fFlags includes the FOF_SIMPLEPROGRESS flag.

File Operation Flags

The flags for the fFlags member of the struct as as follows, as explained on MSDN:

FOF_ALLOWUNDO
Preserve undo information, if possible.

Prior to Windows Vista, operations could be undone only from the same process that performed the original operation.

In Windows Vista and later systems, the scope of the undo is a user session. Any process running in the

user session can undo another operation. The undo state is held in the Explorer.exe process, and as long as that process is running, it can coordinate the undo functions.

If the source file parameter does not contain fully qualified path and file names, this flag is ignored.

FOF_CONFIRMMOUSE
Not used.

FOF_FILESONLY
Perform the operation only on files (not on folders) if a wildcard file name (*.*) is specified.

FOF_MULTIDESTFILES
The pTo member specifies multiple destination files (one for each source file in pFrom) rather than one directory where all source files are to be deposited.

FOF_NOCONFIRMATION
Respond with Yes to All for any dialog box that is displayed.

FOF_NOCONFIRMMKDIR
Do not ask the user to confirm the creation of a new directory if the operation requires one to be created.

FOF_NO_CONNECTED_ELEMENTS
Version 5.0. Do not move connected files as a group. Only move the specified files.

FOF_NOCOPYSECURITYATTRIBS
Version 4.71. Do not copy the security attributes of the file. The destination file receives the security attributes of its new folder.

FOF_NOERRORUI
Do not display a dialog to the user if an error occurs.

FOF_NORECURSEREPARSE
Not used.

FOF_NORECURSION
Only perform the operation in the local directory. Don't operate recursively into subdirectories, which is the default behavior.

FOF_NO_UI
Version 6.0.6060 (Windows Vista). Perform the operation silently, presenting no UI to the user. This is equivalent to FOF_SILENT | FOF_NOCONFIRMATION | FOF_NOERRORUI | FOF_NOCONFIRMMKDIR.

FOF_RENAMEONCOLLISION
Give the file being operated on a new name in a move, copy, or rename operation if a file with the target name already exists at the destination.

FOF_SILENT
Do not display a progress dialog box.

FOF_SIMPLEPROGRESS
Display a progress dialog box but do not show individual file names as they are operated on.

FOF_WANTMAPPINGHANDLE
If FOF_RENAMEONCOLLISION is specified and any files were renamed, assign a name mapping object that contains their old and new names to the hNameMappings member. This object must be freed using SHFreeNameMappings when it is no longer needed.

FOF_WANTNUKEWARNING
Version 5.0. Send a warning if a file is being permanently destroyed during a delete operation rather than recycled. This flag partially overrides FOF_NOCONFIRMATION.

Flag Operation Values

These flag values are not understood by Liberty BASIC, so we must set the values ourselves. Liberty BASIC reserves the underscore character in variable names for Windows Constants that it recognizes, so we'll replace the underscore with a dot.

```
FO.MOVE = hexdec("0001")
FO.COPY = hexdec("0002")
FO.DELETE = hexdec("0003")
FO.RENAME = hexdec("0004")

FOF.MULTIDESTFILES = hexdec("0001")
FOF.CONFIRMMOUSE = hexdec("0002")
FOF.SILENT = hexdec("0004")
FOF.RENAMEONCOLLISION = hexdec("0008")
FOF.NOCONFIRMATION = hexdec("0010")
FOF.WANTMAPPINGHANDLE = hexdec("0020")
FOF.ALLOWUNDO = hexdec("0040")
FOF.FILESONLY = hexdec("0080")
FOF.SIMPLEPROGRESS = hexdec("0100")
FOF.NOCONFIRMMKDIR = hexdec("0200")
FOF.NOERRORUI = hexdec("0400")
FOF.NOCOPYSECURITYATTRIBS = hexdec("0800")
FOF.NORECURSION = hexdec("1000")
FOF.NO.CONNECTED.ELEMENTS = hexdec("2000")
FOF.WANTNUKEWARNING = hexdec("4000")
```

```
FOF.NORECURSEREPARSE = hexdec("8000")
```

Demo

This short demo creates a text file in the program's default directory, then uses to move the file to the recycle bin, without asking the user for confirmation.

```
    struct SHFILEOPSTRUCT,_
    hWnd as uLong,_
    wFunc as Long,_
    pFrom as ptr,_
    pTo as ptr,_
    fFlags as long,_
    fAnyOperationsAborted as long,_
    hNameMappings as Long,_
    lpszProgressTitle as ptr

FO.DELETE = hexdec("0003")
FOF.ALLOWUNDO = hexdec("0040")
FOF.NOCONFIRMMKDIR = hexdec("0200")

testfile$ = DefaultDir$ + "\shfiletest.txt"

'create a test file to delete
open testfile$ for output as #1
print #1, "Testing."
close #1

'name of file on disk
SHFILEOPSTRUCT.pFrom.struct = testfile$+chr$(0)+chr$(0)
'move to recycle bin without asking user for confirmation
SHFILEOPSTRUCT.fFlags.struct = FOF.NOCONFIRMATION  or FOF.ALLOWUNDO
'function to perform = delete file
SHFILEOPSTRUCT.wFunc.struct =  FO.DELETE ' delete flag

CallDLL #shell32, "SHFileOperationA", _
    SHFILEOPSTRUCT as STRUCT,_
    result as long

if result = 0 then
    print testfile$
    print "File was successfully moved to recycle bin."
else
    print testfile$
```

```
    print "File was not able to be removed."
end if
```