Translating documentation for DLLs in MS "C" into Liberty BASIC

-
petermat1
Translating documentation for DLLs in MS "C" into Liberty BASIC | Introduction

Introduction

Why is the note necessary? Because Microsoft documents it's API's using C / C++ examples.

This note was originally intended to follow on from Alyce Watson's ABCs of APIs #10 and still assumes that the reader is familiar with the information contained in that series. However I am afraid this note is not by an expert in this topic, just someone who is struggling to move beyond what the ABC series covered. If I fail, maybe you will be encouraged to pick up and correct the note. You cannot edit this wiki directly, but messages sent to me will be attended to.

The First Example.

The following LB code is due to Alyce Watson
at http://libertybasicuniversity.com/lbnews/nl109/browse.htm.
I don't intend to go into the semantics or use of this - just its translation.

```
calldll #shell32, "SHGetPathFromIDList",_
            lpIDList as long,_
            sPath$ as ptr,_
            r as long
```

The MS definition at https://msdn.microsoft.com/en-us/library/windows/desktop/bb762194(v=vs.85).aspx is:

```
BOOL SHGetPathFromIDList(
  _In_  PCIDLIST_ABSOLUTE pidl,
  _Out_ LPTSTR            pszPath
);
```

This is C++ code. SHGetPathFromIDList is the name of a function - just like an LB function - and the comma delimited items inside the () are the two function parameters. A C++ function returns a value just like an LB function - which is, as described in previous notes, the last parameter in the LB call - here it's "r as long". You do have to know that "BOOL" defines the type of the C++ return and that this needs a "long" type in LB. The "_In_" and "_Out_" mean just what you might think. "pidl" is defined by the C++ code as of type "PCIDLIST_ABSOLUTE" and the MS definition of this is "address of an item identifier list" which has to be type long in LB. Just remember that in C++ the type comes before the parameter name, and vice versa in LB.

The Second Example.

Now let's start with the C++ definition and code from https://msdn.microsoft.com/en-us/library/windows/desktop/bb762115(v=vs.85).aspx

```
PIDLIST_ABSOLUTE SHBrowseForFolder(
  _In_ LPBROWSEINFO lpbi
);
```

Looking at this code we see that the return is of type "PIDLIST" (whatever that is), and there is one parameter which is used as an input by function
SHBrowseForFolder. The C++ definition page is headed "The Windows Shell Shell Reference Shell Functions" which is the clue we are still working with shell32.dll, so we can continue to use #shell32 in our LB code. The definition tells us that type LPBROWSEINFO is "A pointer to a BROWSEINFO structure" - which at least tells us it is of type struct from an LB perspective. And lastly, the definition states that the return is "a PIDL that specifies the location of the selected folder" - so this is an address and hence needs to be 'long'. A PIDL is a "Pointer to an Item iDentifier List". Windows Shell needs to know the identity of all the items it manages. Most, but not all, of these "items" are files with an associated file path. Shell deals with all its items by assigning a unique id number to each item and storing all these ids in an Identifier list. For all purposes associated with files, each PIDL has a one to one relationship with a specific file path.

```
calldll #shell32, "SHBrowseForFolder",_
            lbpi struct,_
            result as long
```

All very well - but what about this struct? The BROWSEINFO structure is defined in
https://msdn.microsoft.com/en-us/library/windows/desktop/bb773205(v=vs.85).aspx
as shown below on the left:

```
      C++ definition                                     Equivalent LB code

typedef struct_browseinfo {                            STRUCT BrowseInfo,_
  HWND                hwndOwner;                        hWndOwner As uLong,
_
  PCIDLIST_ABSOLUTE pidlRoot;                           pIDLRoot As Long,_
  LPTSTR              pszDisplayName;                   pszDisplayName As L
ong,_
  LPCTSTR             lpszTitle;                        lpszTitle$ As ptr,_
  UINT                ulFlags;                          ulFlags As Long,_
  BFFCALLBACK         lpfn;                             lpfnCallback As Lon
g,_
  LPARAM            lParam;                         lParam As Long,_
  int               iImage;                         iImage As Long
} BROWSEINFO, *PBROWSEINFO, *LPBROWSEINFO
```

Translating this is (mostly) straightforward - and produces the LB code on the right above. I have no idea what the "BROWSEINFO, *PBROWSEINFO, *LPBROWSEINFO" text is about - it appears not to be necessary to the translation into LB (Anyone?)

The End . . ..

Well that's it as far as translation from C++ API calls to LB is concerned. "But" do I hear you cry "I don't understand all the terms in the struct you just defined, or what to do with a PIDL, or, or, or . . . !" Well yes, to be honest, that's where the hard work starts. There is no 'one size fits all' explanation from here on in. You just have to dive into the https://msdn.microsoft.com/en-us/library/windows/desktop/ jungle (or appropriate 3rd party dll jungle) and try to make sense of what you need. For example you can follow up on PIDLs at https://msdn.microsoft.com/en-us/library/windows/desktop/cc144090(v=vs.85).aspx. This is very helpful on this topic, but includes terms you may not understand (I don't) like, "UNC path", " Namespace Extensions.", "SHITEMID structure", relative and single- level PIDLs, etc., and you are likely to have to follow up on "Getting a Folder's ID," at https://msdn.microsoft.com/en-us/library/windows/desktop/bb776914(v=vs.85).aspx, which may well lead to yet further explanations on yet further web pages - and so on, and on.

Or you may just get lucky, and the API you want to use has an example at the API introduction page

So now you know why we get paid the big bucks!
SHBrowseForFolderSTRUCT BrowseInfo,_