

Re: Structure Marshalling Question

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.interop/2005-03/0015.html>

From: Thomas Delany (*eshuttle_at_community.nospam*)

Date: 02/28/05

Date: Mon, 28 Feb 2005 10:21:51 -0500

On Sat, 26 Feb 2005 22:02:51 -0500, SB wrote:

```
> Have you looked into Marshal::PtrToStringAnsi? I tested the code below and
> it worked fine for me. I didn't see a need to create a managed
> structure...so I just used the unmanaged one you posted:
> typedef struct tag_KVFilterOutput
> {
>     BYTE *pcText;
>     int cbText;
> }KVFilterOutput;
> Here is the code in the managed C++ app:
> private: System::Void button1_Click(System::Object^ sender,
> System::EventArgs^ e)
> {
>     KVFilterOutput kvf;
>     ZeroMemory(&kvf, sizeof(KVFilterOutput)); // just a habit of mine
>     kvf.cbText = 256;
>     kvf.pcText = (BYTE *) calloc(kvf.cbText, 1);
>     if (kvf.pcText != NULL )
>     {
>         TestFunc(&kvf); // call the dll function
>         IntPtr^ p = gcnew IntPtr(kvf.pcText);
>         System::String^ s =
>         System::Runtime::InteropServices::Marshal::PtrToStringAnsi(*p, kvf.cbText);
>         free(kvf.pcText);
>     }
> }
>
> Note that the above is .NET 2.0b code...so the syntax for you will be
> slightly different.
>
> Oh...and the DLL code I used to simulate your unmanaged code was something
> like:
> extern "C" __declspec(dllexport) void TestFunc(KVFilterOutput *data)
> {
>     BYTE buffer[] = {65, 66, 67}; // "ABC" in ANSI
```

> memcpy(data->pcText, buffer, 4);
>> [quoted text muted]
>
> Note that I also set the calling convention in the dll and the application
> to _stdcall.
>
> Of course, someone else might point out a better way to do this since I'm
> by no means an expert in this area (or in managed C++...since I normally use
> C#).
>
> Also, I wondered why you are writing a managed C++ wrapper and using it from
> C#...when you can just do it all in C#?
>
> HTH,
> -sb
>
>
> "Thomas Delany" <eshuttle@community.nospam> wrote in message
> news:pan.2005.02.26.17.02.04.125000@community.nospam...
>> [quoted text muted]

A couple of things:

You do not allocate the memory before you call the API call. It allocates the memory that it needs, and returns the size in the cbText structure member. It is then the caller's responsibility to free the memory. That complicates matters a bit.

The second thing, and the reason that I decided on a C++ wrapper, was because of the way some of the API functions work... You have to call setup functions before you call some of the API functions. One of the parameters to one of the setup functions is a structure containing function pointers for disk I/O functions that the DLL will use. I could see not way to call that setup function from C#.

Tom Delany