Subject: Re: CALL_EXTERNAL
Posted by Randall Skelton on Wed, 05 Feb 2003 08:47:33 GMT
View Forum Message <> Reply to Message

Hi Thomas,

IDL pointers != C pointers

In short, you cannot pass an IDL pointer into C and expect to do anything useful with it. You most certainly cannot pass an IDL pointer to C, operate on it, and then pass it back with useful data. If you would like this functionality added to IDL, please write support@rsinc.com and add your name to the growing list of people wanting the C API to the heap variable and objects. In reality, however, this is probably not the functionality that you need.

What you are trying to do is a little outside the scope of what call external is truely useful for. You should probably consider the purchase of Ronn Kling's book on calling C from IDL using DLMs:

http://www.kilvarock.com/books/callingCfromIDL.htm

I'm not sure how feasible it is using CALL_EXTERNAL, but the only way you might be able to do this would involve passing an undefined variable (idl type=0) and allocate the memory directly using C. Check out the middle of Chapter 9 (External Development Guide) for the C functions that can help in this regard (IDL_MakeTempArray, IDL_ImportArray, IDL_ImportNamedArray). Once you have allocated the IDL array variable you then need to tie it to the passed (undefined) variable using IDL_VarCopy.

Again, the best way to do this is via a DLM and Ronn's book is, by far, the best way to learn.

Cheers,
Randall

On Wed, 5 Feb 2003, Thomas Gutzler wrote:

> Hi again,
>
> please correct me if I'm wrong.
>
> CALL_EXTERNAL
> - needs the /CDECL keyword to call a dll built by Borland C++ Compiler
> with: extern "C" __declspec(dllexport)
> It works with and without /CDECL for me
> - can have values and references (default) as arguments but cannot

> RETURN a pointer. Just a scalar variable with the value of an address
> - cannot have a NullPointer as argument and receive a Pointer to a
> variable/array (see below)
>
> What I'm trying to do is:
>
> C function:
> int test(int argc, void* argv[])
> {
> if(argc != 2) return 0;
> IDL_UINT *size = (IDL_UINT *) argv[0];
> int *array = (int *) argv[1];
> array = new int[*size];
> for (int i = 0; i < *size; i++) array[i] = i;
> return 1;
> }
>
> IDL:
> IDL> array = PTR_NEW()
> IDL> Result = CALL_EXTERNAL('mydll.dll', '_test', 10, array, /CDECL)
> IDL> print, array
> should be
> 0 1 2 3 4 5 6 7 8 9
> but array is still a NullPointer
>
> If I leave the line
> array = new int[*size];
> I have to initialize with array=intarr(10) and the returned array is:
> 0 0 1 0 2 0 3 0 4 0
>
> This is not what I expected (funny values! maybe type-conversion helps?)
> and not what I want. I want to receive a pointer to an array, because
> IDL doesn't know the size of the array being returned (Yes, I could
> allocate a 1000000x1 array and resize it).
>
> Is this possible ?
>
> Tom
>
>