

Re: Export C++ class from a Borland DLL and use it in Microsoft VC

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vc/2006-10/msg00003.html>

- *From:* Tamas Demjen <tdemjen@xxxxxxxxxx>
 - *Date:* Fri, 29 Sep 2006 10:24:53 -0700
-

Fabry wrote:

I have a DLL with its export library (.lib) wrote in Borland C++ 6. In borland everything is OK and I am able to include the lib and use the class that i have exported creating an instance with new etc... I would like to export this class in microsoft VC++ using the same .lib file. Obviously it doesn't work.

Borland uses a different C runtime (including a different memory manager) than Visual Studio. What you're asking for won't even work between VC++6 and VC++8.

Of course, it is not impossible to use a DLL across compilers, even across different languages. You can write DLLs in VC++ and use them from Delphi. You just have to follow very strict (and super inconvenient) rules to ensure cross-compiler compatibility.

You have a couple of choices here:

1A. Use a C-style DLL. Flatten your object hierarchy into C-style (extern "C" __stdcall) functions. Ensure that all your exported functions have pure C function arguments. So pass a string as const char*, pass a vector<T> as const T*, and so on. Every memory that you DLL allocates MUST be freed by the same DLL. You either have to use the concept of allocators, or export your functions in pairs (for each function that allocates, there must be another function that frees).

1B. You can go ahead and use C++ in a restricted way. Note that due to name mangling you can't export C++ classes themselves. You still need to use pure extern "C" exports, but the functions themselves can accept and return C++ classes. Instead of exporting those classes, you have to bind them using virtual function calls. Your DLL should define an interface for each class, which is linked to both the DLL and the EXE that uses the DLL. Function calls are dispatched via the VMT, which is guaranteed to be portable between Borland and Microsoft. But you still need to ensure that any memory allocated by the DLL is deleted by the same DLL. You must avoid using STL altogether in the exported functions' argument list (no std::string, no std::vector). Please see my draft article at:

<http://tweakbits.com/articles/dll/index.html>

I do this all the time, I share DLLs between VC++ and C++Builder routinely without any problem. You just

Re: Export C++ class from a Borland DLL and use it in Microsoft VC

have to follow my article very very closely. It's a major pain, because it requires this extra layer of plugin architecture, but it's nothing you can't implement fairly easily. It's just a lot of extra code, attention and time.

Special note: By default, enums are treated as bytes in C++Builder (because Delphi does the same), but they're integers in VC++. If you pass or return them by value, it doesn't matter that much, but if you pass an enum by pointer or reference, your application is going to crash for sure. You must either not use enums in your interface, or modify Borland's compiler settings for your particular cpp file (not the entire project). The setting is called "treat enums as integers" or something like that. This is the only major binary layout incompatibility between the two compilers, and I shoot myself in the foot a few times with that.

2. COM/ActiveX. I find COM much harder and more cumbersome than either 1A or 1B, but it's always a possibility to link different compilers / languages together.

3. .NET. I find .NET many times easier and infinitely more flexible than COM, but it requires the .NET runtime. Besides, C++Builder doesn't have Managed C++ or C++/CLI, so it's not really an option for you, but it's worth mentioning nonetheless.

++ ... I'm sure there are other component sharing technologies, like Web services, CORBA, etc.

Tom

.