Subject: Re: C++ and CALL_EXTERNAL
Posted by jameskuyper on Fri, 30 May 2008 11:12:39 GMT
View Forum Message <> Reply to Message

mark.t.douglas@gmail.com wrote:
> On 29 May, 12:20, James Kuyper <jameskuy...@verizon.net> wrote:
...
>> Wouldn't it be simpler to disable the name mangling by declaring the
>> functions as 'extern "C"' ? You can still use any feature of C++ that
>> you want, inside the definition of the function. Of course, you can't
>> use any C++ features in the function interface of an 'extern "C"'
>> function that are not also supported by C, but CALL_EXTERNAL probably
>> couldn't handle those features anyway.
>
> That would have worked fine and made life simpler for the two
> functions I outlined here, certainly. However there are other things
> in the DLL which are "proper" C++ so I elected not to use extern "C"
> for the sake of consistency, as the DLL was designed as a C++ library
> in the first instance. I probably should have mentioned this in the
> original post!

I think that hard-coding the name-mangling scheme of one particular
implementation of C++ in your IDL code is a bad idea. It makes your IDL
code harder to read, and it might have to be changed if you use a
different C++ compiler, or even a different version of the same C++
compiler. Declaring the function 'extern "C"' is a lot cleaner and more
portable.

Don't let your concerns about "consistency" make your job harder than it
needs to be. There's nothing wrong with using C++-specific features in
the body of a C++ function with "C" language linkage. This is quite
normal, because such functions usually serve as the interface between
C++ code and non-C++ code. Nor is there any problem with having
functions with "C" language linkage in the same translation unit as
functions with "C++" language linkage.