Subject: Re: IDL and Microsoft Object Linking and embedding OLE
Posted by Richard Younger on Fri, 26 Jan 2001 20:10:12 GMT
View Forum Message <> Reply to Message

Peter Brooker wrote:
>
> I have a software package called Prolith that allows "Visual Basic,
> C++, or other programming languages to control PROLITH from other
> OLE-compliant applications."
>
> You can write a program in C++ and make calls to Prolith routines.
> ProLith then return the data back to the C++ program.
>
> This sounds allot like the "call_external" command in IDL.
>
> They go on to define OLE as object linking and embedding that is a
> "strategic technology from Microsoft that is the standard for
> application integration and interchange in Windows."
>
> Is there anyway that an IDL program can access the OLE ProLith
> routines the same way a C++ program can?
>
> thanks-Peter Brooker

Sorry for the delay in my reply. I've been doing data acquisition and device control through IDL for a little while, but I hardly would consider myself an expert on the subject. Consequently, I wanted to mull it over a bit before I stuck my head out too far. :)

Just about every time, the answer to "Can I call X from IDL" has been "Write a C wrapper function." Occasionally for routines that return no data, just do something like manipulate files, "Make it into an executable and use SPAWN" has been the answer of choice, if you don't mind a little extra overhead.

As to the question if you can use Your Favorite Language (YFL), it depends. AFAIK, Call_External works acceptably if you can get YFL to accept C-style calling conventions. There is an example floating around of calling a Fortran routine. But generally, accepting C calling conventions means some facility for dereferencing pointers, a compiled (not interpreted) language, and some sort of typing.

Linkimage and DLMs prefer C natively, as you need to manipulate C-style IDL internals and there are C libraries that you need to call in your routines. In my experience (admittedly only with MS VC++) C++ will work too, as long as you make sure to set C calling conventions and prefix all the IDL-callables with the (export.h) IDL_CDECL macro. Other languages would be much more work.

There is one major caveat, however. Especially when you start thinking towards callable packages and away from programming languages. Scope and data persistence. Every time you go back to IDL, you loose all your local YFL memory, all your variables. You can play games with dynamically allocating things or global memory blocks and passing pointers to them, or passing data back and forth through IDL variables, but it gets klugey if you've got too many things up in the air. If you are trying to use IDL as an interface interactively with ProLith, you're in for some work. If each call is independent of every other call, it will go more smoothly.

To sum up, I would think that you could write a C++ Call_external function that would call the Prolith routines. Although the package itself might do funky things that IDL doesn't like in its memory space. I know that IDL, for one, doesn't support (nor will it soon support) threaded code. Maybe someone else knows more about whether Prolith will actually interfere with IDL. No guarantees here. I think you'd have to try it to know. Look to the External Development Guide and the online help get you started.

If you haven't used them before, I would guess it would take a few days of playing to get a rudimentary call_external interface up for a few functions, and maybe a few weeks to get a more sophisticated DLM interface going. If you take the DLM route, get Ronn's new book, as it has lots of helpful examples that will save you no end of frustration.

best of luck,

Rich

--
Richard Younger
