Subject: IDL and pointers
Posted by Jonathan Joseph on Wed, 12 Jan 2000 08:00:00 GMT
View Forum Message <> Reply to Message

As far as I can tell, the analog to pointers in IDL is handles.
This is very inconvenient for dereferencing pointers to
structures - especially if you want to do multiple dereferencing.
This is because you need to call HANDLE_VALUE
(twice [using NO_COPY] if you don't want to waste time
making a copy of the data structure).

So, I thought that now that they have been here for a while,
IDL objects should be very useful in these cases - since
they have the dereference operator ".".

But, there does not seem to be a way to access the instance
data outside of an object method. I don't want to have
to write object methods for every little piece of my object I want
to access. In C++ I can define parts of the class to be public and I
can access them at will. Is there any analog to this in IDL?

The current setup seems very inconvenient - especially for
debugging. If I have an object "a" and I want to see what
is in it's "foo" field, I can't even do "IDL> print, a.foo" I get:

% Object instance data is not visible outside class methods: A

Am I missing something fundamental here? I just want
the equivalent of some nice dynamically allocated structures
that I can have multiple pointers to and easily dereference.

For example,

structure A has a field that is a pointer to structure B
structure B has a field that is a pointer to structure C
structure C has the field FOO

I want the equivalent of something like the following C++
assignment: F = A.B->C->FOO
or if we just had a pointer to A: F = A->B->C->FOO

Thanks for any insight.

-Jonathan