Subject: Structures
Posted by Michael Wallace on Tue, 01 Mar 2005 21:29:21 GMT
View Forum Message <> Reply to Message

This isn't a problem, just an academic question. I'm curious why IDL does what it does, but aren't we all?

Once you create a structure, you cannot change type of the variables within the structure, including array sizes. Once the structure is created and named, you can't change an integer array[100] to an integer array[10] or anything else with a different size. This came as a surprise to me since I think of "int array" as being the type, not "int array[100]" or "int array[10]". Why does IDL enforce such a rigid *structure* on structures when the rest of the language which leaves you free to change types, sizes and everything else under the sun? It just doesn't seem to mesh with the rest of the language.

I learned all this the hard way by trying to figure out why I kept getting errors when I'd attempt to use a different array size for some of my structure variables. I finally ran across this in the IDL documentation, but there wasn't any actual reason given for such draconian policy, especially when compared to the rest of the language. It's really not a big deal since I can create anonymous structs everywhere instead of using named structures. Do named structures actually serve a useful purpose other letting you condense your syntax when creating them?

-Mike