Subject: Re: Function referencing/automatic defintion question.
Posted by David Fanning on Thu, 29 May 2003 19:04:55 GMT
View Forum Message <> Reply to Message

Paul van Delst (paul.vandelst@noaa.gov) writes:

> What this tells me is that the "default action" for IDL in this case is to assume that my
> function call is really an array operation where I'm using () instead of [] to subscript
> the array despite the fact that a function with the same name is compiled and resolved in
> the current scope. Huh?
>
> This totally bamboozles me since I have a load of other source code files (including the
> main file for this little project) that have more than one pro/function in them (e.g.
> widget code with all the event handlers up front) with the "main" routine at the end. This
> is the *only* time I've ever had problems. My assumption that the compilation of automatic
> structure definition source files (the XXX__define type) is handled in the same way as
> other multi-pro/function source files is apparently wrong. If so, I wonder what bright
> spark decided that that would be a good idea?

Well, I'm a little confused, too. But I've been burned too
many times to jump on the bash the good folks at RSI
bandwagon just yet. Most of the time I end up finding
something stupid in my own code. :-(

I'm not sure the notion of "compiled and resolved
in the current scope" is terribly helpful. (For one
thing, I don't even know what it means.) My understanding
of the IDL compiler is that when it encounters an unresolved
token it checks (1) to see if something by that name
is already compiled and saved in the IDL code area,
(2) for a *.sav file with the same name as the token, then
(3) for a *.pro file with the same name as the token. Failing
all this, IDL gives you the benefit of the doubt
and assigns the token to its variable list.

It certainly isn't going to satisfy 2 or 3, so we have to
assume it is not on the "compiled already" list at the
time it checks the token. The real question is "Why not?"

Given the convoluted way this function was called, and
the sort of one-thing-after-another way computer software
is written, I think a plausible explanation might be that
even though IDL has compiled the function, it hasn't yet
had a chance to write the function on its function list,
so that at the time the list is *checked*, it is not there.

I don't feel confident enough about this to bet the ranch,
but I would wager a beer or two that the answer turns out

to be something like this. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155