Subject: Re: IDL 8.0 bug -- line number of errors not given
Posted by David Fanning on Wed, 13 Oct 2010 12:43:32 GMT
View Forum Message <> Reply to Message

Paulo Penteado writes:

> The way I see it, debugging when developing the code is usually most
> helped by the default on_error (0), so that one is left at the scope
> and line of the error, and can navigate between scopes to inspect
> things.
>
> Using 2 for on_error could be appropriate only when one expects that
> the only source of problems occurring in the routine is in the
> parameters it gets passed, and every error inside the routine is being
> handled. So it would just get back to the caller, using message to
> inform the caller what was wrong with the parameters. In that
> situation, it would be less confusing for the user to know that the
> problem was with the parameters used in the routine call (through the
> message passed), instead of seeing a long sequence of errors from some
> routine called several levels below, which may not make it obvious why
> that routine call created a problem. In that sense, this is locating
> the source of the error more precisely, as it was the parameters used
> in the routine call, not some obscure code deep in some other routine,
> where the error was thrown.
>
> The trouble with 2 for on_error is when the error is unhandled. Then
> just getting some like "% Specified offset to array is out of range:
> A" is really unhelpful. In such a situation, I would argue that
> 'on_error,0' should have been used.

OK, thank you, Paulo. I think I understand this argument
in principal now, anyway. My biggest problem with it is that
I don't think this is how most people write code. I know its
not how I write code. I typically use On_Error, 2 in program
utility routines. I want *any* error in these utility routines
to be returned to the main program module, where I am catching
and reporting errors.

It's all well and good to say its clear that the
"general philosophy" of error handling is such and so.
But when you turn sometimes poorly documented software
over to real users, they use what you give them, and often
in ways you never anticipated. I think it is unfair (not
to say arrogant) to suddenly (after 20 years!) change the
rules and then claim that the use of "library" in the
documentation makes it clear what the "general philosophy"
was all along. Hell, I barely understand it when it's
spelled out for me!

I just think that if one of your goals is backward compatibility (and I am *extremely* appreciative that this is one of ITTVIS's goals), then you don't change something as fundamental as your error handling mechanism, no matter how poorly thought out you think it might have been originally from a prospective 20 years on.

I have no problems with tweaking it to do something else, but you can't fundamentally change the way it works just because you suddenly "discovered" in your own programs that it throws long error messages. Yes, it does. And there is a lot of code out there that relies on just this fact.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")
