Subject: Re: _Ref_Extra : BUG? (in Win2K 55b) corrected test file
Posted by Martin Downing on Fri, 02 Nov 2001 09:59:58 GMT
View Forum Message <> Reply to Message

----- Original Message -----
From: "Mark Hadfield" <m.hadfield@niwa.cri.nz>
>
> So the routine that's crashing your machine (and mine) is as follows
> (shortened)...
>
> pro test_ref_extra1,_REF_EXTRA=e
> test_ref_extra2, _REF_EXTRA=e ; Here be dragons!
> end
>
> This isn't merely bad practice, it's wrong. The _REF_EXTRA keyword is only
> used in routine definitions. When *calling* a routine, the acceptable
forms
> are "_EXTRA" and (as of 5.5) "_STRICT_EXTRA".
>
> Still, it shouldn't crash IDL. It should be detected as a syntax error.
>
Hi Mark

It seemed logical to me that if arguments have to be passed through many preprocessing routines to a later routine, that the best way would be by reference to avoid copying (as would be the case if the Keyword was specified), if the final routine insists on a local copy (by value) why should that matter?
Well I guess I just didnt understand the manual (no change there!). Are you saying that this is determined purely by the definition, and that calling with _REF_EXTRA as below
> test_ref_extra2, _REF_EXTRA=e ; Here be dragons! (indeed)
is supposed to be a syntax error? That was just not clear to me from the help (and for that matter the programmer who wrote the syntax parser!!!)
I am concerned as to which causes passing by reference/value, the definition or the call?
This is almost as horrible as C++ inheritance simple in principle but boy can some strange things happen when things get complicated - for now I guess will have to stick to _extra which at least is easy to understand :(

roll on coffee time!

cheers

Martin

Help on Keyword Inheritance from 5.4 follows:

"Choosing a Keyword Inheritance Mechanism

The "pass by reference" (_REF_EXTRA) keyword inheritance mechanism was introduced in IDL version 5.1, and in many cases is a good choice even if values are not being passed back to the calling routine. Because the _REF_EXTRA mechanism does not create an IDL structure to hold the keyword/value pairs, overhead is slightly reduced. Two situations lend themselves to use of the _REF_EXTRA mechanism:

1. You need to pass the values of keyword variables back from a called routine to the calling routine.
2. Your routine is an "inner loop" routine that may be called many times. If the routine is called repeatedly, the savings resulting from not creating a new IDL structure with each call may be significant.

It is important to remember that if the routine that is passing the keyword values through also needs access to the values of the keywords for some reason, you must use the "pass by value" (_EXTRA) mechanism.

Note - Updating existing routines that use _EXTRA to use _REF_EXTRA is relatively easy. Since the called routine uses _EXTRA to receive the extra keywords in either case, you need only change the _EXTRA to _REF_EXTRA in the definition of the calling routine.

By contrast, the "pass by value" (_EXTRA) keyword inheritance mechanism is useful in the following situations:

1. Your routine needs access to the values of the extra keywords for some reason.
2. You want to ensure that variables specified as keyword parameters are not changed by a called routine.

Example: Keywords Passed by Value

One of the most common uses for the "pass by value" keyword inheritance mechanism is to create "wrapper" routines that extend the functionality of existing routines. In most "wrapper" routines, there is no need to return

set of keywords available to the existing routine in the wrapper routine.