Subject: Re: One RETALL is not enough
Posted by Jim Pendleton on Fri, 02 Nov 2012 03:41:32 GMT
View Forum Message <> Reply to Message

On Friday, October 26, 2012 2:26:22 PM UTC-6, wlandsman wrote:
> While debugging a program, I've been getting error messages after a RETALL like the following:
>
>
>
> IDL> retall
>
> % Invalid pointer: <POINTER (<PtrHeapVar2858>)>.
>
> % Execution halted at: XYZ_DEFAULTS::CLEANUP 456
>
> IDL> retall
>
> % Invalid pointer: <POINTER (<PtrHeapVar2578>)>.
>
> % Execution halted at: XYZ_DEFAULTS::CLEANUP 456
>
> IDL> retall
>
> % Temporary variables are still checked out - cleaning up...
>
> IDL> retall
>
>
>
> So one RETALL is not enough to get a normal return , but if I give four RETALLs then there is enough of an extra "push" to give a normal return ;-) I first thought this was just a timing problem, and that the pointer cleanup wasn't complete at the time of the first RETALL, but it was complete by the time of the fourth RETALL. But the errors always appear in the same pattern as above, requiring 4 RETALLs no matter how much time I give. Any suggestions as to what is happening? Thanks, --Wayne
>
>
>
> P.S. Line 456 where the first errors occurs is the following.
>
>
>
> IF OBJ_VALID(self.files.class.Revclasshash) THEN OBJ_DESTROY, self.files.class.Revclasshash
>
>
>

> where 'files' and 'class' are structures, and Revclasshash is an object

At the risk of double-posting a reply...

The need for multiple RETALLs in the context of objects usually means the ::CLEANUP for one or more objects lacks a CATCH block or an ON_ERROR, 2 and the ::CLEANUP fails to handle unexpected error conditions.

I'm a big fan of adding CATCH statements that at the least throw HELP, /LAST_MESSAGE output, at least during the development of code.

Since RETALL isn't the "happy path" for object destruction, there's the potential for circular heap references and other pathologies that were unanticipated by the class' original programmer.

For the illustrative purposes, introduce an error into your own cleanup method in a test object, then execute a RETALL during execution from somewhere in code that uses an object of this class.

The original posting appears to indicate that the contents of a HASH at the time the destructor was executed included some "bad" pointer references internally. Whether these are references to pointers that are used only internally by the hash or if they are references to "user mode" pointers that have been stored in the hash by user code is unclear. A repeatable case that can be sent to Exelis VIS (support@exelisvis.com) would be helpful if the issue can be tracked to a pathology in the internal IDL class cleanup code.
