Subject: Re: How to traverse/inquire a class object structure in IDL?
Posted by Pavel Romashkin on Wed, 13 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Paul,
I see no flaw (at least on my programming level) in what you did. Why can't you just do the loop 13 times (this is the number of fields in your object structure, and it will never change since it is a named structure)? The method is tied to the type and 13 will always be OK, unless you are trying to write a generic method. If you want a generic method (that would wipe out pointers in any pointer-containing class), there is one way I saw it could be done. If you call "help, self, /object, /output=report" inside the cleanup method, then it returns also local information on the instance of that object, for example (I made a new method called "names"):

```
a -> names
** Object class NASTI, 0 direct superclasses, 1 known method
   Known Procedure Methods:
   NASTI::NAMES
   Instance Data:
     ** Structure NASTI, 13 tags, length=52:
     WAVENUMBER      POINTER   <NullPointer>
     RADIANCE        POINTER   <NullPointer>
     ALTITUDE        POINTER   <NullPointer>
     FOV_ANGLE       POINTER   <NullPointer>
     FOV_INDEX       POINTER   <NullPointer>
     LATITUDE        POINTER   <NullPointer>
     LONGITUDE       POINTER   <NullPointer>
     AIRCRAFT_ROLL   POINTER   <NullPointer>
     AIRCRAFT_PITCH  POINTER   <NullPointer>
     SCAN_LINE_INDEX POINTER   <NullPointer>
     DATE            POINTER   <NullPointer>
     TIME            POINTER   <NullPointer>
     DECIMAL_TIME    POINTER   <NullPointer>
```

Then you could programmatically inspect REPORT and find the number of tags in the object instance. Then, clean it up with a loop, checking for the type of the field being a pointer "if size(self.(i), /type) eq 10 then ptr_free, self.(i)".
I don't use objects of my own; so far I was happy with those IDL comes with. So, don't blame me if my approach is awkward. I just tried to come up with a solution. Maybe pros like David can give a much more elegant solution.
Good luck,
Pavel

> Not good. As more objects are created and destroyed, the valid pointer list
> grows. I would like to do the following in a CLEANUP method:
>
> FOR i = 0, n_object_structure_elements - 1 DO $

> IF (PTR_VALID(self.(i))) THEN $
> PTR_FREE, self.(i)
>
> that is, *explicitly* free up the pointers. This works great if I have a value
> for n_object_structure_elements.
