

---

Subject: Re: Asynchronous IDL\_IDLBridge causing memory leak  
Posted by [Seth Johnson](#) on Tue, 31 Aug 2010 15:31:59 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On Aug 30, 10:02 am, Seth Johnson <seth.spjoh...@gmail.com> wrote:  
> On Aug 30, 8:35 am, Bennett <juggernaut...@gmail.com> wrote:  
>  
>  
>  
>> On Aug 27, 2:39 pm, Seth Johnson <seth.spjoh...@gmail.com> wrote:  
>  
>>> Sorry, I realized there was a mistake in the second example, it should  
>>> be:  
>  
>>> oBridge=OBJARR(5)  
>>> FOR chain=0,4 DO BEGIN oBridge[chain]=Obj\_New('IDL\_IDLBridge')  
>  
>>> FOR i=0,999 DO BEGIN  
>>> FOR chain=0,4 DO BEGIN  
>>> a=bindgen(1E4,1E3)  
>>> oBridge[chain]->SetVar,'a',a  
>>> oBridge[chain]->Execute,'a=a+a',/NOWAIT  
>>> ENDFOR  
>  
>>> FOR chain=0,4 DO WHILE oBridge[chain]->Status() NE 0 DO wait,0.0001  
>>> ENDFOR  
>>> OBJ\_DESTROY,oBridge  
>  
>>> I do not destroy the objects until the very end as there are  
>>> parameters and routines that need to be loaded into each IDL\_IDLBridge  
>>> for various computations in addition to parameters that change with  
>>> every iteration. Destroying and recreating would be a rather large  
>>> boon to processing time while the initial problem caused by  
>>> asynchronous operation still remains.  
>  
>> I've noticed that leak in 6.3 but not in 7.0+. Which version are you  
>> running?  
>  
> Strange, I have tested this on IDL versions 7.0 and 7.1, both of which  
> produce the leak. Could the cause perhaps lie in the setup or one of  
> the required packages? I have noticed while testing on different  
> machines that 7.0 and 7.1 use different versions of the shared library  
> libstdc++.so.

It is not the most elegant of solutions, but I have found a temporary work around for the memory leak. Rather than calling the asynchronous processes from the main routine, I create a single child process that then creates its own children and performs the asynchronous calls

similar to:

```
oBridge=Obj_New('IDL_IDLBridge')
oBridge->SetVar,'a',a
oBridge->Execute,"oBridge=Obj_New('IDL_IDLBridge')"
oBridge->Execute,"oBridge->SetVar,'a',a"
FOR i=0,999 DO BEGIN
    tmp=memory()
    oBridge->Execute,"oBridge->Execute,'a=a+a',/NOWAIT"
    print,memory(/high)
    WHILE oBridge->GetVar('oBridge->Status()') NE 0 DO wait,0.0001
ENDFOR
```

The child process (and its children) do not appear to leak memory as the parent call does. I find it rather peculiar that this method works, even after loading the IDL startup file into the child processes.

---