Subject: Re: testing IDL_IDLBridge status
Posted by Jim Pendleton on Fri, 06 Nov 2015 02:02:28 GMT
View Forum Message <> Reply to Message

On Thursday, November 5, 2015 at 6:32:24 AM UTC-7, superchromix wrote:
> After further investigation, it seems that there is really no way to do this. When the application object is busy, even though the status of the bridge object remains 0 or 2 (Idle or Command completed), the bridge object will not respond to the Execute method. I need to use Execute in order to query the status of the application.
>
> For example: (pseudocode)
>
> ---
>
> my_bridge = obj_new('IDL_IDLBridge')
>
> cmd = 'my_app = obj_new(''my_application'')'
> my_bridge -> Execute, cmd
>
> test_quit = 0
>
> while test_quit eq 0 do begin
>
> cmd = 'test_app_closed = ~obj_valid(my_app)'
> my_bridge -> Execute, cmd
>
> test_app_closed = my_bridge -> GetVar('test_app_closed')
>
> if test_app_closed eq 1 then begin
>
> obj_destroy, my_bridge
> test_quite = 1
>
> endif
>
> endwhile
>
> ---
>
> This doesn't work, because when the application is busy, the program gets stuck at the Execute statement. If this program is running in the main IDL process and gets stuck, then I can't do anything else in IDL while I'm waiting for the application to finish its task.
>
> If I use the NOWAIT keyword to Execute, and specify a callback procedure etc., it makes no difference. The program still gets stuck at the Execute statement.
>
> The "STATUS" property of the bridge is constantly reporting either 0 or 2 (IDLE or COMMAND COMPLETE) throughout this process. Even though the application is busy and is effectively

blocking the bridge from doing any other processing, the status reports 0 or 2. Effectively, it is impossible for my program to query the bridge without getting hung up!
>
> Therefore, it seems there is no way for my to automatically kill the IDL bridge processes when the application objects are closed. Unless, there is a way for the application object to kill the IDL bridge process on its own, from within IDL?
>
> ideas?
> thanks
> Mark

You might consider using a non-blocking bridge EXECUTE call, locking a semaphore (SEM_LOCK) on the bridge side before the execution of long-running jobs, and checking the state of the semaphore through a timer() in the main process.

It's best to stop and close bridge processes from the main IDL process that created them rather than via EXIT on the bridge side. The originating IDL process is a little touchy about the object lifecycle and references may be left in limbo.
