Subject: Re: 6.3 reactions?
Posted by codepod on Tue, 09 May 2006 17:30:49 GMT
View Forum Message <> Reply to Message

The following code gives an idea of how this can work. Note: This isn't exact, but should outline how the IDL-IDL bridge is used.

Preconditions:
Assume you have a variable MyData that you want to use as input to a time consuming routine called MyRoutine. If you ran this in an IDL application, the application would block until processing is done. However with the bridge you can have it process in the background. The general structure of the code is the following:

```
oBridge = obj_new("IDL_IDLBridge")    ; creates the child process

;; Put the data in the child process session. This creates
;; a variable called Data in the child process and copies the
;; value of MyData into it. This exchange uses shared memory
;; to transfer values, but the actual data memory is not shared
;; between each process. (2 copies exist).
oBridge->SetVar, "Data", MyData

;; Lauch background processing
 oBridge->Execute, "result = MyRoutine( Data )", /nowait

 ;; This call returns immediately

 ;; At this point this work will execute in the background
 ;; process. At a later time you can check and see if the
 ;; task is completed
 iStatus = oBridge->Status()
 if(iStatus eq 2)then begin ;; execution is completed
    Result = oBridge->GetVar("Result") ;; get the result
 endif

 ;; When you are completed with the process, just destroy the object
 obj_destroy, oBridge
```

This simple example shows a "polling" method to check when the task is completed. The object also has a callback methodology which operates similar to an event callback. It will call a routine you specify when an action happens (command done, error signaled).

Hopefully this helps.

-CP