Subject: Re: DLM's and C code
Posted by Dominik[1] on Thu, 10 Jan 2002 08:44:27 GMT
View Forum Message <> Reply to Message

Hi Richard,

I do it a little bit easier way. If an error occurs in my C routine (in a DLL) I return with an error code, lets say:
#define ERROR_DIVISION_BY_ZERO -128
return ERROR_DIVISION_BY_ZERO;

In IDL I can check the return value

status = call_external()
if status EQ -128 then...

This works really good for me. The calculations which the C routine is doing for me, is written into a piece of memory and can then be seen by IDL. Therefor I create a variable in IDL, pass it to my DLL by reference (I think it is the normal way to pass it by referenze), can manipulate the value and on returning to IDL, the variable will hold the new value.

Hope it helps you
Dom

"Richard Tyc" <Richard_Tyc@sbrc.umanitoba.ca> schrieb im Newsbeitrag news:a1fsur$78a$1@canopus.cc.umanitoba.ca...
> A somewhat IDL related question.
> I am trying to link in some C code via a DLM. I use a wrapper routine to
> handle the call from IDL and manipulate the args and return data. Within the
> wrapper, I call C functions linked in through another DLL.
>
> What is the best way to handle errors while deeply nested within layers of C
> functions.? The ANSI C code I am using essentially had exit(1) calls for
> major errors. Is there an IDL_ function (like say an exit handler) I can
> call to cleanly return to IDL rather than a trying to modify the call stack
> and get back to the IDL wrapper function to perform something like a return
> IDL_StrToSTRING("ERROR") ;
>
> Thanks
>
> --
> Richard Tyc

> Project Engineer
> St. Boniface Hospital Research Center
> 351 Tache Ave
> Winnipeg, MB
> Canada
> Tel: 204-237-2557
> Fax: 204-231-0485
> Email: richt@sbrc.umanitoba.ca
>
>

---