Subject: Re: Call external 64bit
Posted by Foldy Lajos on Mon, 13 Jul 2009 14:56:47 GMT
View Forum Message <> Reply to Message

On Mon, 13 Jul 2009, Wox wrote:

> Hi
>
> Calling a 32bit dll from IDL 64 bit doesn't work (see below). Is this
> normal behaviour?
>
> IDL> print,!version
> { x86_64 Win32 Windows Microsoft Windows 7.1 Apr 21 2009 64 64}
> IDL> print,call_external('test.dll','test',2u,/ALL_VALUE,/I_VALUE)
> % CALL_EXTERNAL: Error loading sharable executable.
> Symbol: test, File = Test.dll
> %1 is not a valid Win32 application.
>
>
> Normal behaviour (IDL 32bit):
> IDL> print,!version
> { x86 Win32 Windows Microsoft Windows 7.1 Apr 21 2009 32 64}
> IDL> print,call_external('test.dll','test',2u,/ALL_VALUE,/I_VALUE)
> 10
>
>
> Dll file used: http://xrdua.ua.ac.be/public/Test.dll

I think this is normal. You can not mix 32 and 64 bit code in a single app
(eg. call_external passes 64 bit pointers to the DLL while the DLL expects
32 bit ones).

regards,
lajos

Subject: Re: Call external 64bit
Posted by Toolbox on Mon, 13 Jul 2009 19:10:04 GMT
View Forum Message <> Reply to Message

On Jul 13, 8:56 am, FÖLDY Lajos <fo...@rmki.kfki.hu> wrote:
> On Mon, 13 Jul 2009, Wox wrote:
>> Hi
>
>> Calling a 32bit dll from IDL 64 bit doesn't work (see below). Is this
>> normal behaviour?
>

>> IDL> print,!version
>> { x86_64 Win32 Windows Microsoft Windows 7.1 Apr 21 2009 64 64}
>> IDL> print,call_external('test.dll','test',2u,/ALL_VALUE,/I_VALUE)
>> % CALL_EXTERNAL: Error loading sharable executable.
>> Symbol: test, File = Test.dll
>> %1 is not a valid Win32 application.
>
>> Normal behaviour (IDL 32bit):
>> IDL> print,!version
>> { x86 Win32 Windows Microsoft Windows 7.1 Apr 21 2009 32 64}
>> IDL> print,call_external('test.dll','test',2u,/ALL_VALUE,/I_VALUE)
>> 10
>
>> Dll file used:http://xrdua.ua.ac.be/public/Test.dll
>
> I think this is normal. You can not mix 32 and 64 bit code in a single app
> (eg. call_external passes 64 bit pointers to the DLL while the DLL expects
> 32 bit ones).
>
> regards,
> lajos

There is an undocumented keyword in IDL 7.1 (and a few minor versions earlier) on the idl_idlbridge object that will allow you to do what you want. (Of course, all caveats apply.) The keyword is "ops" and it lets you set the bit mode of the slave OPS process that loads idl that you intereact with via the idl_idlbridge object. By default, the slave OPS uses the same bit mode as the master IDL process that created it. But, the undocumented "ops" keyword let you override it. For example, you can have the following combinations:

master: 64-bit slave: 32-bit
master: 32-bit slave: 64-bit

Usage: o = obj_new('idl_idlbridge',ops='32') ; create a 32-bit slave idl process

You can then use the 'setvar' & 'getvar' methods on the newly created idl_idlbridge object to marshal data between the different bit mode processes.

So, for your example, you could do the following from within the master, 64-bit IDL process:

IDL> o32 = obj_new('idl_idlbridge',ops='32')
IDL> o32->execute, "dlm_load, 'test' "
IDL> o32->execute, "a = test_function(42)"
IDL> local_a = o32->getvar('a')

This would load your 32-bit 'test' DLM functionality into the 32-bit slave process. Call a function in your 32-bit DLM called 'test_function' that creates a variable named 'a' at the $MAIN scope level in your 32-bit slave process. Then, marshal variable 'a' back to the 64-bit master process using 'getvar'.

You can also create as many mixed 32-bit and 64-bit slave processes from the same master process as you want. This would let you do things like test the 64-bit port of your old 32-bit DLM at the same time by creating a 32-bit slave process and 64-bit slave process and sending commands to each and verify the results. For example:

```
IDL> oOld = obj_new('idl_idlbridge',ops='32') ; creates a 32-bit slave
process
IDL> oNew = obj_new('idl_idlbridge',ops='64') ; creates a 64-bit slave
process
IDL> oOld->execute, 'aOld = dosomething()'    ; execute 32-bit code
IDL> oNew->execute, 'aNew = dosomething()'    ; execute 64-bit code
IDL> local_aOld = oOld->getvar('aOld')
IDL> local_aNew = oNew->getvar('aNew')
IDL> bEqual = compare_values(local_aOld, local_aNew)
```

Don't try using the "ops" keyword on a run-time distribution: only on a full installation.

Regards,
Toolbox

Subject: Re: Call external 64bit
Posted by Toolbox on Mon, 13 Jul 2009 19:36:36 GMT
View Forum Message <> Reply to Message

On Jul 13, 1:10 pm, Toolbox <tool....@live.com> wrote:
> On Jul 13, 8:56 am, FÖLDY Lajos <fo...@rmki.kfki.hu> wrote:
>
>
>
>> On Mon, 13 Jul 2009, Wox wrote:
>>> Hi
>
>>> Calling a 32bit dll from IDL 64 bit doesn't work (see below). Is this
>>> normal behaviour?
>
>>> IDL> print,!version
>>> { x86_64 Win32 Windows Microsoft Windows 7.1 Apr 21 2009 64 64}
>>> IDL> print,call_external('test.dll','test',2u,/ALL_VALUE,/I_VALUE)

>>> % CALL_EXTERNAL: Error loading sharable executable.
>>> Symbol: test, File = Test.dll
>>> %1 is not a valid Win32 application.
>
>>> Normal behaviour (IDL 32bit):
>>> IDL> print,!version
>>> { x86 Win32 Windows Microsoft Windows 7.1 Apr 21 2009 32 64}
>>> IDL> print,call_external('test.dll','test',2u,/ALL_VALUE,/I_VALUE)
>>> 10
>
>>> Dll file used:http://xrdua.ua.ac.be/public/Test.dll
>
>> I think this is normal. You can not mix 32 and 64 bit code in a single app
>> (eg. call_external passes 64 bit pointers to the DLL while the DLL expects
>> 32 bit ones).
>
>> regards,
>> lajos
>
> There is an undocumented keyword in IDL 7.1 (and a few minor versions
> earlier) on the idl_idlbridge object that will allow you to do what
> you want. (Of course, all caveats apply.) The keyword is "ops" and
> it lets you set the bit mode of the slave OPS process that loads idl
> that you intereact with via the idl_idlbridge object. By default, the
> slave OPS uses the same bit mode as the master IDL process that
> created it. But, the undocumented "ops" keyword let you override it.
> For example, you can have the following combinations:
>
> master: 64-bit   slave: 32-bit
> master: 32-bit   slave: 64-bit
>
> Usage: o = obj_new('idl_idlbridge',ops='32') ; create a 32-bit slave
> idl process
>
> You can then use the 'setvar' & 'getvar' methods on the newly created
> idl_idlbridge object to marshal data between the different bit mode
> processes.
>
> So, for your example, you could do the following from within the
> master, 64-bit IDL process:
>
> IDL> o32 = obj_new('idl_idlbridge',ops='32')
> IDL> o32->execute, "dlm_load, 'test' "
> IDL> o32->execute, "a = test_function(42)"
> IDL> local_a = o32->getvar('a')
>
> This would load your 32-bit 'test' DLM functionality into the 32-bit
> slave process. Call a function in your 32-bit DLM called

> 'test_function' that creates a variable named 'a' at the $MAIN scope
> level in your 32-bit slave process. Then, marshal variable 'a' back
> to the 64-bit master process using 'getvar'.
>
> You can also create as many mixed 32-bit and 64-bit slave processes
> from the same master process as you want. This would let you do
> things like test the 64-bit port of your old 32-bit DLM at the same
> time by creating a 32-bit slave process and 64-bit slave process and
> sending commands to each and verify the results. For example:
>
> IDL> oOld = obj_new('idl_idlbridge',ops='32') ; creates a 32-bit slave
> process
> IDL> oNew = obj_new('idl_idlbridge',ops='64') ; creates a 64-bit slave
> process
> IDL> oOld->execute, 'aOld = dosomething()' ; execute 32-bit code
> IDL> oNew->execute, 'aNew = dosomething()' ; execute 64-bit code
> IDL> local_aOld = oOld->getvar('aOld')
> IDL> local_aNew = oNew->getvar('aNew')
> IDL> bEqual = compare_values(local_aOld, local_aNew)
>
> Don't try using the "ops" keyword on a run-time distribution: only on
> a full installation.
>
> Regards,
> Toolbox

Update: If you get an error using quotes around the "ops" keyword value, e.g. ops='32', try it without the quotes, e.g. ops=32

Subject: Re: Call external 64bit
Posted by Wout De Nolf on Tue, 14 Jul 2009 08:41:05 GMT
View Forum Message <> Reply to Message

On Mon, 13 Jul 2009 12:36:36 -0700 (PDT), Toolbox <tool.box@live.com>
wrote:

>> There is an undocumented keyword in IDL 7.1 (and a few minor versions
>> earlier) on the idl_idlbridge object that will allow you to do what
>> you want. ï¿½(Of course, all caveats apply.) ï¿½The keyword is "ops" and
>> it lets you set the bit mode of the slave OPS process that loads idl
>> that you intereact with via the idl_idlbridge object.

Nice! Thanks a lot.
