Subject: Re: First catch of the day
Posted by Haje Korth on Wed, 12 Jan 2005 14:23:03 GMT
View Forum Message <> Reply to Message

Robert,
The library msvcrt.lib should be included in your MS Visual C++
installation. On my machine it is in "C:\Program Files\Microsoft Visual
Studio .NET 2003\Vc7\lib". The def files are pretty trivial. They look like
this:
LIBRARY IDL_SPHINT

EXPORTS IDL_Load @1

Hope this helps,
Haje

"Robert Barnett" <retsil@zipworld.com.au> wrote in message
news:41E4D6CA.4030908@zipworld.com.au...
> Does anyone know how to get around this one?
>
>
> I have had problems running IDL processes using both Watsyn and the IDL
> Broker on Siemens E.Soft workstations. The E.Soft application runs in
> full-screen mode obscuring the IDL windows. I've been trying to work out
> a way to keep the IDL windows 'always on top'.
>
> To solve this, I'm using CALL_EXTERNAL to access MS-Windows functions
> from IDL. The particular library is C:\WINNT\system32\user32.dll which
> is common over windows 98/NT/2000/XP.
>
> I create a wrapper as recommended in the IDL documentation
> result =
>
CALL_EXTERNAL('C:\WINNT\system32\user32.dll','GetForegroundW indow',WRITE_WRA
PPER='user32GetForegroundWindow.c')
>
> I successfully compiled using Microsoft Visual C++ Toolkit 2003.
> cl -D_DLL -DWIN32 -D_MT /nologo /I"D:\RSI\IDL60\external\include" /c
> "GetForegroundWindow.c" /Fo"GetForegroundWindow.obj"
>
> However, I experience problems when attempting to link the compiled file
> into a DLL. The contents of the generated C program
> (GetForegroundWindow.c) indicate that I should use the following command:
>
> link /out:"GetForegroundWindow.dll" /nologo /nodefaultlib /dll
> "GetForegroundWindow.obj" /def:"GetForegroundWindow.def"

> "D:\RSI\IDL60\bin\bin.x86\idl32.lib" msvcrt.lib kernel32.lib
>
> It appears that only kernel32.lib is included with Microsoft Visual C++
> yet msvcrt.lib is not. Is this library really required? Do you know
> where I can obtain this library from?
>
> The linking also fails because there is no GetForegroundWindow.def file.
> This is explained in the documentation of the
generatedGetForegroundWindow.c
> * Note that these 2 commands are probably not quite enough: You
> * will also need to supply a linker options file to control which
> * symbols are exported. Under Unix, this file has various names
> * (Solaris calls it a mapfile). Microsoft Windows calls it a def file.
> * Read your systems linker documentation for further information.
> Do you know where I can obtain documentation on how to generate this
> .def file?
>
> Any assistance would be appreciated and would assist me in making the
> IDL product usable in the supplied environment.
>
>
> --
>
> nrb@
> Robbie Barnett
> imag
> Research Assistant
> wsahs
> Nuclear Medicine & Ultrasound
> nsw
> Westmead Hospital
> gov
> Sydney Australia
> au
> +61 2 9845 7223
>

Subject: Re: First catch of the day
Posted by Peter Mason on Wed, 12 Jan 2005 22:18:10 GMT
View Forum Message <> Reply to Message

Robert Barnett wrote:
> Does anyone know how to get around this one?
>
>
> I have had problems running IDL processes using both Watsyn and the
> IDL Broker on Siemens E.Soft workstations. The E.Soft application

> runs in full-screen mode obscuring the IDL windows. I've been trying
> to work out
> a way to keep the IDL windows 'always on top'.
>
> To solve this, I'm using CALL_EXTERNAL to access MS-Windows functions
> from IDL. The particular library is C:\WINNT\system32\user32.dll which
> is common over windows 98/NT/2000/XP.
>
> I create a wrapper as recommended in the IDL documentation
> result =
>
CALL_EXTERNAL('C:\WINNT\system32\user32.dll','GetForegroundW indow',WRITE_WRA
PPER='user32GetForegroundWindow.c')
>
> I successfully compiled using Microsoft Visual C++ Toolkit 2003.
> cl -D_DLL -DWIN32 -D_MT /nologo /I"D:\RSI\IDL60\external\include" /c
> "GetForegroundWindow.c" /Fo"GetForegroundWindow.obj"
>
> However, I experience problems when attempting to link the compiled
> file into a DLL. The contents of the generated C program
> (GetForegroundWindow.c) indicate that I should use the following
> command:
>
> link /out:"GetForegroundWindow.dll" /nologo /nodefaultlib /dll
> "GetForegroundWindow.obj" /def:"GetForegroundWindow.def"
> "D:\RSI\IDL60\bin\bin.x86\idl32.lib" msvcrt.lib kernel32.lib
>
> It appears that only kernel32.lib is included with Microsoft Visual
> C++
> yet msvcrt.lib is not. Is this library really required? Do you know
> where I can obtain this library from?
>
> The linking also fails because there is no GetForegroundWindow.def
> file. This is explained in the documentation of the
> generatedGetForegroundWindow.c
> * Note that these 2 commands are probably not quite enough: You
> * will also need to supply a linker options file to control which
> * symbols are exported. Under Unix, this file has various names
> * (Solaris calls it a mapfile). Microsoft Windows calls it a def
> file.
> * Read your systems linker documentation for further information.
> Do you know where I can obtain documentation on how to generate this
> .def file?
>
> Any assistance would be appreciated and would assist me in making the
> IDL product usable in the supplied environment.

Robert, I've never heard of E.Soft but I doubt that it's worth the trouble using external code for this problem - unless the full-screen E.Soft window has been created with the WS_EX_TOPMOST style for some bizarre reason. (I can't imagine anyone writing such a badly-behaved program though.) You would then want to find it and deal with it.

It's hard to recommend a solution because interface issues like "which window is in the foreground" should really be left to the user. It's bad form to mess with this sort of thing. Can't you get at your IDL windows via the taskbar? Or is the E.Soft window periodically re-instating itself as the topmost window?

If your IDL application has a widget interface, here's an IDL-only approach that you might try. It goes against the grain of how a decent GUI should behave but it might be justified under the circumstances.
Attach a timer event to some "unused" widget (e.g., some label or intermediate base) in your heirarchy. (Use WIDGET_CONTROL, timer_widget_id, TIMER= for this.) You might set the timer for a couple of seconds, for example. A timer event will get sent to your event handler. When handling this event, use WIDGET_CONTROL, tlb, /SHOW, where "tlb" is the widget ID of your IDL app's top-level base. (If you have more than one window in your app then you might do this to each one.) Also, re-attach the timer trigger as IDL's TIMER= mechanism is a one-shot affair.

If you want to pursue a CALL_EXTERNAL solution then as Haje said msvcrt.lib should be there. It's the C runtime library. It's bound to be amongst your C compiler's files. First thing I'd try is to ditch the /nodefaultlib from your link. If this doesn't work then check that the environment variables are set up correctly for your compiler. (e.g., My C compiler wants a "lib" environment variable that tells it library directories.)

HTH
Peter Mason

Subject: Re: First catch of the day
Posted by Robert Barnett on Mon, 17 Jan 2005 04:23:33 GMT
View Forum Message <> Reply to Message

I think that the problem is that this E.Soft application is designed to be the only application on the system. It's really an embedded piece of software. However, both IDL and Siemens market it as though it is some kind of user firendly development environment.

This tweak is provided on an as-is basis and
comes with absolutely no warranty

I have managed to partially solve the problem by using the "Minimize All" feature in Internet Explorer. I make a batch file with the following code

```
explorer ToggleDesktop.scf
sleep 2
%IDL_DIR%\bin\bin.x86\idlrt.exe %APPLICATION%
```

I create a file called ToggleDesktop.scf with the following contents

```
[Shell]
Command=2
IconFile=explorer.exe,3
[Taskbar]
Command=ToggleDesktop
```

The usual result of when I run the batch file is that all windows are minimsed and then my application is exceuted and visible on the desktop. However, occasionally ESoft STILL manages to fight windows explorer. I notice a subtle flicker as the ESoft application detects that it has been minimized and then restores itself.

Another little trick I've found is that I can call MFC functions without creating a wrapper if no arguments are required. This is very useful for detecting if E.Soft has pushed itself forward:

```
if (!VERSION.OS eq 'Win32') then begin
                        ; For Windows boxes. If the foreground
                        ; window doesn't change after /realize
                        ; has been called then the widgets are
                        ; probably not visible
hwnd_current = $
 CALL_EXTERNAL('C:\WINNT\system32\user32.dll','GetForegroundW indow')
;
; Widget code for realizing the window
;
if (hwnd_current eq $
 CALL_EXTERNAL('C:\WINNT\system32\user32.dll','GetForegroundW indow')) $
then begin $
;
; What to do if the window wasn't realized? I guess that's up to you
;
endif
endif else begin
;
; What to do if we're not in windows
;
```

endelse

Lastly,

I've concluded that it is impossible to use the "Microsoft Visual C++ Toolkit 2003" to develop in tandem with IDL. The IDL wrappers require the use of libraries not included in this package. I am looking into purchasing "Visual C++ .NET 2003 Standard".

--

nrb@
Robbie Barnett
imag
Research Assistant
wsahs
Nuclear Medicine & Ultrasound
nsw
Westmead Hospital
gov
Sydney Australia
au
+61 2 9845 7223
