Erik Stevenson

From: johnlu
To: bradsi; denisg; karlst
Subject: RE: Slimy things that Visual C++ does
Date: Wednesday, April 07, 1993 4:33PM

i have checked with all the obvious candidates on my team -- thomasf, jimmo, jimh, gregj, edst -- none of them know anything about this. denis, who gave you the source and consulted with you on this design?

> ---
> From: David Maritz
> To: David Cole (davidcol)
> Cc: Brad Silverberg; David Maritz; John Ludwig; Denis Gilbert (denisg); Paul
> Maritz (paulma)
> Subject: FW: Slimy things that Visual C++ does
> Date: Tuesday, April 06, 1993 8:53AM
>
> Below Raymond describes a very bad thing done in VC++. This breaks every rule in the book and was done knowing it would cause compatibility problems.
> Is there no way that internally we can penalize the people that do this sort
> of thing or made the decision to do this. Fine them heavily or some other punishment to discourage it in the future.
>
> Thx - David
>
> ---
> From: raymondc
> To: davidma
> Subject: Slimy things that Visual C++ does
> Date: Tuesday, April 06, 1993 2:09AM
>
> Per your request. Bear in mind that my brain is zonked right now, so my choice of words may not be entirely diplomatic.
>
> Background
>
> Under Windows 3.1, WinOldAp (ring 3) communicates with ring 0 through
> an officially private interface with the Shell VxD. USER and Control
> Panel also communicate with the Shell VxD through this private interface.
>
> Under Cougar, the mechanism for this communication changed greatly, in anticipation of a potential stand-alone DOS 7 product. (The Windows-specific shell services were moved into a separate VxD called
> WShell.) Although the plans for stand-alone DOS 7 have apparently retreated into dormancy, the separation of powers between the Shell and WShell nevertheless remains in Chicago.
>
> What Visual C++ Does
>
> In order to create its build VM, Visual C++ uses one of the undocumented private function calls that was used by WinOldAp 3.1

to talk to the Shell VxD. The call it makes is the "Create a VM for me" call.

Under Chicago, this function call no longer works, due to the changes described above.

What Happens As A Result

Chicago Build 40e fails the request as an "unrecognized function". Visual C++ gets bent out of shape, and the end user is baffled.

What I Did To Fix It, Pass 1

I hacked up the Shell and WShell VxDs to recognize the old-style (Windows 3.1) Shell function call, so that Visual C++'s illegal function call still creates a VM like in Windows 3.1.

What Happened

The VM is created just fine, but all your DOS boxes are hung.

Why That Happened

The argument to the private function call is a pointer to a structure (undocumented) which contains information used by the Shell VxD to keep tabs on WinOldAp; one of the fields is the HWND of WinOldAp. The Visual C++ people insert the HWND of the Visual C++ window in this field.

Now, when I see that HWND, I assume that the caller is indeed WinOldAp 4.0 (for who else could it possibly be?), and therefore I expect it to react to prodding in a manner befitting its station.

One of the things that I expect is that when I ask WinOldAp to do something, it will do it, then notify me when it's done. Until that notification comes back, multitasking of DOS boxes is suspended.

The HWND that Visual C++ gives me is not in fact WinOldAp, but is rather Visual C++'s window. Visual C++ essentially "pretends to be WinOldAp", acknowledging messages as required.

But Chicago adds new functionality (dynamic DOS box titles, global data protection, closeable DOS boxes) which requires WinOldAp co-operation in order to succeed. Visual C++, having been written to emulate WinOldAp 3.1 and not WinOldAp 4.0, does not understand the new notifications, and hence does not acknowledge them.

What I Did To Fix It, Pass 2

I added a signature field to the DGROUP of WinOldAp, and when I receive the "Create a VM for me" call, I validate that the signature is present. If the signature is not present, I know that somebody is lying to me, and I zero out the HWND field, as a signal to the rest of WShell that the VM has no controlling WinOldAp. Elsewhere in WinOldAp, code is present which makes sure not to send notifications on a VM which has no controlling WinOldAp.

What They Should've Done

They should've created their VM the way all other ISVs do.

By creating a PIF and running it, a True Blue copy of WinOldAp 4.0 is placed in charge of the VM, and all continues to run well.

What I've Learned

Apparently, somebody in the Winball group gave the Visual C++ people the source code to WinOldAp 3.1. The Visual C++ program manager I've been in contact with mentioned the name of a function in WinOldAp 3.1, correct even down to the bizarre capitalization. In fact, it seems that the Windows people he interacted with even helped him pull this off, offering suggestions, etc. And he also says that portability was explicitly *not* one of their concerns.

Why I'm So Upset

We'd certainly not give Borland or Zortech a copy of the source code to WinOldAp. Why should we give a copy of it to MS Languages?

I have copies of all the email I've received from the Visual C++ people regarding this, if you want to see it.

--rjc