



## Re: Problem closing app with big UDT

All the array sizes are multiples of 4, so the 4-byte alignment should be maintained throughout unless we screwed up somewhere. We didn't think of passing the whole thing as a byte array, and it's too late to go back and rework it now.

It would take about 1 hour max to re-work it

In any one app, maybe, but this dll is in production use in several different apps, in 3 different languages.

The re-work time is the re-work time is the re-work time.  
If it's one dll.

Plus the changes in the apps to read and write it as a byte array instead of a structure.

Testing is of course another matter.

You wouldn't be here if you hadn't already recognized the need. Adding more band-aids would most likely only make the inevitable worse when you're finally forced to really fix it.

Believe me, I'm really looking for a real solution, not just a bandaid to hide the access violation error, but the fix pretty much needs to be handled on the VB side only, if that is possible; more on that below

I'd like to call attention to your own words:  
"unless we screwed up somewhere"  
I don't actually see how that's in question, since /someone/ posted already that it doesn't seem to be closing correctly.

Guilty as charged <GGG>, assuming we're not hitting some kind of VB limit, which is what I originally thought was going on here. However, additional testing prompted by what you guys have posted, indicates that the trouble really is related in some way to what the .dll is doing, because if I create that big structure and work with it in VB, but don't do anything with the .dll's functions, then there's no trouble.

Unfortunately, the Delphi programmer is completely swamped with other, higher-priority jobs, so getting him to rework his .dll to pass a byte

Re: Problem closing app with big UDT

array is not in the cards right now.

So coming at it from a different direction is there any way I can, just in my VB app, send and receive the structure I need without actually passing it as a structure? Maybe using copymem to handle it as a byte array, maybe? I think I've seen that kind of thing before, but can't seem to locate it right now.

I do have one truly \*horrible\* idea for you though.  
It's nothing but a further cover-up.  
Be aware that it doesn't fix anything, though it might \*hide\* it.  
And at the very real potential cost of causing /other/ crashes later.  
We've already told you how you can probably cure the problem.  
But if all you want now is a band-aid to cover it up, then  
you might as well try the mother-of-all-VB-band-aids:

<http://obob.com/cis58/evilend.html>

LOL! I've ALWAYS used the End statement, and it's blowing up when it hits it. Just for grins, I've tried removing it, and there's no change in what happens.

As for "cleaning up", how can I de-allocate a structure once it's created? I've tried making it into an array (dim aStruc(0 to 0) as MyStruc), but that didn't help either, even though I did an "Erase aStruc" before exiting.

—

Remove the ns\_ from if replying by e-mail (but keep posts in the newsgroups if possible).

.