Can I stop a COM dll from displaying forms?

Asked 15 years, 9 months ago Modified 15 years, 9 months ago

Viewed 212 times

2

To be more specific:

We have a web service (written in .Net) which utilizes a large number of COM dlls for core business logic (written in VB6). Assume for now that these are sealed libraries.

Unfortunately, many of these COM libraries implement their own error handling or validation messages; if we pass in data which isn't complete, or if another COM dependency is missing, they'll display a dialog box. These dialog boxes are the problem...when running in IIS, these message boxes hang the request, waiting for the user to click ok to a box that can't be seen.

Does anyone know a way to trap these UI calls (potentially it could be a form.show instead of a messagebox) on the .Net side, so we can throw an exception instead?

c# .net com vb6 interop

Share

asked Mar 18, 2009 at 13:49

Improve this question

Follow

mdryden
995 ● 1 ● 9 ● 12

3 Answers

Sorted by: Highest score (default)

2

With the caveat that I haven't done this personally, I believe the intercept and redirect you desire could be achieved through Win32 Hooks

Share Improve this answer

Follow

answered Mar 18, 2009 at 13:57

cmsjr
59k ● 11 ● 74 ● 63

I think this is the best solution if he does not want to dig into COM dlls code, but it needs more work!! – Ahmed Mar 18, 2009 at 14:00

I agree, will update this evening when I have my reference library at hand. – cmsjr Mar 18, 2009 at 14:20

Probably not more work, actually. The legacy Dlls are part of a 10+ year old insurance app, with individual Dlls for each company/rating date and region. There's somewhere in the neighbourhood of 10,000 libraries! – mdryden Mar 18, 2009 at 14:22

10,000 libraries!!! That's insane! Surely it could be made data-driven? I.e. many times fewer DLLs that looked up company/rating/region data from some central location? mrdryden, I'm sure you've thought of that - any hope? – MarkJ Mar 19, 2009 at 18:52

2

Display of forms is the least of your problems if these were written in VB6 to be used in a desktop application. These COM objects probably don't expect to be accessed by multiple threads at the same time, either. This can blow up very spectacularly or very subtly.

If management are depending on this to work for 10,000 libraries, then you need to make them understand that violating the assumptions of 10,000 pieces of ten year-old code is not a good idea.

If management is old enough (and in the US), then remind them of the old margerine commercials about "It's not *nice* to fool Mother Nature". Bad Things could happen.

I think I need to be more specific about "Bad Things".

I'm assuming these were VB6 COM objects created to interact with a VB6 forms application or applications. These COM objects could then reasonably assume there was only one thread at a time accessing them, only one user at a time accessing them, and in fact only one thread and one user accessing them during their entire lifetime.

As soon as you run them in a server, you violate their assumptions.

Who can tell what that will cause? Nobody will have done that analysis because they were basic (and valid) assumptions! Will the code maybe cache something in thread-local storage? That would have worked for the original scenario. Maybe Shared data are used to cache information. It needs to be interlocked if it wil be used by multiple threads, and then you'll have to hope the information doesn't vary per-user, because the different threads may be running on behalf of different users.

I once was told to go fix a memory leak. Long story short, it wasn't a memory leak. It was a piece of legacy, unmanaged code that assumed it was running in a desktop or batch application. In a web service, it was spewing garbage all over the heap. Therefore, it was throwing exceptions all over the place, and causing other, unrelated code to do the same. Unmanaged exceptions don't come with much detail, so it was impossible to see what had caused the problem, or how to solve it.

In this case, I was able to simply put an interlock around all access. This was good enough because this code assumed a batch environment, not an interactive environment. This might not be enough if your COM objects are making assumptions that their ten year-old requirements haven't changed out from under them.

If the COM objects can all run under a single user identity, that saves you one piece of grief. Beyond that, you may simply have to make sure there is only one instance of a given object at a time, and that all access to it is serialized. Use a SyncLock statement in VB.NET for this.

The ultimate solution would be to do a "Test-driven port" of the code. I would use the existing codebase to create automated unit tests (using vbunit, maybe). Once a piece of code has adequate unit test coverage, you can port that code (still as a COM object) to VB.NET. The unit tests can be used to confirm that the port is still working.

Such a port may not be as tough as you may think. "Copy and paste and fix the compiler errors" works fairly well between VB6 and VB.NET. There are even tools to help out. But it's the automated tests that make this practical. Otherwise, you'd have reasonable fears of how well the port was done.

Note that the new COM objects should still be usable by the original VB6 code. That should be a test, in fact.

Note also that this would be a good opportunity to document the code being ported, so that this is less of a problem in another ten years. Just don't lose the documentation!

Share Improve this answer Follow

edited Mar 18, 2009 at 17:01

answered Mar 18, 2009 at 15:11

John Saunders
162k ● 26 ● 249 ● 402

Oh, how I wish that was an option. Management doesn't want to throw away years of rating rules, and I'm not in the position to change their minds. The threading issue is something that concerns me greatly as well. – mdryden Mar 18, 2009 at 16:40

+1. A threading nightmare. BTW there are some questions on StackOverflow about converting VB6 to VB.NET with info on additional resources (including the leading tools). You could cross-reference them. – MarkJ Mar 19, 2009 at 15:37

Thanks guys. You both have no idea how close to the mark you really are. For now, I'm going to have to try and put out this fire with hooking, but sooner or later, this option will likely turn out to be the only option. Isn't management great?
– mdryden Mar 19, 2009 at 15:56

0

Check out some of the articles by Jesse Kaplan in msdn magazine. His column is called CLR Inside Out. Look for the COM Interop articles. Issues are available for download here: http://msdn.microsoft.com/en-us/magazine/cc159440.aspx

Edit: Looks like Mr. Kaplan is only an occasional contributor to that section. It is still an excellent resource for Interop related advice.

edited Mar 18, 2009 at 15:59

answered Mar 18, 2009 at 15:25

WakeUpScreaming
199 ● 6