How to UAC elevate a COM component with .NET

Asked 16 years, 3 months ago Modified 14 years, 10 months ago Viewed 15k times

16

I've found an article on how to elevate a COM object written in C++ by calling `CoCreateInstanceAsAdmin`. But what I have not been able to find or do, is a way to implement a component of my .NET (c#) application as a COM object and then call into that object to execute the tasks which need UAC elevation. MSDN documents this as the admin COM object model.

I am aware that it is possible and quite easy to launch the application (or another app) as an administrator, to execute the tasks in a separate process (see for instance the post from Daniel Moth, but what I am looking for is a way to do everything from within the same, un-elevated .NET executable. Doing so will, of course, spawn the COM object in a new process, but thanks to transparent marshalling, the caller of the .NET COM object should not be (too much) aware of it.

Any ideas as to how I could instanciate a COM object written in C#, from a C# project, through the `CoCreateInstanceAsAdmin` API would be very helpful. So I am really interested in learning how to write a COM object in C#, which I can then invoke from C# through the COM elevation APIs.

Never mind if the elevated COM object does not run in the same process. I just don't want to have to launch the whole application elevated; I would just like to have the COM object which will execute the code be elevated. If I could write something along the lines:

```
// in a dedicated assembly, marked with the following attributes:
[assembly: ComVisible (true)]
[assembly: Guid ("....")]

public class ElevatedClass
{
    public void X() { /* do something */ }
}
```

and then have my main application just instanciate `ElevatedClass` through the `CoCreateInstanceAsAdmin` call. But maybe I am just dreaming.

c# com uac elevation moniker

Share

Improve this question

edited Feb 10, 2010 at 7:05

asked Sep 24, 2008 at 13:13

Follow

Pierre Arnaud
10.5k ● 13 ● 82 ● 112

3 Answers

Sorted by: Highest score (default)

8

Look at Windows Vista UAC Demo Sample Code

(You also need the Vista Bridge sample for UnsafeNativeMethods.CoGetObject method)

Which gives you C# code that shows a few different ways to elevate, including a COM object

(Incomplete code sample - grab the files above)

```
[return: MarshalAs(UnmanagedType.Interface)]
static internal object LaunchElevatedCOMObject(Guid Clsid, Guid InterfaceID)
   {
   string CLSID = Clsid.ToString("B"); // B formatting directive: returns
{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}
   string monikerName = "Elevation:Administrator!new:" + CLSID;

   NativeMethods.BIND_OPTS3 bo = new NativeMethods.BIND_OPTS3();
   bo.cbStruct = (uint)Marshal.SizeOf(bo);
   bo.hwnd = IntPtr.Zero;
   bo.dwClassContext = (int)NativeMethods.CLSCTX.CLSCTX_ALL;

   object retVal = UnsafeNativeMethods.CoGetObject(monikerName, ref bo,
InterfaceID);

   return (retVal);
}
```

Share
Improve this answer
Follow

edited Mar 30, 2009 at 20:03
Richard Szalay
84.7k ● 19 ● 181 ● 240

answered Nov 22, 2008 at 21:52
Ryan
24.4k ● 24 ● 89 ● 133

3

I think the only way CoCreateInstanceAsAdmin works is if you have registered the COM component ahead of time. That may be a problem if you intend your application to work in an XCopy deployment setting.

For my own purposes in Gallio I decided to create a little hosting process on the side with a manifest to require admin privileges. Then when I need to perform an elevated action, I spin up an instance of the hosting process and instruct it via .Net remoting to execute a particular command registered in Gallio's Inversion of Control container.

This is a fair bit of work but Gallio already had an out of process hosting facility so adding elevation into the mix was not too hard. Moreover, this mechanism ensures that Gallio can perform privilege elevation without requiring prior installation of any other COM components in the registry.

Share Improve this answer Follow

answered May 26, 2009 at 21:00

Jeff Brown
1,178 ● 5 ● 8

Thank you for your feed-back; running a separate hosting process is not what I intended. I'd really love to be able to interact with the elevated COM component as if it were a local object (i.e. the elevated COM component should be able to call back into the other local .NET objects). – Pierre Arnaud May 27, 2009 at 7:50

2

The elements of elevation are processes. So, if I understand your question correctly, and you want a way to elevate a COM object in your process, than the answer is you can't. The entire point of CoCreateInstanceAsAdmin is to NOT run it in your process.

Share Improve this answer Follow

answered Sep 24, 2008 at 15:00

MSalters
179k ● 11 ● 164 ● 368