

RE: installer woes (MSI generated by VS.NET 2003)

Source:

<http://www.tech-archive.net/Archive/VisualStudio/microsoft.public.vsnet.general/2005-02/0375.html>

From: Patrick Baker [MSFT] (PatBak_at_online.microsoft.com)

Date: 02/15/05

Date: Tue, 15 Feb 2005 00:32:41 GMT

There could be any number of issues here. I'll just through out some ideas/info

- Upgrading can be a tricky thing. There are a number of ways to do so outlined within the SDK each with its own drawbacks. If I remember correctly, Setup Project should do a full uninstall and then a full re-install

- There is currently a known issue with upgrades and custom actions where an upgrade executes previous custom actions. Work-Arounds Include:

- 1) Use `InstallerClass=False`; set `Arguments` property to `"/Install"`, `"/Uninstall"`, etc.; and then implement command-line handlers for each action that delegates to the corresponding routine.

- 2) Use a unique filename for each version (i.e. rename `TestApp.exe` to `TestApp2.exe` in `Setup1.2.msi`).

- 3) Implement custom actions in C, VBScript, or JScript, as native code custom actions don't have this problem.

- You should not have to use a custom action to register (REGASM) an Assembly. Simply setting its `Register Property` to `vsdrCOM` should work.

Patrick Baker – Visual Basic – Deployment & Designer Quality Assurance Team

This posting is provided "AS IS" with no warranties, and confers no rights.

>Reply-To: "Arun Bhalla" <bhalla@arun.groogroo.com>

>From: "Arun Bhalla" <bhalla@arun.groogroo.com>

>Followup-To: microsoft.public.windows.msi

>Subject: installer woes (MSI generated by VS.NET 2003)

>Date: Tue, 1 Feb 2005 00:46:10 -0600

>Lines: 93

>X-Priority: 3

>X-MSMail-Priority: Normal

>X-Newsreader: Microsoft Outlook Express 6.00.2800.1437

>X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1441

>Message-ID: <Otd73mCCFHA.2016@TK2MSFTNGP15.phx.gbl>

>Newsgroups:

microsoft.public.dotnet.framework.sdk,microsoft.public.dotnet.general,microsoft.public.platformsdk.msi,microsoft.public.vsnet.general,microsoft.public.w

indows.msi

>NNTP-Posting-Host: visit3.ncsa.uiuc.edu 141.142.105.33

>Path:

cpmsftngxa10.phx.gbl!TK2MSFTFEED01.phx.gbl!TK2MSFTNGP08.phx.gbl!TK2MSFTNGP15.phx.gbl

>Xref: cpmsftngxa10.phx.gbl microsoft.public.dotnet.general:159655

microsoft.public.platformsdk.msi:24345 microsoft.public.vsnet.general:24046

microsoft.public.windows.msi:3966

microsoft.public.dotnet.framework.sdk:10259

>X-Tomcat-NG: microsoft.public.vsnet.general

>

>I'm having some inconsistency problems with my deployment project ("Setup")

>and its custom actions ("Installer"). I'm using Visual Studio .NET 2003

>(.NET 1.1, no service pack) on Windows XPSP1.

>

>My main project is a band object (Explorer Bar) in an assembly (.DLL) that

>needs to be registered on installation and unregistered on uninstallation.

In

>addition, during installation, Setup creates registry subkeys, sets registry

>values, installs some application files, and adds a new file-type. On

>uninstallation, Setup is supposed to undo all of that. More specifically,

I

>have custom actions for Setup that are run within the Installer class. The

>Installer class is within the main assembly, and perhaps surprisingly, that

>seems to work, some of the time. I have also tried putting the Installer

>class in its own assembly, both as a class library and as a console

>application, and that doesn't seem to work any better than having the

>Installer class in the main assembly. Sometimes the registration or

>unregistration of the main assembly fails. The workaround we tell our

users

>(a small userbase so far, fortunately) is to "repair" after installation,

or

>to uninstall completely before installing a new version. Neither are as

>ideal as being able to simply install over the previous version. I think

>there are two subproblems.

>

>1) The Uninstall custom action in the Installer class isn't always called.

>This is especially inconsistent. My Installer class overrides

>OnBeforeUninstall() [so that the main assembly is unregistered before the

>files are removed] and Uninstall() [so that savedState is empty, in order

>for the installState to be removed]. Lacking better debugging tools for

>MSIs, I added code in these methods to set registry entries containing the

>timestamp at the beginning and end of each of these methods. I verified

>that usually the Uninstall custom action is (but not always) called when

the

>user explicitly asks to remove the package (via "Add/Remove Programs"), but

>when the user is installing over a previous version, sometimes (but not

>always) the Uninstall custom action is called. As a side note, I change

the

>installer version, PackageCode, and ProductCode whenever I build a new

>installer. I don't understand why these inconsistencies occur. For a while
>I thought VS.NET might randomly corrupt the Setup.vdproj file and that I'd
>have to remove and add the custom actions again, in order to be sure that
>the Uninstall custom action was registered, and once or twice I built the
>Setup.vdproj file from scratch, fearing general corruption in the file.
This
>seemed to help (especially the latter), but not for too long, and it's not
a
>sustainable solution, even if it's not a placebo. In addition, which may
or
>may not be related, the Setup.MSI doesn't seem to always uninstall all
>registry entries, and because of that, I've had to add a few lines to my
>Uninstall custom action. It doesn't seem right to me that the MSI would
>sometimes not uninstall registry entries that it added in the first place.
>
>2) The second subproblem is more specific to my project, and this occurs
>sometimes when I'm registering the main assembly. The main assembly is
>registered in the overridden OnCommitted() method of the Installer class,
>presumably after the previous version's main assembly was unregistered in
>the OnBeforeUninstall() method. In the case where the previous version and
>the new version were installed in the same "Program Files" folder, the
>assembly containing the Installer class will have the same exact CodeBase,
>but they'll of course have a different FullName because the assembly
version
>has been automatically incremented. It seems that sometimes the Commit
>custom action will re-register the old assembly instead of registering the
>new assembly. My registration code looks like this:
> RegistrationServices rs = new RegistrationServices();
> rs.RegisterAssembly(MyAssembly, AssemblyRegistrationFlags.SetCodeBase);
>and my unregistration code looks like this:
> RegistrationServices rs = new RegistrationServices();
> rs.UnregisterAssembly(MyAssembly);
>MyAssembly is defined as a property:
> Assembly MyAssembly
> {
> get
> {
> return Assembly.LoadFile(InstallDirectory + "assembly.dll");
> }
> }
>I used to use Assembly.Load(path) instead, but it seemed that
>Assembly.LoadFile(path) worked better. My concern is that either the
>Uninstall and Commit custom actions share an AppDomain, so sometimes
>MyAssembly might return the previous assembly during the Commit phase if
the
>previous assembly wasn't somehow still in Commit's AppDomain after
Uninstall
>finished. I tried creating a temporary AppDomain and then loading the
>assemblies within that, but it threw a deserialization exception. I
suppose

microsoft.public.vsnet.general: RE: installer woes (MSI generated by VS.NET 2003)

>the overall issue of this subproblem is that I'm trying to register and
>unregister assemblies without knowing the FullName of either assembly, as
>I'm trying to semi-automate the build process and stay within VS.NET as
much
>as possible (for now).
>
>I know these are complex and possibly peculiar problems! While I'd love
for
>someone to post a solution for either (or both!) subproblems, at the very
>least, I'd appreciate hearing tips regarding debugging deployment projects
>and custom installation classes, similar stories of installer woes and
>solutions, and recommendations for possibly better installation/deployment
>tools than VS.NET's deployment projects. Something like InstallShield
seems
>like overkill, but maybe it's what I need to have dependable installations.
>
>Thanks for reading this far!
>Arun
>
>
>