



 **Freitag, 12. März 2010**

## What is useLegacyV2RuntimeActivationPolicy for?

This post is intended to fill a gap in the current MSDN documentation for this attribute ([http://msdn.microsoft.com/en-us/library/bbx34a2h\(VS.100\).aspx](http://msdn.microsoft.com/en-us/library/bbx34a2h(VS.100).aspx)). This gap should be filled by the time .NET 4 ships.

There is a lot of confusion about what the useLegacyV2RuntimeActivationPolicy attribute does. Most often, it is used to allow a pre-v4 mixed-mode assembly to load in v4. In that context, the name makes very little sense. Below is an explanation I've provided to people internally that explains the attribute in the context for which it was named. This should give people a better idea of what it does, as well as understand some of the subtleties of in-proc SxS.

Ultimately, this attribute has to do with the behavior of the "legacy shim APIs". You can think of these as encompassing several categories of CLR activation:

- CorBindToRuntimeEx and friends - This includes most of the flat exports of mscoree.dll defined in mscoree.h (GetCORSystemDirectory, GetCORVersion, LoadLibraryShim, etc). Note, this also includes the strong name APIs defined in strongname.h)
- Pre-v4 COM activation – This includes CoCreateInstance of a CLSID (or type identifier) whose latest registration is against a pre-v4 runtime version. Note this includes both the "new" operator on such a co-class from managed code, or the result of Activator.CreateInstance against a type created by Type.GetTypeFromCLSID on such a CLSID.
- Pre-v4 IJW (mixed mode) activation – For example, calling into a native export on such an assembly
- Native activation of a native runtime-provided COM CLSID – Such as CoCreateInstance on ICLRRuntimeHost's CLSID
- Native activation of a managed framework CLSID – Such as CoCreateInstance on System.Arraylist's CLSID (extremely rare)

All these have a "single runtime per process" view of the world, so we try to make those codepaths believe they still exist in that world by "unifying" the version that they see. After a given version has been chosen by one of these codepaths, that's the version that all of them see for the remainder of the process lifetime. Additionally, all of these activation paths had some kind of roll-forward semantics associated with them. We "cap" those semantics at v2, meaning by default none of these codepaths see v4 at all. This allows us to claim that installing v4 is "non-impactful". It should not change the behavior of existing components when installed. (Note that this has the interesting side-effect of a v4 only machine appearing to have no runtimes installed at all via these codepaths.)

This is all well and good until someone WANTS those codepaths to see v4. Rolling a v2 managed app forward to v4 using a config (without the attribute) works just fine, unless that app also expects interaction with these "legacy" codepaths to be associated with the current runtime (v4). For instance, a p/Invoke to GetCorSystemDirectory in order to construct a path to Fusion.dll (please don't do that, BTW) will give you v2's fusion.dll. COM activation of a managed COM object

### Navigation

[Home](#)  
[dasBlog](#)  
[CodePlex](#)  
[newtelligence AG](#)



### On this page

[What is useLegacyV2RuntimeActivationPolicy for?](#)

### Archive

< September 2012 >

Mo	Di	Mi	Do	Fr	Sa	So
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7



[April, 2010 \(1\)](#)

[März, 2010 \(1\)](#)

[Mai, 2009 \(1\)](#)

[Dezember, 2008 \(2\)](#)

[November, 2008 \(1\)](#)

[Oktober, 2008 \(1\)](#)

[September, 2008 \(1\)](#)

[Juli, 2008 \(1\)](#)

[Juni, 2008 \(2\)](#)

[Mai, 2008 \(1\)](#)

[April, 2008 \(2\)](#)

[Februar, 2008 \(3\)](#)

[Januar, 2008 \(2\)](#)

will prefer the runtime it was built against rather than load into the current runtime (meaning you may be dealing with interop rather than a concrete CLR type). That may work, and it may not, depending on what you're doing.

The `useLegacyV2RuntimeActivationPolicy` attribute basically lets you say, "I have some dependencies on the legacy shim APIs. Please make them work the way they used to with respect to the chosen runtime." In that context, hopefully the name makes more sense to you. It is **\*mostly\*** equivalent to calling `CorBindToRuntimeEx` using the full version string for v4. We also have a **method in our new shim APIs** to do this programmatically, the difference being that in a config file, it can be done declaratively, which is useful for a host that uses config files to determine which runtime to load plugins into. (the attributes value (or lack of value) is conveyed back to a host via the `pdwConfigFlags` parameter of **ICLRMetaHostPolicy::GetRequestedRuntime**)

One of the big reasons people need to do this is if they have a dependency on a pre-v4 `IJW` assembly. By default, we can't allow those to load into v4\*. Putting this attribute in your config allows this to happen.

Why don't we make this the default behavior? You might argue that this behavior is more compatible, and makes porting code from previous versions much easier. If you'll recall, this can't be the default behavior because it would make installation of v4 impactful, which can break existing apps installed on your machine.

Well, why don't we make this the default behavior for v4 managed apps? Well, that is precisely the behavior we had for beta 1. As we started trying to explain the behavior to people, we found it was very difficult to explain how these legacy codepaths worked. We ultimately decided that making the behavior consistent was better. The example that ultimately convinced me we had made the right choice was that the behavior of a library would change based on whether it was hosted by a native process or a managed one. That seemed really bad to me.

You might say, "Why shouldn't I just set this for every app I have?" Well, the downside of this attribute is that it turns off in-proc SxS with pre-v4 runtimes. It locks them out of the process. This may not matter to your scenario. If you look at some of the runtime tools, they are using this attribute. Even Visual Studio uses this attribute. Don't just blindly use it though. If you find yourself needing it in something other than a migration aid, or for loading pre-v4 mixed-mode assemblies (which we hope becomes more rare moving forward as people start updating the interesting mixed-mode binaries out there), I'd like to know about it. Leave me a comment!

Hopefully, you've got a better handle on exactly what this attribute means, and can make a more informed decision about when it is appropriate to use.

\*There are many engineering challenges around in-proc SxS and `IJW` assemblies. Currently, pre-v4 `IJW` assemblies can only load into the runtime that is associated with the "legacy shim APIs". But any given `IJW` assembly (regardless of version) may only be loaded into a single runtime per process at this time.

## CLR | Software Development | Technical

posted on Freitag, 12. März 2010 19:14:59 (Pacific Standard Time, UTC-08:00) # [Kommentare \[2\]](#)

Verwandte Themen:

[Writing a CLR host that uses v4 MetaHost APIs, but can run without v4 installed](#)

[Random fun book thing and CLR In-Proc SxS](#)

[LinqToStd now on CodePlex](#)

[Image Slicer for Deep Zoom in Silverlight 2](#)

[My team is hiring. Come work on the CLR team!](#)

























[Silverlight limitations and Constrained Callvirt in IL](#)

Januar, 2008 (2)  
 Dezember, 2007 (3)  
 November, 2007 (1)  
 Oktober, 2007 (2)  
 September, 2007 (2)  
 August, 2007 (4)  
 Juli, 2007 (5)  
 Juni, 2007 (3)  
 Mai, 2007 (5)  
 April, 2007 (4)  
 März, 2007 (5)  
 Februar, 2007 (6)  
 Januar, 2007 (7)  
 Dezember, 2006 (6)  
 November, 2006 (8)  
 Oktober, 2006 (10)  
 September, 2006 (13)  
 August, 2006 (16)  
 Juli, 2006 (3)  
 Juni, 2006 (4)  
 Mai, 2006 (8)  
 April, 2006 (4)  
 März, 2006 (1)  
 Februar, 2006 (3)  
 Januar, 2006 (10)  
 Dezember, 2005 (10)  
 November, 2005 (11)  
 Oktober, 2005 (8)  
 September, 2005 (3)  
 August, 2005 (11)  
 Juli, 2005 (4)  
 Juni, 2005 (9)  
 Mai, 2005 (6)  
 April, 2005 (15)  
 März, 2005 (9)  
 Februar, 2005 (18)  
 Januar, 2005 (18)  
 Dezember, 2004 (16)  
 November, 2004 (14)  
 Oktober, 2004 (10)  
 September, 2004 (6)  
 August, 2004 (14)  
 Juli, 2004 (17)  
 Juni, 2004 (11)  
 Mai, 2004 (22)  
 April, 2004 (17)  
 März, 2004 (17)

**Februar, 2004 (5)**  
**Januar, 2004 (7)**  
**Dezember, 2003 (7)**  
**November, 2003 (17)**  
**Oktober, 2003 (9)**  
**September, 2003 (10)**  
**August, 2003 (2)**  
**Juli, 2003 (7)**  
**Juni, 2003 (24)**

[Month View](#)

## Kategorien

-  [Announcements](#)
-  [Becky](#)
-  [Cell Phones](#)
-  [Church](#)
-  [CLR](#)
-  [Crazy stuff](#)
-  [Delegates](#)
-  [Do it yourself](#)
-  [FilmProjects](#)
  
-  [Fun](#)
-  [Gripes](#)
-  [Holidays](#)
-  [Jenna](#)
-  [Movies](#)
-  [New Stuff](#)
-  [News](#)
-  [Photography](#)
-  [Soccer](#)
-  [Software Development](#)
-  [Technical](#)
-  [Vacation](#)
-  [Video Games](#)
-  [Wish Lists](#)
-  [work](#)

## Weblog Liste

### **Disclaimer**

The opinions expressed herein are my own personal opinions and do not represent my employer's view in any way.

© Copyright 2012 Mark Miller

Design von **James Snape**  
with newtelligence dasBlog  
2.3.9074.18820

Wählen Sie ein Design:

marknew

**Anmelden**

