# Kerbal Space Program - Bug #9752

## Assembly dependency mechanism works improperly

05/23/2016 06:29 AM - IgorZ

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 05/23/2016 |
| **Severity:** | High | **% Done:** | 100% |
| **Assignee:** | | | |
| **Category:** | Application | | |
| **Target version:** | 1.2.0 | | |
| **Version:** | 1.1.2 | **Language:** | English (US) |
| **Platform:** | Windows | **Mod Related:** | No |
| **Expansion:** | | | |

**Description**

It looks KSP starting from 1.1 treats any assembly dependency in a "version specific" way regardless to the actual project settings. As a result updating even a minor digit in the version of the assembly (e.g. when a new version is released) makes crashing of all the depending mods.

Here is example. Pretend there are two assemblies: A v1.0 (A.dll), and B v2.0 (B.dll) that depends on A. Let's also assume that B was build with A v1.0, and "specific version" setting was **disabled** for that reference. I.e. in C# world this assembly will work with any A version which is higher than 1.0. See AssemblyInfo files below.

A:
[assembly: AssemblyVersion("1.0")]

B:
[assembly: AssemblyVersion("2.0")]
[assembly: KSPAssembly("A", 1, 0)]

If these two particular versions are deployed in GameData then everything is working fine. It doesn't matter if A and B live in the same folder or in the different folders: dependency mechanism will correctly provide A v1.0 to B on load.

Now, let's assume there was a new version of A released.

A:
[assembly: AssemblyVersion("1.1")]

Remember, that B was build with v1.0. In normal C# world it will continue working just fine but not in KSP. In KSP B will fail badly if A.dll lives not in the same folder. If they live in the same folder then version difference will be fine (I guess, it's because of C# framework deals with versions in this case). In the logs KPS doesn't complain about not satisfied dependency which is expected: A v1.1 does satisfy dependency requirement of B (v1.0 or higher). Though, any attempt to use a type form assembly A will cause an exception like this:

160522T231027.596 [EXCEPTION] FileNotFoundException: Could not load file or assembly 'KIS, Version=1.2.9.0, Culture=neutral, PublicKeyToken=null' or one of its dependencies.

It has a huge impact on the modders. I'm a maintainer of Kerbal Inventory System (KIS), and every new release breaks a pack of other mods. It would be bearable if KSP was just skipping broken mods due to failed dependency but it doesn't happen. Instead, loading of the game goes fine and then somewhere in the gameplay the exception storm starts. The consequences can be really different but it always looks like a weird bug in the mod because of some functionality works while the other is throwing exceptions.

I tried different version management approaches people usually do with normal C# apps but none of them worked. The only working solution so far is using reflections to access KIS but in many cases it's not feasible.

### History

#### #1 - 05/24/2016 07:08 AM - IgorZ

Correction to the assembly files. I meant the following:

A:
[assembly: AssemblyVersion("1.0")]
[assembly: KSPAssembly("A", 1, 0)]

B:
[assembly: AssemblyVersion("2.0")]
[assembly: KSPAssembly("B", 2, 0)]
[assembly: KSPAssemblyDependency("A", 1, 0)]

On A version update:
[assembly: AssemblyVersion("1.1")]
[assembly: KSPAssembly("A", 1, 1)]

I.e. KSP dependencies and assembly versions are specified via the right attributes but it doesn't help. Sorry for the confusion.

#### #3 - 06/20/2016 04:45 AM - NathanKell

The only assembly version checking we appear to be doing is KSPAssembly-based; it looks like Assembly.LoadFrom() is doing the hard ref checking and complaining when the specific version is not found. After some MSDN digging it appears that has nothing whatsoever to do with "strict versioning" which is a build-time not run-time flag. Instead it appears to depend on whether an assembly is strongly named.

#### #4 - 06/20/2016 04:48 AM - NathanKell

See also e.g. http://stackoverflow.com/questions/20537961/how-not-to-reference-a-specific-version-of-an-assembly

#### #5 - 06/20/2016 06:03 PM - IgorZ

The forum thread you've suggested recommends using policies and app config file. This is a recommended way to deal with third-party C# assemblies. Alas, not policy nor app config is working when supplied to KSP with the plugin DLL. I may assume that this mechanism works "out of the box" only for a pure C# applications. In case of KSP it's not a C# application. Instead, Unity (or KSP core?) loads the assemblies, and I would guess some extra work needs to be done at that end to properly handle policies. Using app config doesn't seem to be the right approach since "app" in this case is the game, not the plugin. Though, it doesn't work anyways.

And note that the issue doesn't show up when both DLLs live in the same folder.

#### #6 - 07/06/2016 02:13 AM - NathanKell

Ah, sorry, missed responding.

Since the only thing we do to load the assembly (in the end) is...call Assembly.LoadFrom() with the path to the assembly to load, I'm not sure how to proceed. It's not like we're doing any dependency checking (beyond the KSPAssembly stuff, which *does* work properly as logging can attest). So it's clearly the mono runtime itself failing to see dependencies met. As to why it works when they are in the same folder...no idea.

#### #7 - 08/12/2016 04:20 AM - IgorZ

*- File AddonVersionDependancyBinder.cs added*

So, I continued palying with this issue. And one thing I figured out is that LoadFrom is not the optimal way to load assemblies. I think this is the issue:

```
If an assembly is loaded with LoadFrom, and later an assembly in the load context attempts to load the same as
sembly by display name, the load attempt fails. This can occur when an assembly is de-serialized.
```

This is what happens:

1. A v1.1 is loaded via LoadFrom.
2. Loading of B tries to get A v1.0 which doesn't exist, so it attempts to loading it by a display name "A" which fails due to the mentioned limitation of LoadFrom approach.

That said, the normal version resolution workflow gets broken in the game.

Re-factoring all the KSP versioning code to the Load approach would be the best thing to do but I may guess it's not feasible :) So I managed to create a rather simple workaround. See the code in the attachment (it's released under public domain, feel free to change it as you wish).

Please, let me know if you're willing to incorporate this fix into the next KSP version. If not then I'll release the fix as an addon which is not the best approach but works.

#### #8 - 08/12/2016 10:25 PM - NathanKell

Thanks! Talking with Mike about it. At the least we'll add the binder (again, thanks! <3 ) and if we have time we'll do the deeper fix.

Cheers!

#### #9 - 08/14/2016 04:23 AM - IgorZ

Awesome! Looking forward to have it fixed in the next version :) Please, feel free to ping me if you need any extra info.

#### #10 - 10/13/2016 12:58 PM - Squelch

*- Status changed from New to Ready to Test*

*- Target version set to 1.2.0*

*- % Done changed from 0 to 80*

The assembly dependencies checking should be working correctly within the same major version now for KSP 1.2

**#11 - 11/06/2016 07:50 PM - JPLRepo**

*- Status changed from Ready to Test to Closed*

*- % Done changed from 80 to 100*

Closing. Issue was resolved in KSP 1.2.0 and has been tested.

### Files

| | | | |
|---|---|---|---|
| AddonVersionDependancyBinder.cs | 2.41 KB | 08/12/2016 | IgorZ |