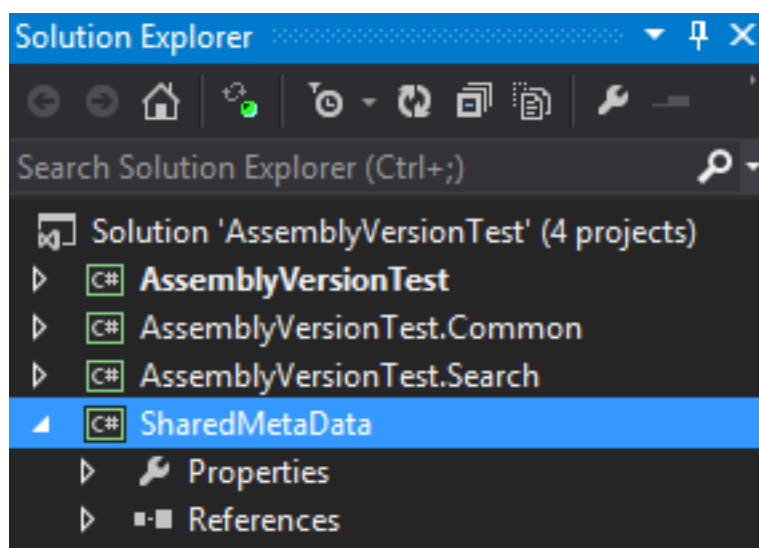


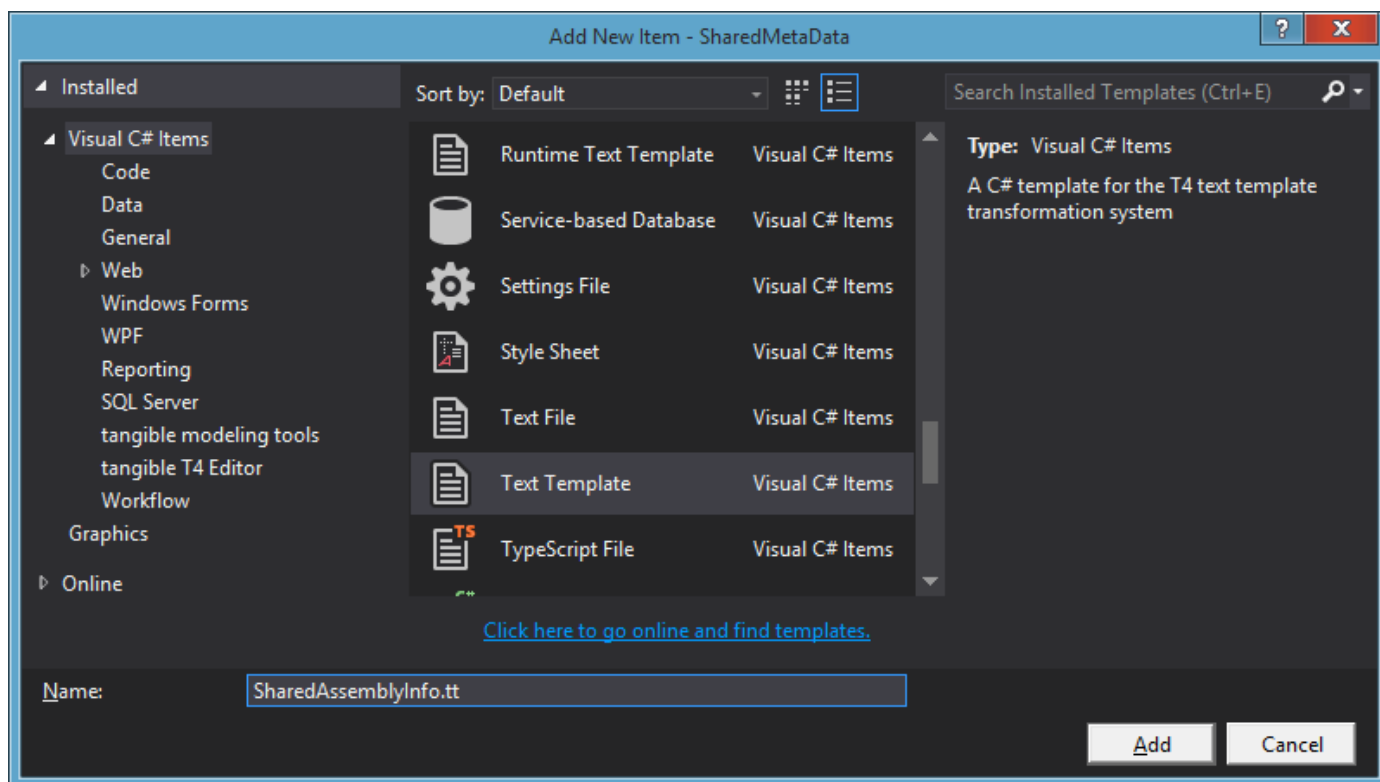
عموماً برای نگهداری ساده‌تر قسمت‌های مختلف یک پروژه، اجزای آن به اسمبلی‌های مختلفی تقسیم می‌شوند که هر کدام در یک پروژه‌ی مجزای ویژوال استودیو قرار خواهند گرفت. یکی از نیازهای مهم این نوع پروژه‌ها، داشتن شماره نگارش یکسانی بین اسمبلی‌های آن است. به این ترتیب توزیع نهایی ساده‌تر شده و همچنین پشتیبانی از آن‌ها در دراز مدت، بر اساس این شماره نگارش بهتر صورت خواهد گرفت. برای مثال در لاگ‌های خطای برنامه با بررسی شماره نگارش اسمبلی مرتبط، حداقل می‌توان متوجه شد که آیا کاربر از آخرین نسخه‌ی برنامه استفاده می‌کند یا خیر. روش معمول انجام این کار، به روز رسانی دستی تمام فایل‌های AssemblyInfo.cs یک Solution است و همچنین اطمینان حاصل کردن از همگام بودن آن‌ها. در ادامه قصد داریم با استفاده از فایل‌های T4، یک فایل SharedAssemblyInfo.tt را جهت تولید اطلاعات مشترک Build بین اسمبلی‌های مختلف یک پروژه، تولید کنیم.

ایجاد پروژه‌ی SharedMetadata

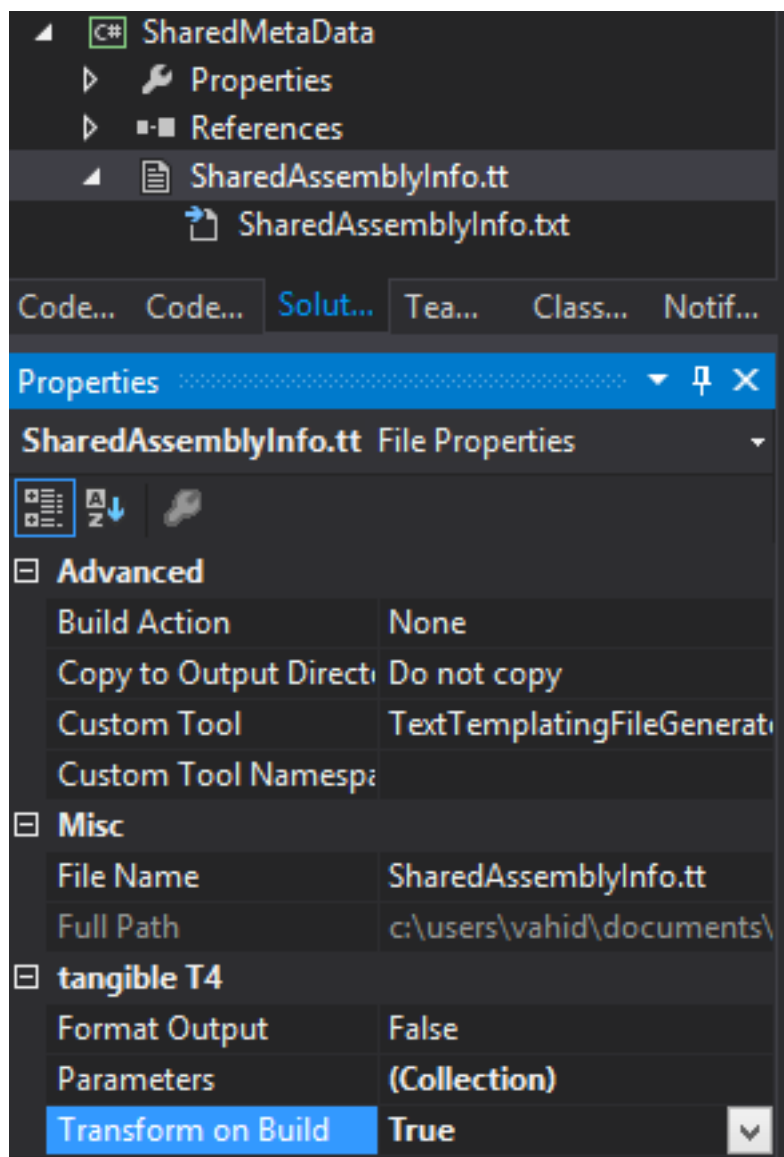
برای نگهداری فایل مشترک SharedAssemblyInfo.cs نهایی و همچنین اطمینان از تولید مجدد آن به ازای هر Build، یک پروژه‌ی class library جدید را به نام SharedMetadata به Solution جاری اضافه کنید.



سپس نیاز است یک فایل text template جدید را به نام SharedAssemblyInfo.tt، به این پروژه اضافه کنید.



به خواص فایل SharedAssemblyInfo.tt مراجعه کرده و [Transform on build](#) آن را [true](#) کنید. به این ترتیب مطمئن خواهیم شد این فایل به ازای هر build جدید، مجدداً تولید می‌گردد.



اکنون محتوای این فایل را به نحو ذیل تغییر دهید:

```
<#@ template debug="false" hostspecific="false" language="C#" #>
//
// This code was generated by a tool. Any changes made manually will be lost
// the next time this code is regenerated.
//
using System.Reflection;
using System.Resources;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

[assembly: AssemblyCompany("some name")]
[assembly: AssemblyCulture("")]
[assembly: NeutralResourcesLanguageAttribute("en")]

[assembly: AssemblyProduct("product name")]
[assembly: AssemblyCopyright("Copyright VahidN 2014")]
[assembly: AssemblyTrademark("some name")]

#if DEBUG
[assembly: AssemblyConfiguration("Debug")]
#else
[assembly: AssemblyConfiguration("Release")]
#endif

// Assembly Versions are incremented manually when branching the code for a release.
```

```
[assembly: AssemblyVersion("<#> this.MajorVersion #>.<#> this.MinorVersion #>.<#> this.BuildNumber
#>.<#> this.RevisionNumber #>")]
// Assembly File Version should be incremented automatically as part of the build process.
[assembly: AssemblyFileVersion("<#> this.MajorVersion #>.<#> this.MinorVersion #>.<#> this.BuildNumber
#>.<#> this.RevisionNumber #>")]

<#+
// Manually incremented for major releases, such as adding many new features to the solution or
introducing breaking changes.
int MajorVersion = 1;
// Manually incremented for minor releases, such as introducing small changes to existing features or
adding new features.
int MinorVersion = 0;
// Typically incremented automatically as part of every build performed on the Build Server.
int BuildNumber = (int)(DateTime.UtcNow - new DateTime(2013,1,1)).TotalDays;
// Incremented for QFEs (a.k.a. "hotfixes" or patches) to builds released into the Production
environment.
// This is set to zero for the initial release of any major/minor version of the solution.
int RevisionNumber = 0;
#>
```

در این فایل اجزای شماره نگارش برنامه به صورت متغیر تعریف شده‌اند. هر بار که نیاز است یک نگارش جدید ارائه شود، می‌توان این اعداد را تغییر داد.

MajorVersion با افزودن تعداد زیادی قابلیت به برنامه، به صورت دستی تغییر می‌کند. همچنین اگر یک breaking change در برنامه یا کتابخانه وجود داشته باشد نیز این شماره باید تغییر نماید.

MinorVersion با افزودن ویژگی‌های کوچکی به نگارش فعلی برنامه تغییر می‌کند.

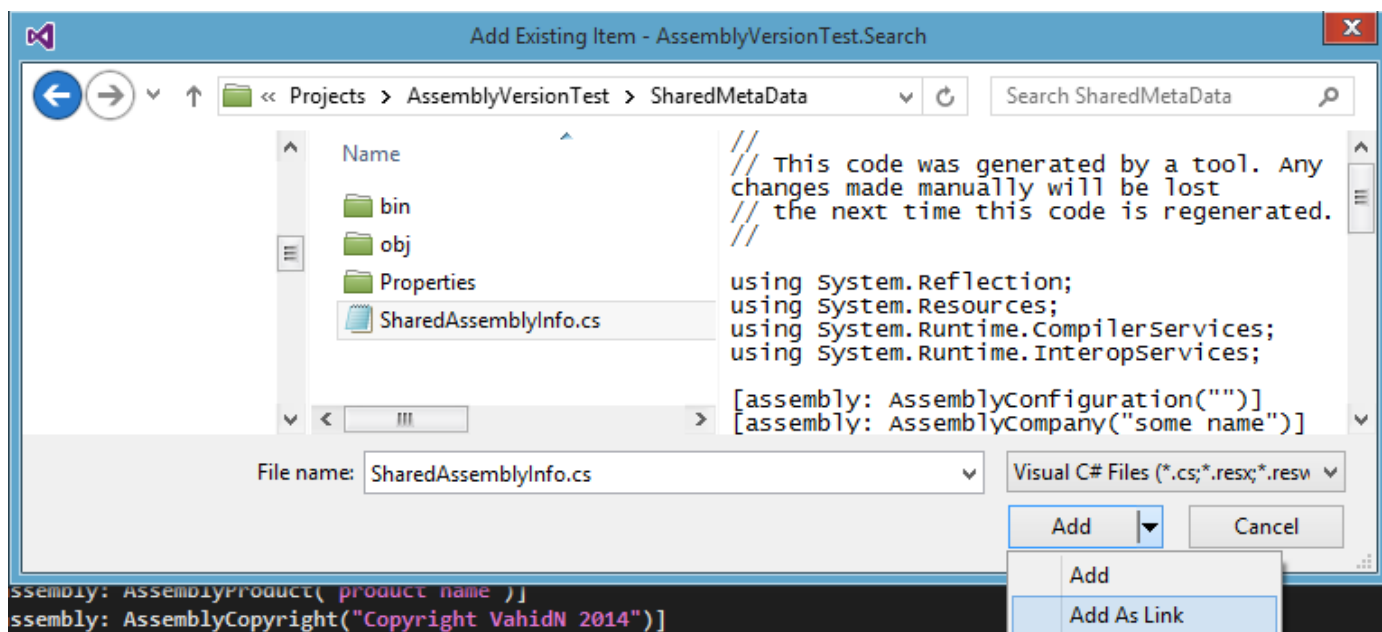
BuildNumber به صورت خودکار بر اساس هر Build انجام شده باید تغییر یابد. در اینجا این عدد به صورت خودکار به ازای هر روز، یک واحد افزایش پیدا می‌کند. ابتدای مبداء آن در این مثال، 2013 قرار گرفته‌است.

RevisionNumber با ارائه یک وصله جدید برای نگارش فعلی برنامه، به صورت دستی باید تغییر کند. اگر اعداد شماره نگارش major یا minor تغییر کنند، این عدد باید به صفر تنظیم شود.

اکنون اگر این محتوای جدید را ذخیره کنید، فایل SharedAssemblyInfo.cs به صورت خودکار تولید خواهد شد.

افزودن فایل SharedAssemblyInfo.cs به صورت لینک به تمام پروژه‌ها

نحوه‌ی افزودن فایل جدید SharedAssemblyInfo.cs به پروژه‌های موجود، اندکی متفاوت است با روش معمول افزودن فایل‌های cs هر پروژه. ابتدا از منوی پروژه گزینه‌ی add existing item را انتخاب کنید. سپس فایل SharedAssemblyInfo.cs را یافته و به صورت add as link، به تمام پروژه‌های موجود اضافه کنید.



اینکار باید در مورد تمام پروژه‌ها صورت گیرد. به این ترتیب چون فایل SharedAssemblyInfo.cs به این پروژه‌ها صرفاً لینک شده‌است، اگر محتوای آن در پروژه‌ی metadata تغییر کند، به صورت خودکار و یک دست، در تمام پروژه‌های دیگر نیز منعکس خواهد شد.

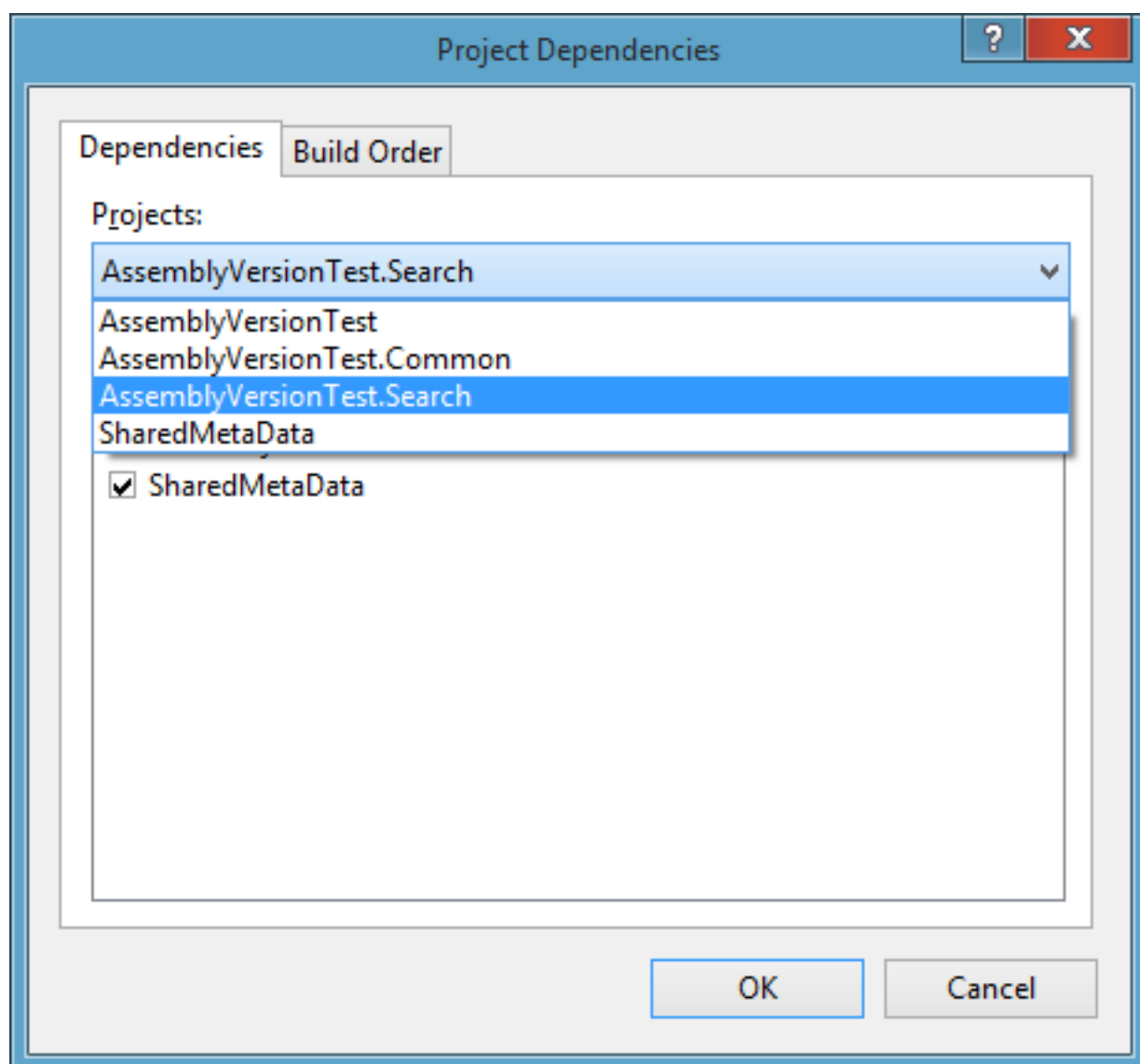
در ادامه اگر بخواهید Solution را Build کنید، پیام تکراری بودن یک سری از ویژگی‌ها را یافت خواهید کرد. این مورد از این جهت رخ می‌دهد که هنوز فایل‌های AssemblyInfo.cs اصلی، در پروژه‌های برنامه موجود هستند. این فایل‌ها را یافته و صرفاً چند سطر همیشه ثابت ذیل را در آن‌ها باقی بگذارید:

```
using System.Reflection;
using System.Runtime.InteropServices;









[assembly: AssemblyTitle("title")]
[assembly: AssemblyDescription("")]
[assembly: ComVisible(false)]
[assembly: Guid("9cde6054-dd73-42d5-a859-7d4b6dc9b596")]
```

اضافه کردن build dependency به پروژه Metadata

در پایان کار نیاز است اطمینان حاصل کنیم، فایل SharedAssemblyInfo.cs به صورت خودکار پیش از Build هر پروژه، تولید می‌شود. برای این منظور، از منوی Project، گزینه‌ی Project dependencies را انتخاب کنید. سپس در برگه‌ی dependencies آن، به ازای تمام پروژه‌های موجود، گزینه‌ی SharedMetadata را انتخاب نمایید.



این مساله سبب اجرای خودکار فایل SharedAssemblyInfo.tt پیش از هر Build می‌شود و به این ترتیب می‌توان از تازه بودن اطلاعات SharedAssemblyInfo.cs اطمینان حاصل کرد. اکنون اگر پروژه را Build کنید، تمام اجزای آن شماره نگارش یکسانی را خواهند داشت:

Name	File version	Product version
 AssemblyVersionTest.Common.dll	1.0.645.0	1.0.645.0
 AssemblyVersionTest.Common.pdb		
 AssemblyVersionTest.exe	1.0.645.0	1.0.645.0
 AssemblyVersionTest.pdb		
 AssemblyVersionTest.Search.dll	1.0.645.0	1.0.645.0
 AssemblyVersionTest.Search.pdb		
 AssemblyVersionTest.vshost.exe	12.0.30723.0	12.0.30723.0
 AssemblyVersionTest.vshost.exe.manifest		

و در دفعات آتی، تنها نیاز است تک فایل SharedAssemblyInfo.tt را برای تغییر شماره نگارش‌های اصلی، ویرایش کرد.

نظرات خوانندگان

نویسنده: لیبرتاد

تاریخ: ۱۳۹۳/۰۷/۱۹ ۱۱:۴۱

آموزش خوبی بود البته در اکثر اوقات بهتر است که شماره نگارش اسمبلی‌های پروژه‌ها یکی نباشد. ممکن است از چندین پروژه یک یا چندتای آنها در آپدیت‌های مختلف هیچ تغییری نداشته باشند

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۷/۱۹ ۱۳:۰۰

یک اسمبلی در پروژه، به خودی خود فاقد مفهوم است و در قالب نگارش کلی برنامه مفهوم پیدا می‌کند. فرض کنید برنامه شما از یک فایل exe به همراه دو اسمبلی A و B، تشکیل شده‌است. اسمبلی A، نگارش یک دارد. اسمبلی B نگارش 2 و کل برنامه در نگارش 2.5 است. خطایی به شما گزارش شده‌است که در آن استثنای حاصل، از نگارش یک اسمبلی A صادر شده‌است. این مشکل که در نتیجه‌ی در یافت پردازش اشتباهی از اسمبلی B بوده و در نگارش 2 آن برطرف شده، به صورت خودکار با ارتقاء به آخرین نگارش برنامه، برطرف می‌شود. سؤال: آیا اکنون می‌توانید تشخیص دهید کاربر از آخرین نگارش محصول شما استفاده می‌کند؟ نگارش یک A، آخرین نگارش آن است و اما برنامه در نگارش 2.5 قرار دارد. کاربر هم مدتی است که برنامه را به روز نکرده‌است. یک سیستم از همکاری اجزای مختلف آن مفهوم پیدا می‌کند. برای مطالعه بیشتر: «[Best Practices for .NET Assembly Versioning](#)». عبارت «ensuring all of the various assemblies in the solution share the same version» حداقل دوبار در آن تکرار شده‌است.