TFS Tools and Project Reference Analyzer

Jason W. Turpin

Contents

[Introduction 1](#_Toc435696583)

[TFS Binding Cleaner 2](#_Toc435696584)

[File Backup Comments 2](#_Toc435696585)

[Operations Performed (aside from file backups) 2](#_Toc435696586)

[Visual Studio Solution Reference Analyzer 2](#_Toc435696587)

[Reference Type Categories 2](#_Toc435696588)

[Specific Reference Analysis 2](#_Toc435696589)

[Internally Shared/Developed Library Reference Considerations 3](#_Toc435696590)

[TFS Changeset Grabber 3](#_Toc435696591)

[Configuration 4](#_Toc435696592)

[Source and Parent Branch Formats 4](#_Toc435696593)

[Navigating the Branch Hierarchy 5](#_Toc435696594)

# Introduction

This utility is actually a set of three utilities, each of which provides specific Visual Studio/Team Foundation Server project management functions.

* **TFs Binding Cleaner** – This utility removes any TFS/VSS (yes, Visual Source Safe…) control files as well as TFS references from a selected Solution and all related Projects. It is designed to ease the transition of a project from one TFS Server/Project to another by clearing out existing source code control binding information.
* **Visual Studio Solution Reference Analyzer** – This utility analyzes a solution and summarizes all Project references into three categories:
  + Project References – References between Projects in the Solution.
  + Framework References – References that are part of the .Net Framework, including Extention References such as Microsoft.VisualStudio.TestTools.\*.
  + Library References – External library references to 3rd party Controls/Libraries and potentially internally developed libraries where the library is referenced via a binary DLL rather than having the Project included in the Solution itself (see the Some Library Reference Considerations section below).
* TFS Changeset Grabber – This utility allows you to pull down Changeset files outside of Visual Studio. This can be useful if you have migrated to another TFS server but opted not to move an entire Project Collection (so the Changesets would only exist on the old TFS Server and need to be retrieved and incorporated into the new Servers’s Source Code repository).

# TFS Binding Cleaner

The TFS Binding Cleaner removes TFS and VSS related source control files as well as references to TFS in the Solution or related Project files. This makes it easy to move a project to a different TFS Project Collection or Server without Visual Studio complaining about source control related issues (keep in mind that audit/change tracking is only persisted on the original server).

### File Backup Comments

* A backup location much be specified and it makes a complete backup before processing.
* Any Solution or Project files that are updated are also backed up in-place.

### Operations Performed (aside from file backups)

* This process will delete files associated with TFS bindings including mssccprj.scc, [project].vssscc, vsserver.scc, [project].vbproj.vspscc, and [project].csproj.vspscc files.
* This process will also edit the solution and related project files, removing TFS server references.

# Visual Studio Solution Reference Analyzer

The Solution Reference Analyzer provides a convenient way to better understand the References contained in a Solution and related Projects. It not only enumerates all References by type but it also does some analysis to identify potential issues or things that can cause confusion.

## Reference Type Categories

There are three categories of References that they are broken down into:

* **Project References** – References between Projects in the Solution.
* **Framework References** – References that are part of the .Net Framework, including Extention References such as Microsoft.VisualStudio.TestTools.\*.
* **Library References** – External library references to 3rd party Controls/Libraries and potentially internally developed libraries where the library is referenced via a binary DLL rather than having the Project included in the Solution itself (see the Some Library Reference Considerations section below).

## Specific Reference Analysis

Aside from listing and categorizing a Solution’s Referenes, it also looks for a couple of specific, actionable situations that can cause problems and which can be difficult to identify.

The following situations are part of the analysis:

* Check for Assemblies with the same Assembly Name but differing Fully Qualified Names or physical paths.
  + These are Assemblies having the same name referenced by two or more Projects in the Solution where the actual references are two different versions or implementations of the Assembly, usually due to different versions of the Assembly.
  + These should be resolved by selecting the correct “reference” Assembly and updating Projects to all use this version.
* Check for Framework References that may actually be Library References
  + If the Assembly Name doesn’t match the Fully Qualified Name for an Assembly, then it is not part of the Core .Net Framework.
  + These assemblies are in the GAC on the system the analysis is run on, but there is no guarantee that other computers or servers will have these Assemblies in their respective GAC repositories.
  + These Assemblies should probably be included as Library references rather than relying on the local GAC version.

## Internally Shared/Developed Library Reference Considerations

If you have Assemblies that are shared internally across Projects or Teams (things like data access, logging, shared service interfaces, etc.) you should setup consider setting them up as dedicated Libraries.

This means that, rather than having the Projects themselves in your solution (easy to make change shared code), you keep them in a separate Solution and only include DLL references to compiled binary files (hard to change shared source code).

This allows for shared Library Assemblies to be properly versioned so that dependent Solutions use reference copies rather than the most recent source code (which has no guarantee of compatibility).

During initial development, Reference Assemblies should be included in other Solutions for ease of development, but once they are mature they should be maintained separately.

So, pay attention to the Project References and consider whether they should be changed to Library References if other groups need to use them.

# TFS Changeset Grabber

If you migrate to a new version or a different TFS server you may break from your source code history and any changesets that have accumulated. This isn’t a bad thing as long as you keep your old TFS projects and servers intact for historical reference (I would actually recommend it).

And that’s where the TFS Changeset Grabber comes in. It allows for easy review and downloading of all changesets (for all users) across specified branches in a given Project Collection.

It takes a bit to get configured, but after that it’s rather powerful.

## Configuration

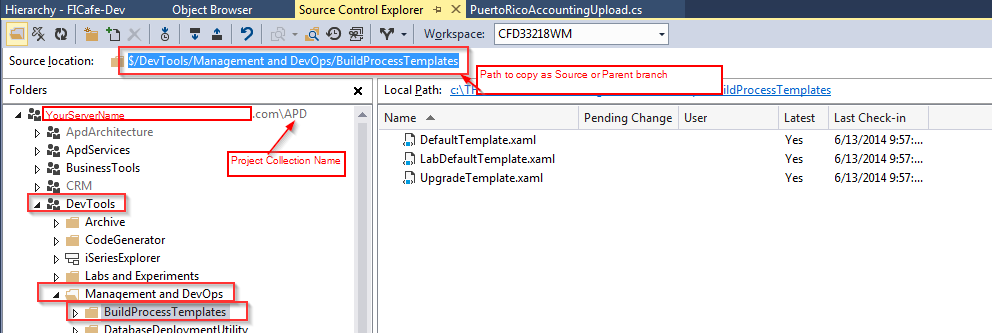
There are several required configuration values that have to be pretty exact for the Grabber to work (sorry for the lack of prompts/better UI).

1. Tfs Url – This URL needs to resolve to your project collection, the format is generally:
   1. <http://tfsServerNameOrDnsName>:8080/tfs/projectCollectionName
2. If you would like you can enter a domain\username and password, but this is optional. The account you are currently logged in as is used by default if these are left blank.
3. The Source Branch is where the changesets in question actually reside (the location of the changes).
4. The Parent Branch is where the “reference” files the changeset is based on reside.
   1. The application should be able to handle Parent branches more than one branch up the tree.
5. The Save to Folder at the bottom needs to be entered or selected.
   1. It must exist.
   2. **Folders will be created to represent the changesets structure with files copied to the appropriate sub-folders.**
6. There are some additional options at the bottom for replacing existing files and/or backing up existing files.

### Source and Parent Branch Formats

The Source and Parent branch values need to follow the “$” format that TFS uses.

Here’s an example, you can navigate to the Source or Parent branch in Source Control Explorer and then just select and copy the path from Visual Studio:



## Navigating the Branch Hierarchy

Here’s a link that describes how to determine the branch hierarchy for a project and a couple of steps summarizing how to do this.

<https://msdn.microsoft.com/en-us/library/dd465202.aspx>

Steps:

1. Select the branch you want to start from in Source Control Explorer.
2. Do File->Source Control->Branching and Merging->View Heirarchy