**Software**

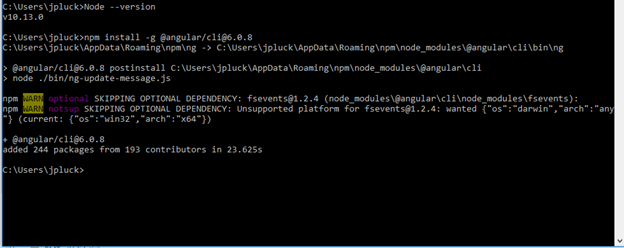
**Angular and Node.js Setup**



Simpson, Tabatha (RT-PHL)

Sep 2

1. On your local machine, install the latest stable version of Node.js from the following URL.  
   <https://nodejs.org/en/>
2. Open a new command prompt window and verify that Node is installed by typing node --version.
3. Install Angular. If the version is not specified the Node Package Manager will install the latest version.  
   To install Angular type the following command - npm install -g @angular/cli@6.0.8 (current version is 7.3.9, ask Lead if you should use this instead).
4. Your command prompt window should look similar to the screenshot below once the installation is complete.



1. Install [Visual Studio Code](https://code.visualstudio.com/Download) . This is needed to work on Angular applications.
2. Open Visual Studio Code.
3. Open a project and select *Terminal* >> *New Terminal*
4. Our local machine setup doesn't allow to run any Angular scripts. So, before we start the session we can use the command (Powershell mode, Admin):  
   Set-ExecutionPolicy unrestricted -Scope LocalMachine -Force
5. Navigate to your project directory and run the command npm install.
6. To start the project, open cmd as admin, navigate to your project folder, and run the command ng serve. Then open the localhost URL it specifies in a browser. When done, in the cmd window hit Ctrl + C to terminate the listener.
7. After the session is completed we should restore the setup with command:  
   Set-ExecutionPolicy restricted -Scope LocalMachine –Force

**Git Branching Strategy for Large Projects**

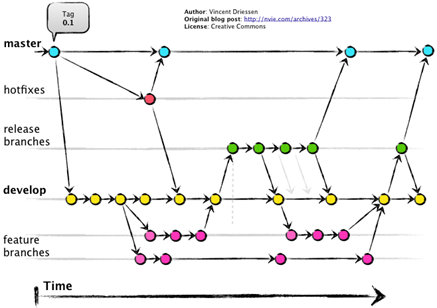


O'Boyle, Julianna (RT-PHL)

Sep 23, 2020

This strategy or pattern should be used when managing code across Government Teams/Developers for large projects in the Government space. It is not meant to be used for small projects or where one or two developers are assigned.

**Branch Flow**



**Branch Names**

**1. innovation**

* For experimentation/POCs or to do experimental work on a project that is already in production, thereby keeping the *develop* branch for support and maintenance
* Code may or may not get promoted to higher environments
* Can build pipeline
* Separate release pipeline into QAC

**2. feature(s)**

* For work going out in current or future release
* When code is finished, merge *feature* branch into *develop*
* If applicable, merge from *innovation*

**3. develop**

* For work that “should be” going into the next release
* Merge from *feature* branches into *develop*
* If applicable, merge from *innovation*
* QAT will occur in *develop*

**4. <SprintName>Release**

* Create from *master* branch at the beginning of the Sprint to establish baseline
* After QAT, each developer will merge their code from *develop* into the *release* branch
* “staging area” until ready for QAV
* no pipelines needed

**5. master**

* For QAV or production-ready code
* Merge from *Release* to *master*
* “what’s live”

**6. hotfix**

* Created from *master*
* When hotfix is confirmed, merge into *master* and *develop* branches

**Modifying Setup Project in Solution Migrated from TFS to Git**

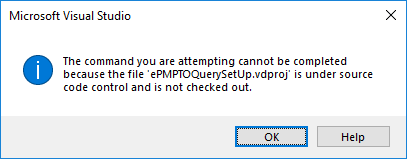


Imboden, Tom (RT-PHL)

Mar 17

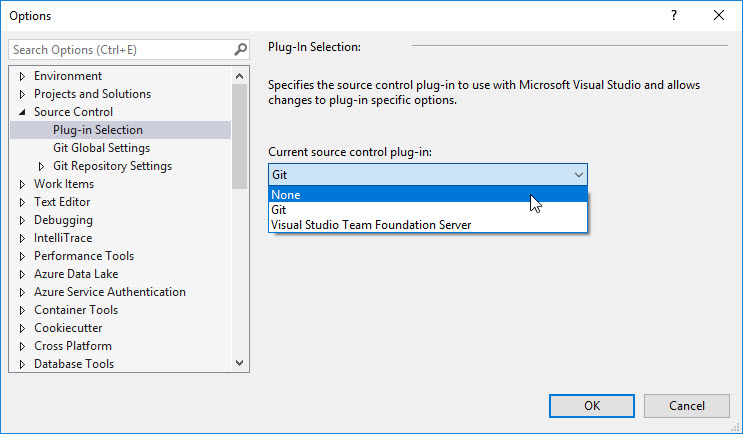
<https://stackoverflow.com/a/39733532>

With a solution migrated to Git, trying to make a change to the Setup project may result in the following message:

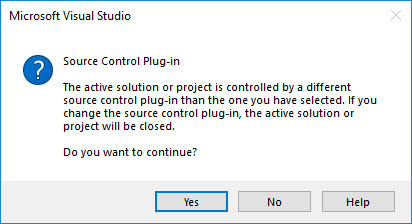


Try the following steps to make the Setup project modifiable under Git:

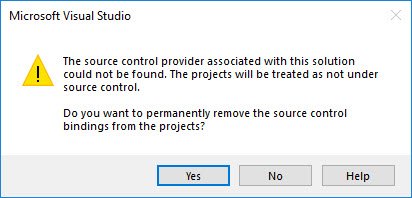
1. Open your solution file in notepad, and delete the following section: GlobalSection(TeamFoundationVersionControl)
2. Open the solution containing locked Setup Project in Visual Studio.
3. Go to Tools > Options > Source Control > Plug-in Selection, and pick 'None' from the dropdown.



1. Click "Yes" to the dialog that warns you the project will be closed.



1. Open the solution again.
2. Click Yes to permanently remove source control bindings from the project.



1. You can now go back to Tools > Options > Source Control > Plug-in Selection and pick Git again.
2. At this point the problem is fixed, and you will be able to modify your Setup Project without any issues.

**Analyzing Memory Dumps from Desktop Applications**

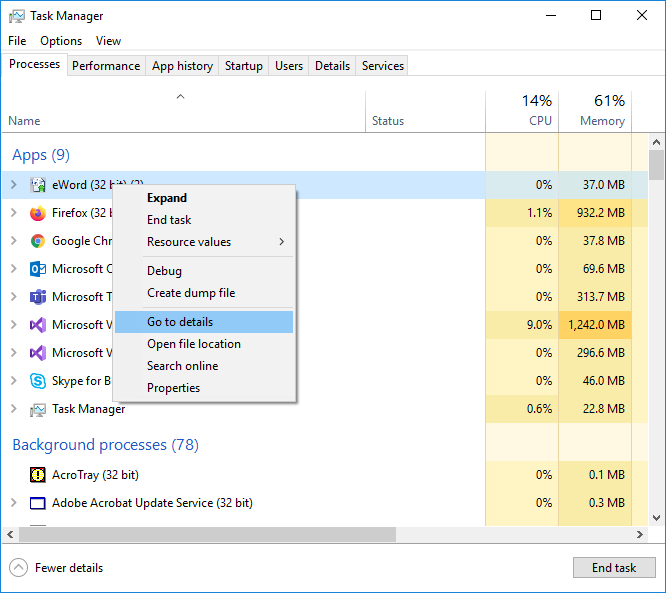


Moyer, Dan (RT-PHL)

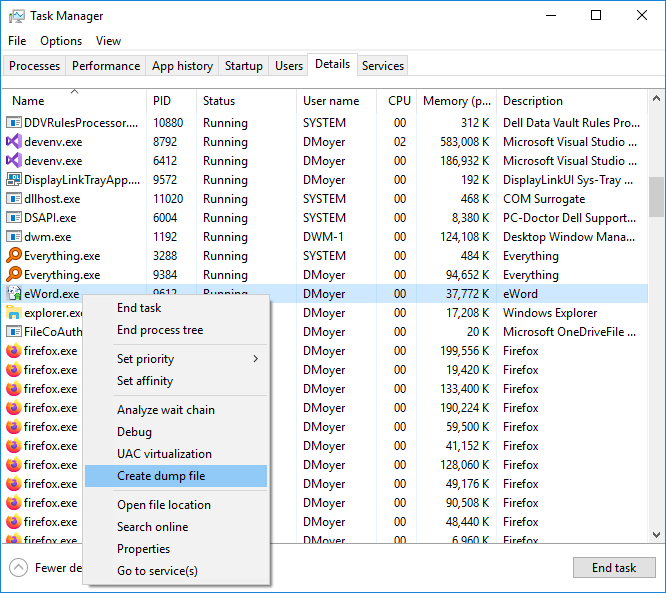
Jan 14, 2020

Purpose: If a program is frozen or hung, this process will help you determine where in the code the process is stuck. I will use eWord as an example, users were having it freeze and this process requires no installation on the user's machine to get what is needed.

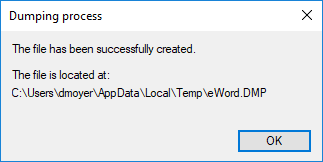
1. Acquire a memory dump by having the user open up Task Manager and then go to details...



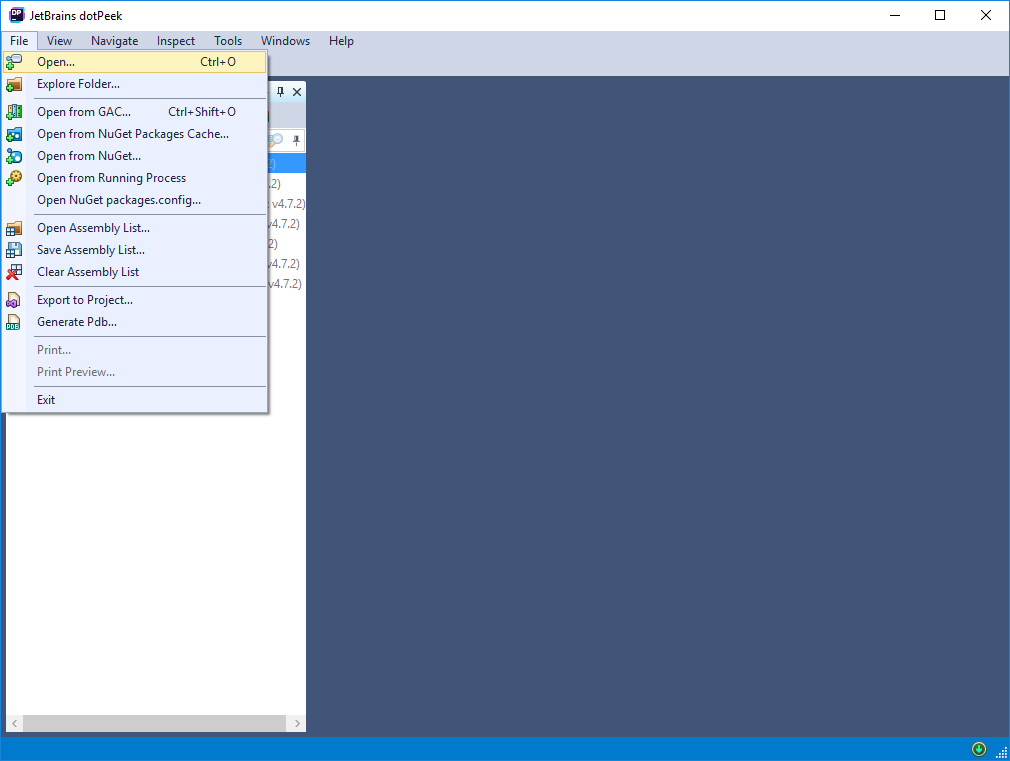
1. Then have them right-click on the process and do "Create dump file".



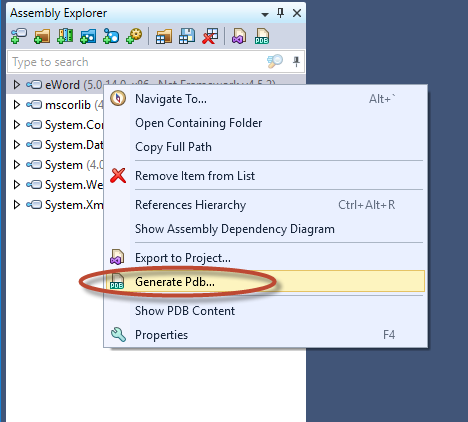
1. Have them send the dump file, it will be large, and the directory is shown below:



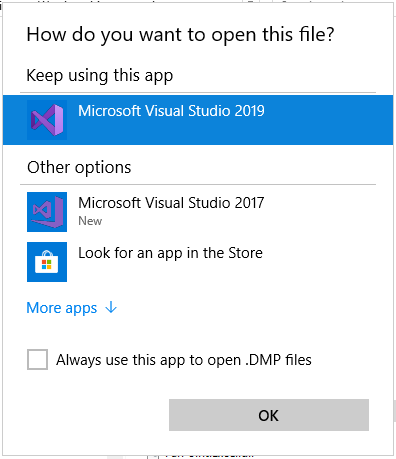
1. DotPeek (approved software) can be used to generate a .pdb, but this step is optional.
2. Open the same .exe as in the hung application:



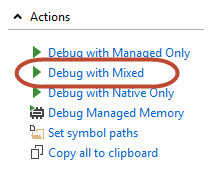
1. Right-click and generate the .pdb:



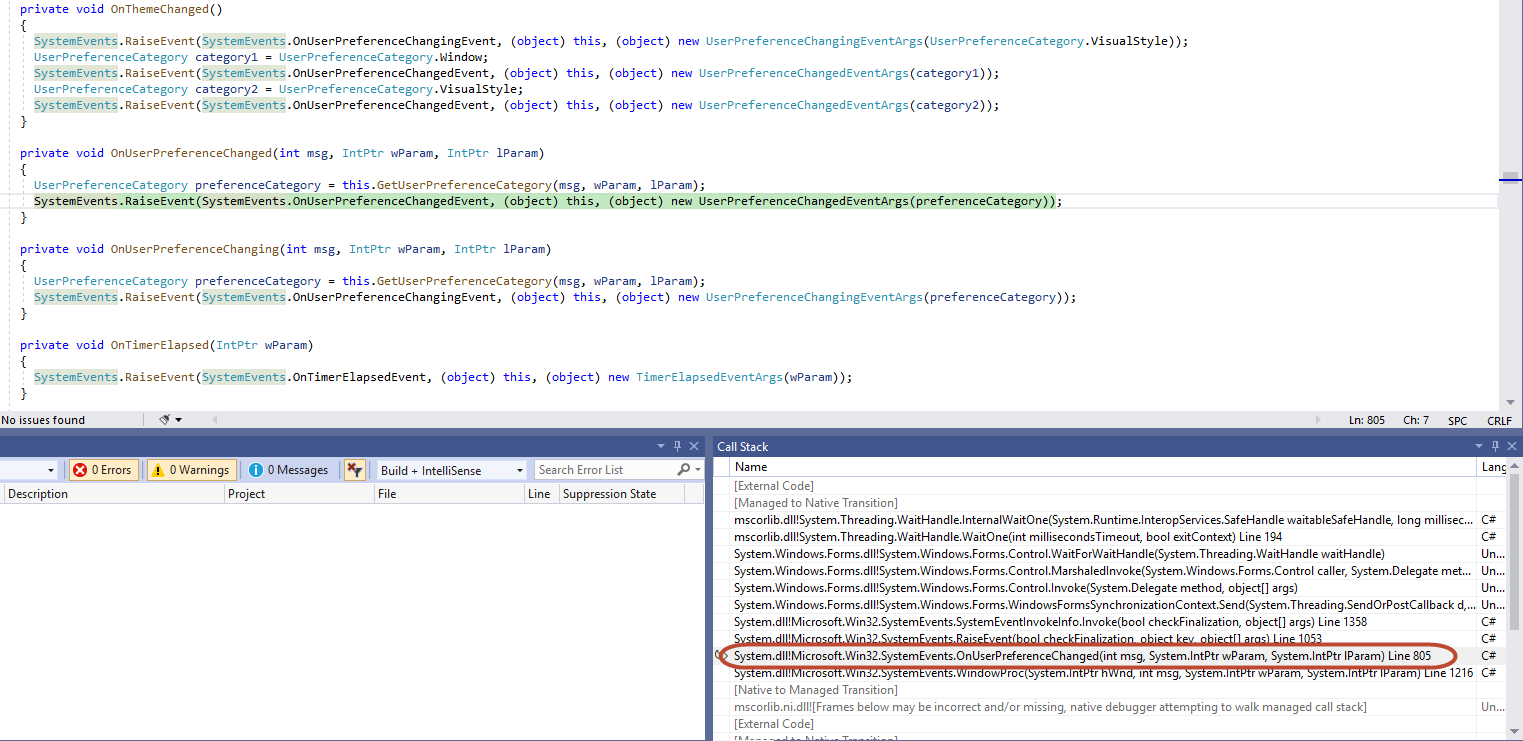
1. Place the .pdb in the same folder as the .DMP file.
2. Open the .DMP file and use Visual Studio 2019 to open it:



1. Try Debug with Mixed over to the right:



1. The call stack can now be seen, in this example, the program was hung on OnUserPreferenceChanged verified by 2 separate .DMP files, which had to do with locking the workstation:



1. And that is it, one can now see where the program has stopped. Also, I've tried WinDbg without success, this write-up gave me what I needed, WinDbg was not so helpful.

**Analyzing the CPU Profiler in Visusal Studio 2019 for bottlenecks**

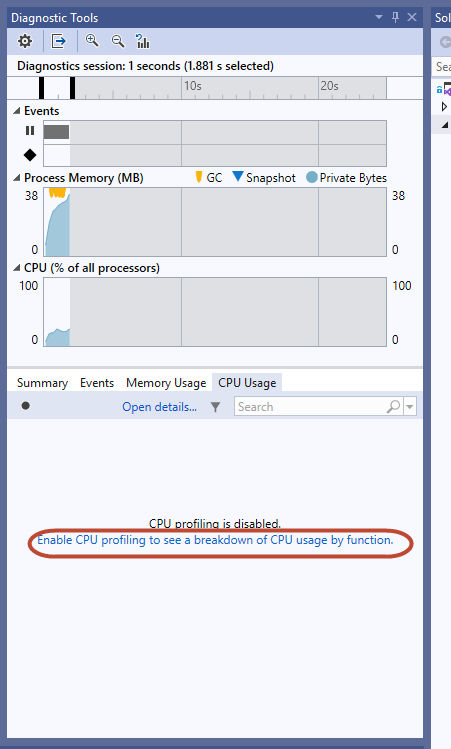


Moyer, Dan (RT-PHL)

Jan 10, 2020

Purpose: In order to speed up your application, this tutorial will show you how to profile a Windows desktop application to find bottlenecks and hopefully mitigate them to improve performance.

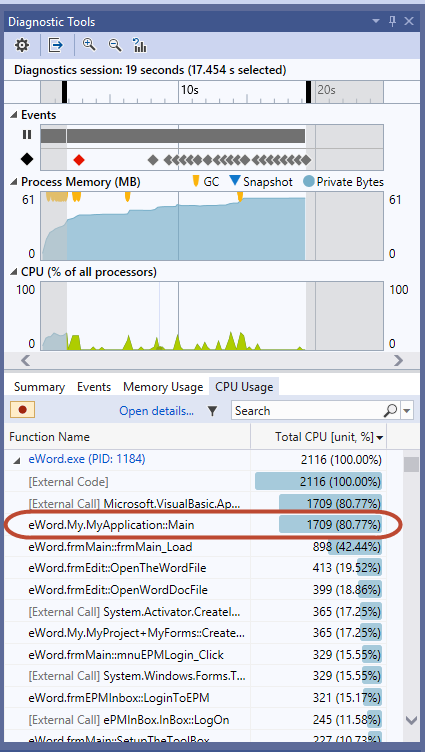
1. Create 2 breakpoints that you wish to profile between.
2. Run your Application by clicking F5.
3. Click on the following:



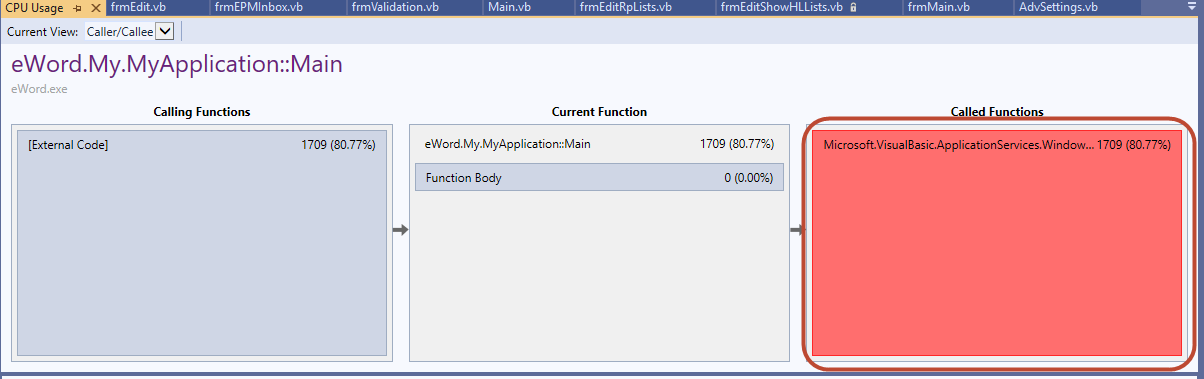
1. Click Continue:

C:\Users\ghan\Downloads\Pic\02.png

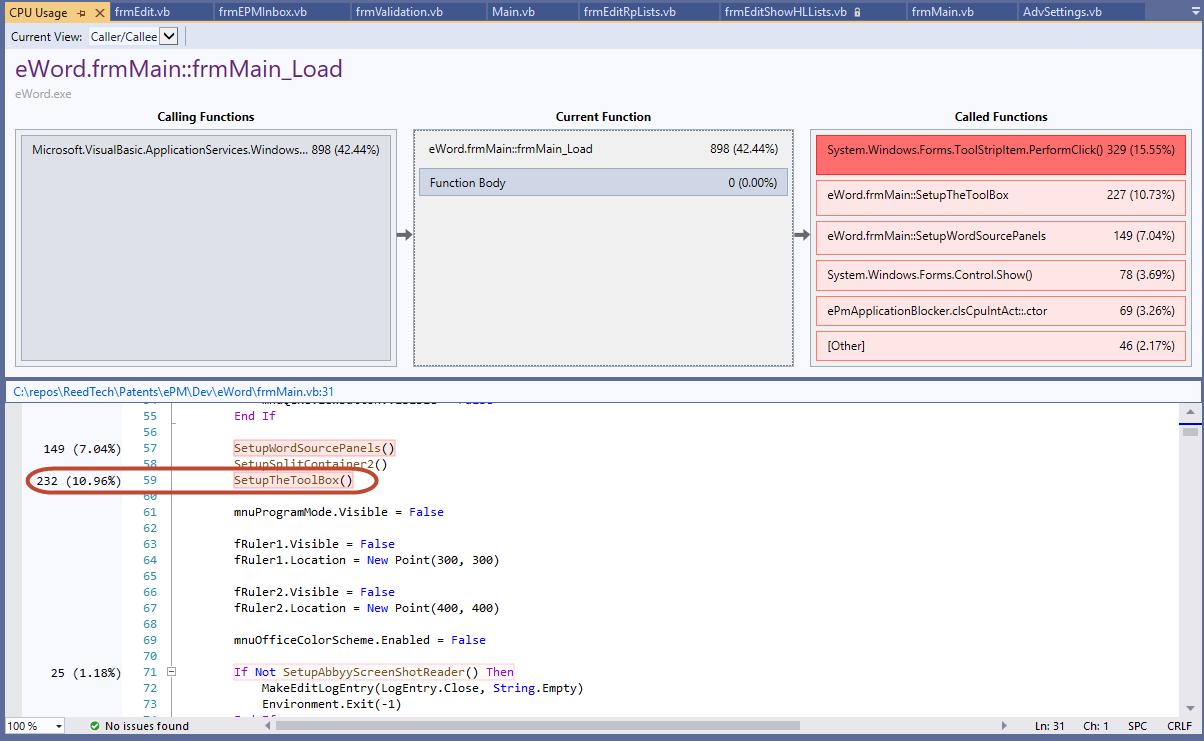
1. Use your application until the next breakpoint is hit.
2. Double-click the high CPU usage Function:



1. Click on the "Called Functions" until you find something interesting:



1. In this example, SetupTheToolBox() is causing 10.96% of the CPU usage, it would be beneficial to repeat 1 - 7 on this function to find what is taking so long:



1. To navigate around, click either Calling Functions or Called Functions.

**Creating multiple Build Configurations for client applications**



Imboden, Tom (RT-PHL)

Aug 13

These are instructions to add multiple build configurations for .NET Windows applications.

One of the biggest benefits of multiple configurations is that this will allow multiple setup projects to be created with each targeting a specific environment (QAC/DEV/QAT/QAV/PROD/TRN/UAT) without having to make separate builds or to **manually update configuration files** in order to test or deploy these builds.

This also will ensure that the correct environments are targeted when merging from Personal Branches to DEV or from DEV to MAIN. All code, including all configuration files, can then be merged without having to make any exclusions or updates.

**Contents**

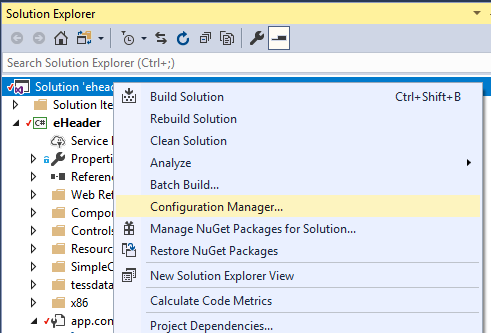
* [Creating Build Configurations for the Project](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#creating-build-configurations-for-the-project)
* [Adding SlowCheetah Visual Studio Extension](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#adding-slowcheetah-visual-studio-extension)
* [Adding Build Configuration Transform Config Files](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#adding-build-configuration-transform-config-files)
* [Adding and Previewing Environment Configuration Transformations](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#adding-and-previewing-environment-configuration-transformations)
* [Suggestion for Multiple Configurations within a Single Environment](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#suggestion-for-multiple-configurations-within-a-single-environment)
* [Maintaining Projects that Contain Build Configurations](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#maintaining-projects-that-contain-build-configurations)
* [Building Projects that Contain Build Configurations](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/148/Creating-multiple-Build-Configurations-for-client-applications#building-projects-that-contain-build-configurations)

# Creating Build Configurations for the Project

Visual Studio supports web.config transform files for web applications natively, but has no such functionality for Windows applications and their app.config files. We can, however, emulate this same behavior with a few simple steps.

**Note**: Make sure that the "Setup" project is properly loaded in Visual Studio when creating the build configurations. If it is not loaded, it may mean that the [Visual Studio Installer Projects](https://marketplace.visualstudio.com/items?itemName=VisualStudioClient.MicrosoftVisualStudio2017InstallerProjects) Extension may need to first be installed.

1. In Visual Studio, open Configuration Manager in Solution Explorer



1. Create new default configurations: **qac, dev, qat, qav, prod, trn, uat**

**Note**: Apps designed for Queue servers may need additional configurations for Patents Subdivisions (IDC/PGP/FDC/RTFM/COFC). To target these, the subdivisions can be added as additional configurations in the format "app.environment\_subdivision.config". Example: **app.dev\_idc.config** or **app.prod\_pgp.config**. The rest of the information in this Wiki applies to these configurations exactly as the other configuration types.

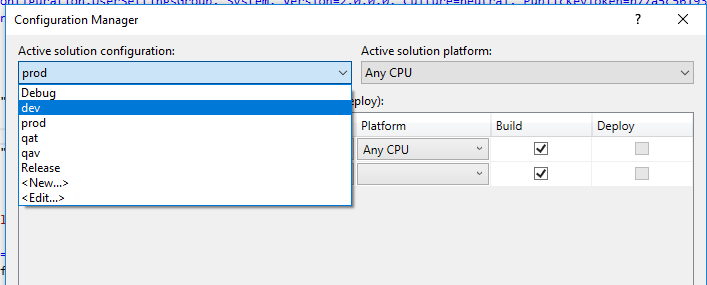
The full active list of recognized configuration names can be found in our Patents.DevOps build scripts [BuildMsiDevEnv.ps1](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_git/Patents.DevOps?path=%2FAzureDevOps%2FPowerShell%2FBuildScripts%2FBuildMsiDevEnv.ps1):

| **QAC** | **DEV** | **QAT** | **QAV** | **UAT** | **PROD** | **TRN** |
| --- | --- | --- | --- | --- | --- | --- |
| qac | dev | qat | qav | uat | prod | trn |
| qac\_idc | dev\_idc | qat\_idc | qav\_idc | uat\_idc | prod\_idc | trn\_idc |
| qac\_pgp | dev\_pgp | qat\_pgp | qav\_pgp | uat\_pgp | prod\_pgp | trn\_pgp |
| qac\_fdc | dev\_fdc | qat\_fdc | qav\_fdc | uat\_fdc | prod\_fdc | trn\_fdc |
| qac\_rtfm | dev\_rtfm | qat\_rtfm | qav\_rtfm | uat\_rtfm | prod\_rtfm | trn\_rtfm |
| qac\_cofc | dev\_cofc | qat\_cofc | qav\_cofc | uat\_cofc | prod\_cofc | trn\_cofc |

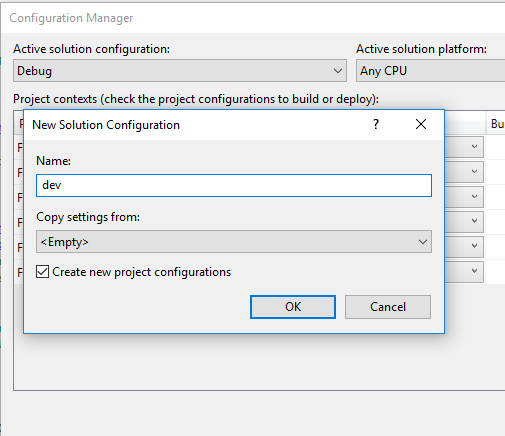
**Note**: For any new configurations you create for your solution, you will also want to add these configuration names to your .gitignore file so that these build result files and folders are excluded your Git repository. You will want to do this *before* making any builds with your new configurations; if you push build output directory (or any other) files to your Git repository before adding the exclusion patterns to the .gitignore file, the .gitignore will not delete them from the server.

In your .gitignore file, under the "# Build results" section, it should look like this. Include any alternative configuration names you have also added.

# Build results  
[Dd]ebug/  
[Dd]ebugPublic/  
[Rr]elease/  
[Rr]eleases/  
qac/  
dev/  
qat/  
qav/  
uat/  
trn/  
prod/  
x64/  
x86/  
[Ww][Ii][Nn]32/  
[Aa][Rr][Mm]/  
[Aa][Rr][Mm]64/  
bld/  
[Bb]in/  
[Oo]bj/  
[Oo]ut/  
[Ll]og/  
[Ll]ogs/



1. When adding the Solution configurations, make sure that the "Copy settings from:" is <Empty> and the "Create new project configurations" is checked.

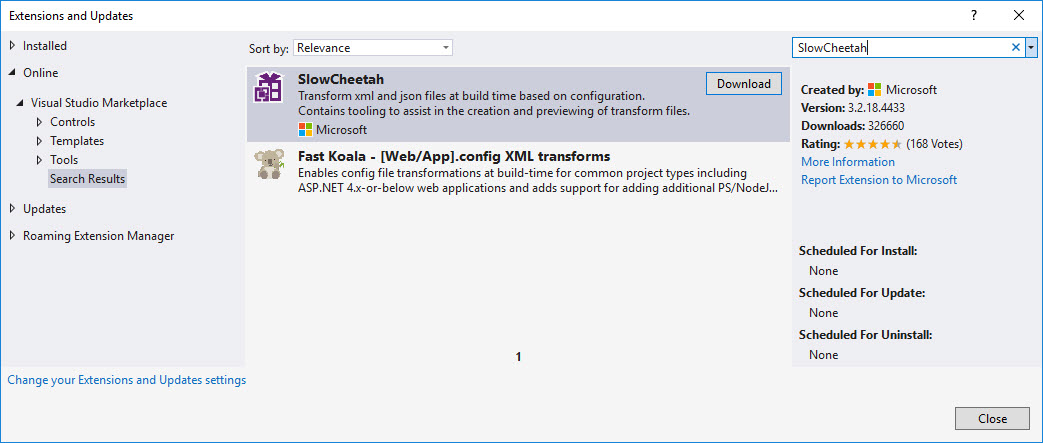


# Adding SlowCheetah Visual Studio Extension

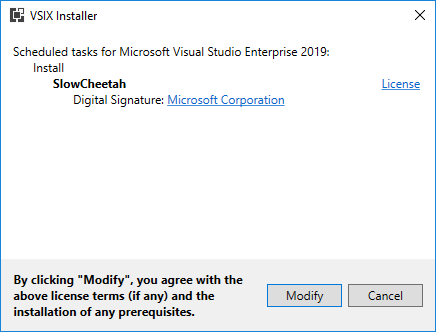
[SlowCheetah](https://marketplace.visualstudio.com/items?itemName=vscps.SlowCheetah-XMLTransforms)  is a Microsoft VS plugin that helps with creating and visualizing configuration transformations for multiple build configurations which is very helpful for Windows applications.

You can install the extension by either

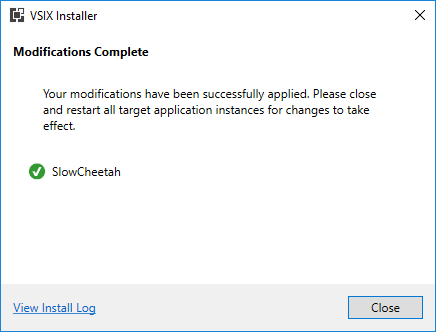
1. Go to <https://marketplace.visualstudio.com/items?itemName=vscps.SlowCheetah-XMLTransforms>  and install the extension. Visual Studio will need to be restarted after the installation.
2. In Visual Studio, go to  
   a. (VS2019) Extensions→Manage Extensions  
   b. (VS2017) Tools→Extensions and Updates...  
   and search for "SlowCheetah" under Online->Visual Studio Marketplace and Download. The extension will be installed after closing Visual Studio.



After closing all Visual Studios windows, the VSIX Installer window will pop up:

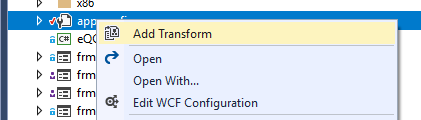


You will be notified when the installation is complete:

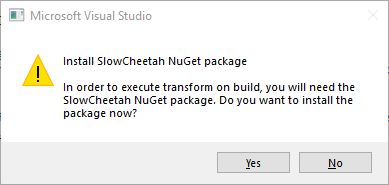


# Adding Build Configuration Transform Config Files

1. Right-click on the app.config file in the Solution Explorer and click on 'Add Transform'. This creates a transform file for every Build Configuration defined.



**Note**: SlowCheetah will ask to install the SlowCheetah NuGet package when creating these Transform files.



Please accept this request as this will add a custom step to the Project's configuration file to execute the XML transformation at the end of the build. Evidence of this SlowCheetah package is added to the project file which can be seen by opening the csproj or vbproj file in a text editor:

<Import Project="packages\Microsoft.VisualStudio.SlowCheetah.3.2.20\build\Microsoft.VisualStudio.SlowCheetah.targets" Condition="Exists('packages\Microsoft.VisualStudio.SlowCheetah.3.2.20\build\Microsoft.VisualStudio.SlowCheetah.targets')" />

<Target Name="EnsureNuGetPackageBuildImports" BeforeTargets="PrepareForBuild">

<PropertyGroup>

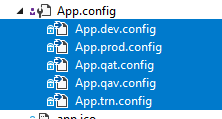
<ErrorText>This project references NuGet package(s) that are missing on this computer. Use NuGet Package Restore to download them. For more information, see http://go.microsoft.com/fwlink/?LinkID=322105. The missing file is {0}.</ErrorText>

</PropertyGroup>

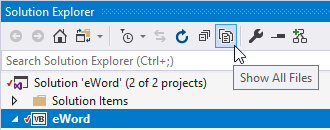
<Error Condition="!Exists('packages\Microsoft.VisualStudio.SlowCheetah.3.2.20\build\Microsoft.VisualStudio.SlowCheetah.targets')" Text="$([System.String]::Format('$(ErrorText)', 'packages\Microsoft.VisualStudio.SlowCheetah.3.2.20\build\Microsoft.VisualStudio.SlowCheetah.targets'))" />

</Target>

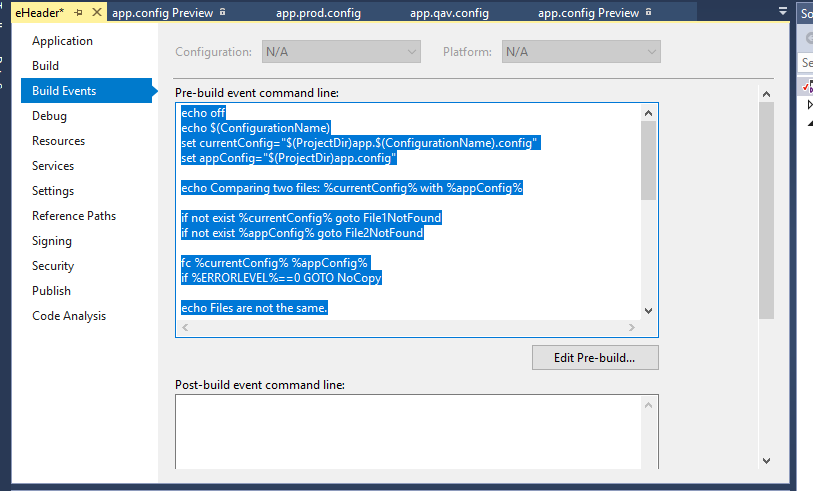
1. Delete any previously created app.[env].config files such as app.Debug.config and app.Release.config if they exist. These will no longer be needed. After this step, app.config should have all transform files attached to it.



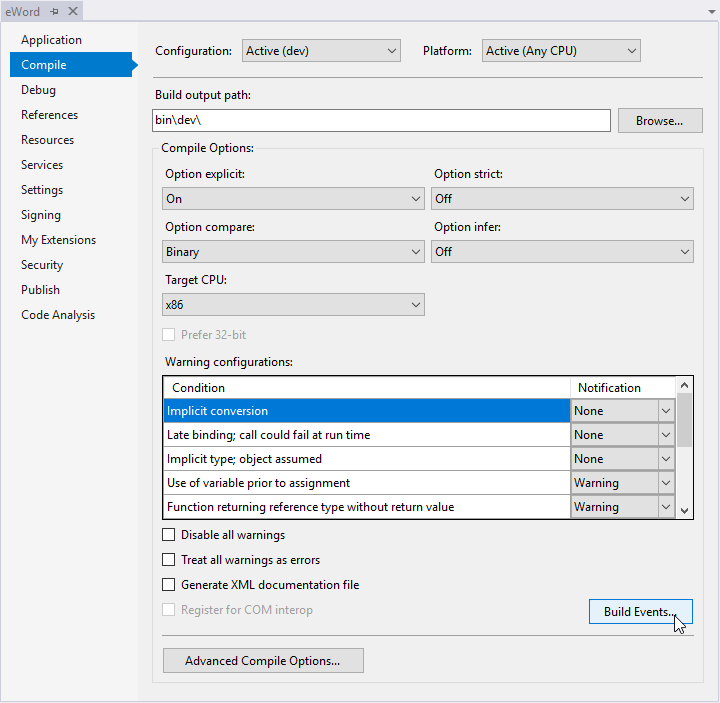
For VB projects in Visual Studio, the dependent files are not shown by default. This is a ["historical" difference](https://developercommunity.visualstudio.com/content/problem/165081/vb-project-not-showing-dependentupon-files-anymore.html) maintained by Visual Studio to show fewer files than C# projects. In order to show these files in a VB project, you must choose "Show All Files" with the project selected in Solution Explorer:



1. Existing projects may have a custom post-build step that copies any existing app.Debug.config or app.Release.config over the app.config file during a build. We do not want this behavior as we will be using transforms instead, so this script needs to be removed by going to the Project Properties and removing it. Here is what that script usually looks like and where to find it.



In VB projects, the Build Events can be accessed from a button under the project Properties window's Compile tab:

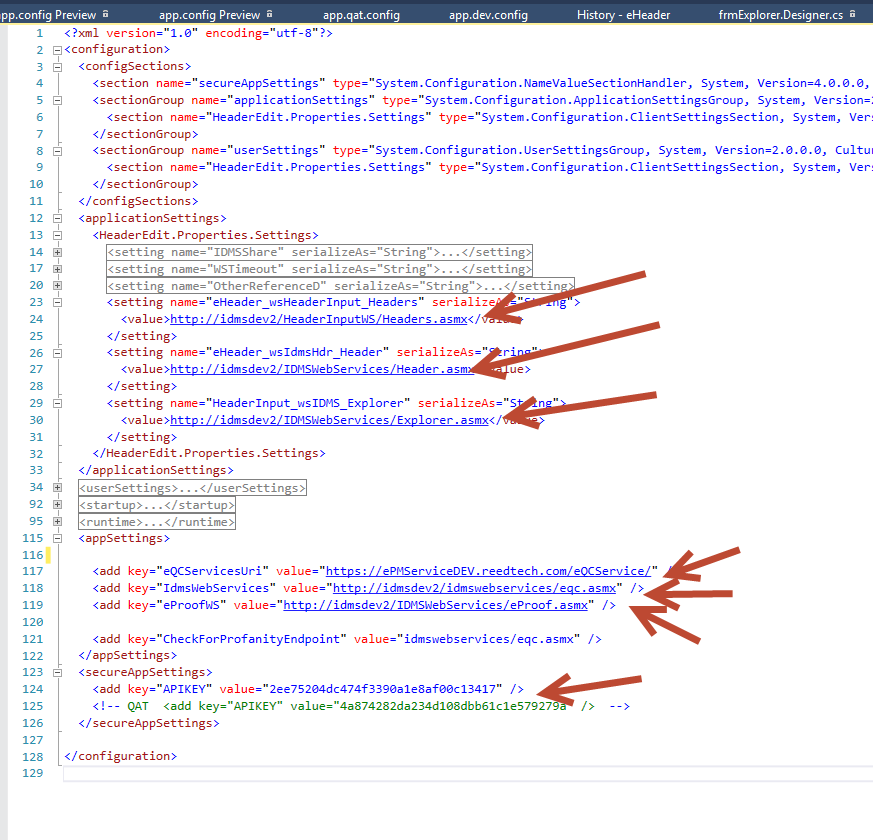


# Adding and Previewing Environment Configuration Transformations

1. Open the base **app.config** file and identify all settings that are environment specific. Some typical settings that will be environment- or configuration-specific:  
   **a.** Internal web service or API URLs, starting with http:// or https://, or contain the name of a known web server, Big-IP/load balancer name, or web service.  
   **b.** Internal network file server paths, starting with \\ such as \\brutus\ or \\porcia\ or \\caesar\; or much less common, a mapped drive beginning with a letter other than C:\ or D:\.  
   **c.** "Send to" or "cc" email addresses  
   **d.** Database names or Oracle database aliases as found in the TNSNames.ora file.  
   **e.** Some workflow indicators (must be evaluated for each application):
   1. Device Names or Intelligent Agent names (names ending with " IA").
   2. Task names that have a "Pxx" or "Gxx" suffix suggesting a team number.
   3. Team numbers or Team IDs.

**f.** See further for [Suggestion for Multiple Configurations within a Single Environment](https://tfsprod.reedtech.com/tfs/reedtech/ReedTech/_wiki/wikis/ReedTech.wiki?wikiVersion=GBwikiMaster&_a=edit&pagePath=%2FReed%20Tech%20Developer%20Wiki%2FPatent%20Team%20Wikis%2FSoftware%2FCreating%20multiple%20Build%20Configurations%20for%20client%20applications&pageId=148&anchor=suggestion-for-multiple-configurations-within-a-single-environment).

For instance, in eHeader there are these 7 environment-specific settings:



**It is very important to keep in mind that the transformations should only target and replace these specific environment settings, not to be a wholesale copy/paste of the entire app.config file.**

1. As an example, here is how to modify the **app.qat.config** settings to transform the services references and the APIKEY to target the QAT environment:

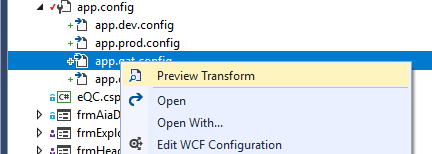


Notice, for ApplicationSettings, the **xdt:Transform="Replace"** attribute is used to replace the entire <value> XML node while for AppSettings keys the **xdt:Transform="SetAttributes"** just sets the value attribute. The **xdt:Locator** attribute is used to find the matching node using a particular attribute.

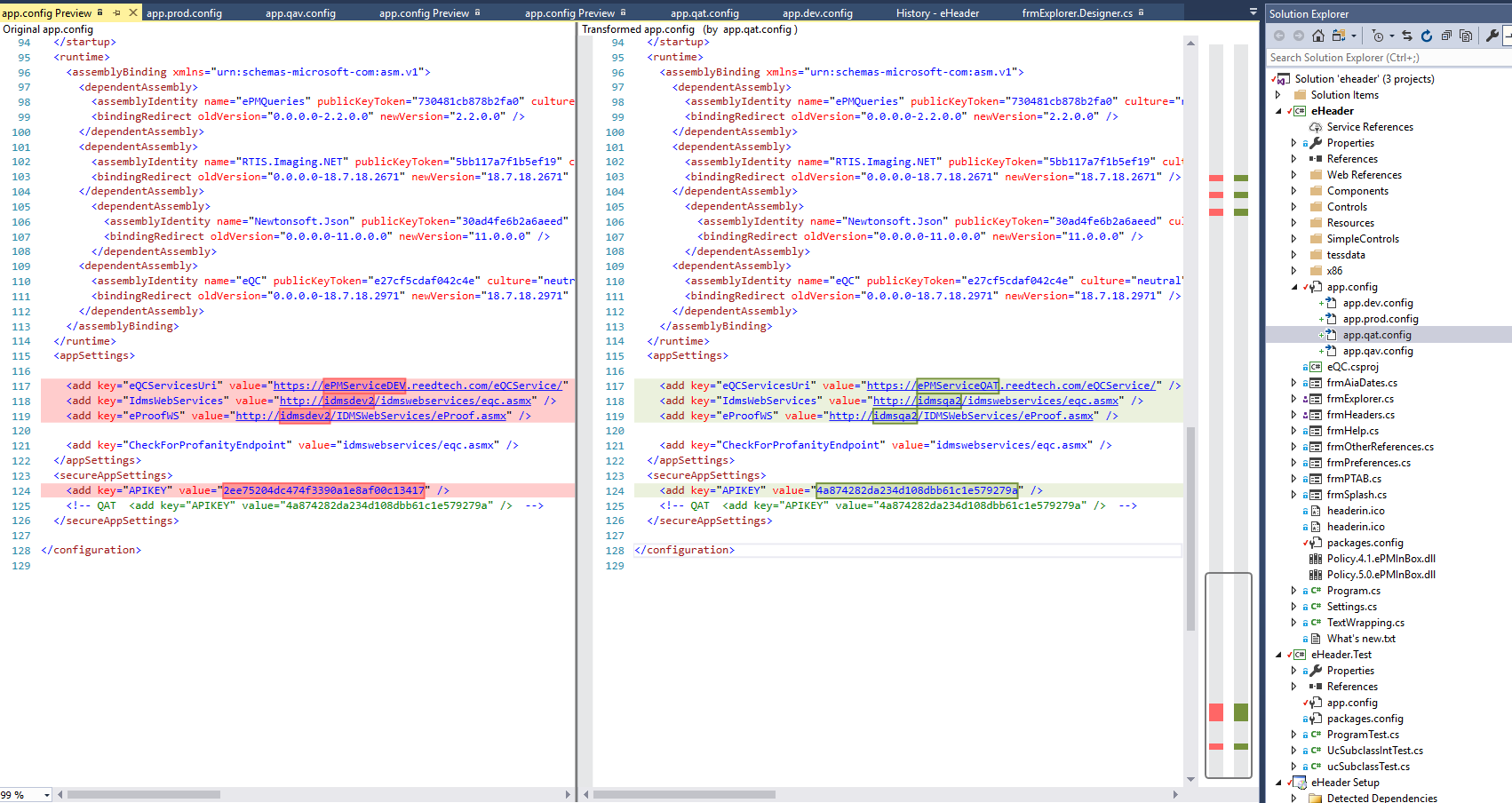
For more details on transformation options go to <https://docs.microsoft.com/en-us/previous-versions/dd465326(v=vs.100)>

IDMS Web Service locations: <https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_wiki/wikis/ReedTech.wiki/469/IDMS-Web-Services>

1. To preview the transformation to make sure it will work correctly, take advantage of SlowCheetah's **Preview Transform** feature by right clicking on a transform config file:



This will show the comparison between the base App.config and how it will look once it is transformed during the build process.

****

1. These transformations should now be added to all the environment transformation .config files:

* **app.qac.config**
* **app.dev.config**
* **app.qat.config**
* **app.qav.config**
* **app.uat.config**
* **app.prod.config**
* **app.trn.config**

This should be done even when just working from the Personal or DEV branches. When it is time to merge the code up to the MAIN branch, all of these configurations will be ready to go and nothing will need to be excluded or updated.

More configurations can be added as needed for particular deployment needs. There are configurations for

* ePM workflow subdivisions (PGP, IDC, RTFM, FDC);
* multiple workflows making shared use of projects (COFC);
* or production process disciplines (TEXT or CWU).

See "Building Projects that Contain Build Configurations" below for the full list of build configurations currently available.

# Suggestion for Multiple Configurations within a Single Environment

Some programs may have multiple installation configuration within one environment, such as queue programs configured for different workflow departments. An alternative use of the config file transforms is to replace entire appSettings or other department-sensitive section within the app.config file.

This example comes from [BuildComp](https://tfsprod.reedtech.com/tfs/reedtech/ReedTech/_versionControl?path=%24%2FReedTech%2FPatents%2FePM%2FDev%2FBuildComp), which has different file locations per environment and department, and has three separate installation configurations per environment to run for PGP, IDC, and FDC.



By replacing the entire section, you can also maintain comments to help determine how to configure the application after installation.

Each app.[environment].config will look the same except for the setting values that change between environment, such as web service URLs and network folder locations.

# Maintaining Projects that Contain Build Configurations

* If a non-environment setting is added to app.config, nothing needs to be done to the transformation .config files.
* If a new environment setting is added to app.config, it will need to be added to each transformation .config file with the appropriate replacement value for each environment.
* If a particular environment setting changes, simply update the appropriate transformation .config file, such as app.prod.config for a production environment change.

**Note**: With the build transformations in place, you do not have to worry about the state of the environment-specific values in the base app.config file when checking into source control. Dev-branch code does not need the base app.config file to be configured for QAT, and Main-branch code does not need to be configured for Production.

When developing and debugging locally, you can set the active solution configuration to the standard "Debug" and, because there is no transform file for "debug" (i.e., no app.debug.config file), the app.config file is not modified on build. If you need to work with a local web service for debugging, you can set your app.config settings to point to localhost and not need to worry about switching them back to some other environment configuration when done. Alternatively, if you want to connect directly to the Dev environment, you can just change your active solution configuration to "dev" and all the settings will be transformed automatically on build.

Alternatively, you can create more configuration transforms if you want to maintain preset configurations for differernt scenarios.

# Building Projects that Contain Build Configurations

The Build Definition that is used to build the applications along with with their MSI setup installer is now capable of determining if custom Build Configurations are present and to automatically create installers for each custom Build Configuration.

Current configuration names recognized by [BuildMsiConfiguration.ps1](https://tfsprod.reedtech.com/tfs/ReedTech/ReedTech/_versionControl?path=%24%2FReedTech%2FPatents%2FAutomationTools%2FDev%2FBuildMsiConfiguration.ps1):

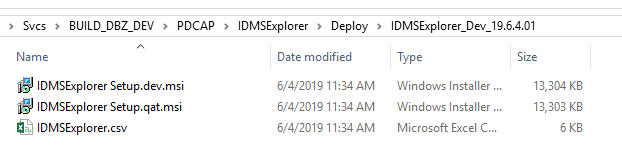
DEV and MAIN branch build configuration environment names:

| **Environment** | **QAC** | **DEV** | **QAT** |
| --- | --- | --- | --- |
|  | qac | dev | qat |
|  | qac\_cofc | dev\_cofc | qat\_cofc |
|  | qac\_fdc | dev\_fdc | qat\_fdc |
|  | qac\_idc | dev\_idc | qat\_idc |
|  |  | dev\_idc\_text | qat\_idc\_text |
|  | qac\_pgp | dev\_pgp | qat\_pgp |
|  |  | dev\_pgp\_cwu | qat\_pgp\_cwu |
|  |  | dev\_pgp\_text | qat\_pgp\_text |
|  | qac\_rtfm | dev\_rtfm | qat\_rtfm |

Additional MAIN branch build configuration environment names:

| **Environment** | **QAV** | **UAT** | **PROD** | **TRN** |
| --- | --- | --- | --- | --- |
|  | qav | uat | prod | trn |
|  | qav\_cofc | uat\_cofc | prod\_cofc | trn\_cofc |
|  | qav\_fdc | uat\_fdc | prod\_fdc | trn\_fdc |
|  | qav\_idc | uat\_idc | prod\_idc | trn\_idc |
|  | qav\_idc\_text |  | prod\_idc\_text |  |
|  | qav\_pgp | uat\_pgp | prod\_pgp | trn\_pgp |
|  | qav\_pgp\_cwu |  | prod\_pgp\_cwu |  |
|  | qav\_pgp\_text |  | prod\_pgp\_text |  |
|  | qav\_rtfm | uat\_rtfm | prod\_rtfm | trn\_rtfm |

Here is a what IDMSExplorer built out of the DEV branch looks like after Build Configurations have been added:



When run, these Setup.[env].msi installers will install the application with the proper configuration files pointing to the environment in the Setup file name which means that there is no longer any need to manually update .config files after installing.