

### Dependency Management with NuGet – Foreword

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, you should not interpret this to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft grants you a license to this document under the terms of the Creative Commons Attribution 3.0 License. All other rights are reserved.

© 2014 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.



### Dependency Management with NuGet – Foreword

# Table of Contents

Foreword	. 4
Introduction	5
Understanding NuGet6	
Managing Shared Resources with NuGet	6
Using NuGet for Dependency management	1
Additional references	15
Hands-on Lab (HOL) – Using NuGet	5
Exercise 1 – Find and install a NuGet Package	
Exercise 2 – Using Package Manager Console	)
Exercise 3 – NuGet Package Manager	)
In Conclusion	22



## Foreword

Since the first writing of the TFS branching guide, a lot has changed in the world of version control. Hosted version control solutions are everywhere, and many of them include integration with build, project tracking, and other services. Distributed Version Control is no longer a niche, and has changed the way that many developers think about what it means to version their code. More developers using version control than ever before – and this is a great thing for the billions of end users of those software development projects.

More developers are using version control. This means developers need solid, practical, and easy-to-digest guidance that is industry proven now more than ever. This guide, and those that came before it, strive to do just

that – provide the version control guidance that development teams need to be effective while being approachable and flexible. In this latest edition, we have streamlined the guidance and simplified concepts with the goal of helping teams of all shapes and sizes to develop strategies that enable the flexibility and agility that modern development teams demand.

I also need to mention that this guide would not be in its current form without the readers. Thanks to all of you whom have contributed your feedback and ideas that have helped shape this guide over the years. As is in the past, if you see something in this guide that you would like to see changed or improved, please let us know!

Happy versioning!

Matthew Mittrik - Program Manager, Cloud Dev Services



## Introduction

This guide aims to provide an insight and practical guidance on dependency management and managing shared resources, using NuGet with Visual Studio. It is one of a set of companion guides as outlined in chapter "What's New" in the TFS Version Control Part 1 - Branching Strategies guide.

### Intended audience

In this guide, we primarily address Garry, the development lead, Doris, the developer, and Dave, the TFS administrator who are new to NuGet. See ALM Rangers Personas and Customer Profiles 1 for more information on these and other personas.

### Visual Studio ALM Rangers

The Visual Studio ALM Rangers provide professional guidance, practical experience, and gap-filling solutions to the ALM community. They are a special group with members from the Visual Studio Product group, Microsoft Services, Microsoft Most Valuable Professionals (MVP), and Visual Studio Community Leads. Membership information is available online<sup>2</sup>.

#### Contributors

Alan Wills, Anna Galaeva, Howard Dierking, Krithika Sambamoorthy, Stawinski Fabio, Taavi Kõosaar and Tommy Sundling A special thank you to the ALM Ranger teams who laid the foundation with v1 and v2: Anil Chandr Lingam, Bijan Javidi, Bill Heys, Bob Jacobs, Brian Minisi, Clementino de Mendonca, Daniel Manson, Jahangeer Mohammed, James Pickell, Jansson Lennart, Jelle Druyts, Jens Suessmeyer, Krithika Sambamoorthy, Lennart Jansson, Mathias Olausson, Matt Velloso, Matthew Mitrik, Michael Fourie, Michael Learned, Neno Loje, Oliver Hilgers, Sin Min Lee, Stefan Mieth, Taavi Koosaar, Tony Whitter, Willy-Peter Schaub, and the ALM Community.

Using the sample source code, Errata and support

All source code in and revisions of this guide are available for download from the CodePlex site Version Control from the Hold Revision for the team using the CodePlex discussion forum.

### Additional ALM Rangers Resources

Understanding the ALM Rangers – <a href="http://aka.ms/vsarunderstand">http://aka.ms/vsarunderstand</a> Visual Studio ALM Ranger Solutions - <a href="http://aka.ms/vsarsolutions">http://aka.ms/vsarsolutions</a>

<sup>3</sup> http://aka.ms/treasure18



http://vsarguidance.codeplex.com/releases/view/88001 http://aka.ms/vsarindex

## Understanding NuGet

### Managing Shared Resources with NuGet

Sharing and managing resources between projects can be complicated. *NuGet* helps simplify the whole process. You have probably already used NuGet in Visual Studio to download extensions from the public NuGet gallery. However, you can also set up your own local repository to share resources in your project or your organization. "NuGet is a Visual Studio Extension that makes it easy to install and update libraries and tools in Visual Studio" (<a href="http://nuget.org">http://nuget.org</a>). It comes pre-installed on Visual Studio 2012 and later versions, and you can install it as an extension in earlier versions. There is also a standalone tool, NuGet.exe, which you can use outside Visual Studio.

#### Here is what NuGet provides:

- A NuGet package can install shared resources, update project references, install Visual Studio extensions, and
  update project files. It can provide everything you need for doing development in a particular domain.
- Private repository You can create your own NuGet package repository, either through file sharing or as a
  web server.
- Public repository You can also access and contribute to the official <u>NuGet gallery</u> <sup>4</sup>. As well as other public repositories such as myget.org
- Easy overview of which installed packages are outdated
- Easy upgrade process when shared packages are updated, including for dependencies of those packages
- Command-line tool can run in the build process to automatically download the right versions of shared resources

We are basing this guidance on NuGet version 2.7.1. This guidance is complementary to the <u>NuGet tool</u> documentation<sup>5</sup>.

The aim of this guidance is to offer an overall approach on how NuGet helps manage shared resources and dependencies when working with projects using Visual Studio and Team Foundation Server.

### When should you consider NuGet to share resources?

- If the project uses NuGet for external dependency management (for example libraries such as Entity Framework, jQuery, or ASP.NET MVC), then you can simplify management using the same approach for internal dependencies.
- If several projects use the shared resources then NuGet helps simplify sharing and managing those resources including dependencies and versions.
- If a project's shared resources have complex dependencies between them.
- If there is a separate team (or teams) that build shared resources and many other projects use those resources.

As you see, most scenarios that involve references are a great fit for NuGet. However, if you are working with a single solution that has projects in that solution that depend on each other you can still simply leverage project references and forgo the NuGet package scenarios. You should likely only do this if you do not intend to share

those projects beyond that single solutions boundary since NuGet would make it easier to package and share

<sup>&</sup>lt;sup>4</sup> http://nuget.org/ <sup>5</sup> http://docs.nuget.org/



the project artifacts in a scalable way that can reach many solutions easily. It is not necessary to decide early on, because turning component(s) into sharable NuGet package(s) is straightforward.

### Using NuGet for sharing resources

NuGet consists of the following components:

- NuGet specification file (.nuspec) a specification for the creation of a NuGet package. This defines
  common information about the package such as title, description, version, etc. It also defines dependencies
  on other NuGet packages.
- **NuGet package (.nupkg)** a package in NuGet format (Open Packaging Conventions based Zip) that contains the shared resources and libraries. The content of the package is determined by the <u>NuGet</u>
- specification file
   NuGet feed a local folder, UNC share or a NuGet RSS feed built using the NuGet.Server component.
- NuGet.exe a standalone command line tool.
- NuGet Package Manager and NuGet Package Manager Console a Visual Studio Extension with which you manage NuGet packages for a specific solution
- **NuGet.config** <u>NuGet configuration per machine or per solution</u><sup>7</sup>. Among other things, one can define common repositories (e.g. internal NuGet feeds) for the team or the location of packages for a specific solution.

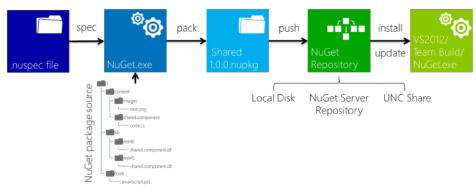


Figure 1– The general flow with NuGet

### Creating a NuGet package

Step	Instructions
1 Create package structure	First, create a folder structure for packaging. You have to follow the documented NuGet package folder structure 8. The structure shown below supports three type of resources – assemblies (dll's), tools (PowerShell scripts/programs that can be run from the Package Manager Console), content (different files that will become part of the target project on installation such as images, C# code files, web files, css) and build (MSBuild targets files that are inserted into the project). Note: Keep in mind you can also pass the

<sup>6</sup> http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#From\_a\_convention\_based\_working\_directory

<sup>8</sup> http://docs.nugetorg/docs/creating-packages/creating-and-publishing-a-package#from\_a\_convention\_based\_working\_directory



<sup>&</sup>lt;sup>7</sup> http://docs.nugetorg/docs/release-notes/nuget-2.1#Hierarchical Nuget.config

proj file directly to a pack command that can help generate the nuspec based on the assembly data automatically via the command line. (See Step 3 below) NuGet package structure \lib\net40\Shared.Component.dll \lib\net45\Shared.Component.dll \tools\SmartScript.ps1 \content\Images\nice.png \content\Shared.Component\Code.cs The lib folder can contain subfolders for specific framework versions (such as net40, net45). When someone installs your studies was the assembly directly will the liberdropen of another target yearier references the contraction of th tools folders can also contain framework specific folders. Creating this folder structure can be an automated build step. Create a NuGet package specification <sup>9</sup>. You can use Nuget.exe to do this (and there are other ways). Let us Create the say you want to create a package that only contains Shared.Component.dll. Run: specificatio nuget.exe spec "Shared.Component.dll' n That generates the xml file Shared.Component.dll.nuspec. It needs further modifications to include correct 🗌 - Done metadata about the package. You can find detailed .nuspec file information in NuGet Docs There is also a <u>GUI Package Explorer</u><sup>11</sup> tool to help ease the creation and modification of NuGet packages. The illustration below shows a sample specification for the "Shared.Component.dll" component <?xml version="1.0"?> -<package > <metadata> <id>Shared.Component.dll</id>
</er>
<version>1.0.0
<authors>ALN Ranger</authors> cauthors>ALN Ranger</authors>
cowners>ALN Ranger</owners>
clicenseUrl>http://licenseserver/view</licenseUrl>
cprojectUrl>http://projectserver/view</projectUrl>
cconurl>http://conicon/Shared.component.gif
clicenseAcceptance>false</re>
cescription>This is an internal shared component
cetaction
credictionseAcceptance>false</re>
component
component 11 12 13 14 15 <tags>Net40 CSharp Common</tags> 16 17 18 19 </metadata> </package> The specification defines the version of the component and any dependencies it might have on other NuGet packages. In the sample, there is a dependency on Shared.Core.dll from version 1.0.2 up to version 3.0.5 (included). You can find more information on versioning in NuGet documentation 12 You can also create a specification based on a Visual Studio project by using the "spec" command within the project folder.

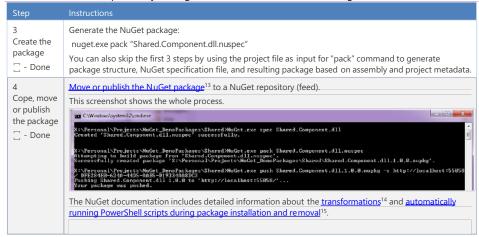
<sup>12</sup> http://docs.nugetorg/docs/reference/versioning



<sup>9</sup> http://docs.nugetorg/docs/creating-packages/creating-and-publishing-a-package#Create\_the\_manifest

<sup>10</sup> http://docs.nugetorg/docs/reference/nuspec-reference

<sup>11</sup> http://docs.nugetorg/docs/creating-packages/using-a-gui-to-build-packages



### Getting an update of a NuGet package

NuGet allows updating of package(s) on Project or Solution level. However, you should do thorough checks per package to verify that updating will not cause any unexpected problems within the solution or project. One approach is to use the –what if switch from the PS console to view the dependency chain that will be updated without actually performing the operation.

### We recommend:

 Updating the package on the solution level to maintain the same version throughout projects within the solution.

### We do **not** recommend:

- Updating all packages in the solution at the same time, unless you have done checks for all the packages.
- To have different versions of a NuGet package applied to different projects in the same solution. It could
  lead to build problems that would be difficult to track down. Exceptional cases may need different
  versions but mostly only temporarily. However if the packages are just for things like scripts in web
  applications such as jQuery, this may be ok in your scenario.

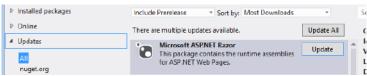


Figure 23– Updates available from Manage NuGet Packages window

 $http://docs.nugetorg/docs/creating-packages/creating-and-publishing-a-package\#Automatically\_Running\_PowerShell\_Scripts\_During\_Package\_Installation\_and\_Removal and the property of the prope$ 



 $<sup>^{13} \ \</sup> http://docs.nugetorg/docs/creating-packages/creating-and-publishing-a-package$ 

<sup>14</sup> http://docs.nugetorg/docs/creating-packages/configuration-file-and-source-code-transformations

Before updating, check the release notes of the NuGet package. Does it contain any functional changes that might break your code? If it has new dependencies, how will that affect your project?

Starting with NuGet v2.8 it is possible to use the PowerShell standard <u>—whatif switch</u> <sup>16</sup> on PS Console with install-package, uninstall-package and update-package commands that provides a dry-run capability before applying the change.

The conventions of <u>semantic versioning</u><sup>17</sup> provide a useful guide to the likely impact of a change on your project. Briefly, in the version number {Major}. {Minor}. {Patch}:

- Major updates may contain incompatible API changes
- Minor updates should contain only backwards-compatible changes
- Patch updates should contain only backwards-compatible bug fixes

### Creating a NuGet feed

A NuGet repository is a feed that NuGet Package Manager, NuGet Package Console, NuGet.exe, or other NuGet tools can consume. The feed is a centralized repository for accessing the NuGet packages that you have pushed to the repository. There are three types of NuGet feeds, but only one of them supports comprehensive capabilities such as rss feed or pushing.

- Local folder —the simplest way to create a feed is to create a folder on local disk and copy the .nupkg into it. You can configure this location in Visual Studio as the NuGet feed location where your projects can consume the packages.
- Shared folder—Use a shared location such as a UNC share. Copy new packages to the share, as they become available Again it is a simple feed configuration in Visual Studio. Figure 3 illustrates the UI from where you can add new feeds.
- NuGet server With your own NuGet web application, you get update notifications and other benefits.
  There is a NuGet package to help you. You can find <u>detailed documentation in the NuGet website</u><sup>18</sup>. It is also possible to <u>clone the full nuget.org gallery implementation</u><sup>19</sup>.

After you have set up any of these repositories, you can configure your Visual Studio installations to use them by opening Options, Package Manager, Package Sources. By default, Visual Studio 2013 is configured to read the public feed from NuGet.org.

<sup>19</sup> https://github.com/NuGet/NuGetGallery/blob/master/README.markdown



Page 10 of 22

http://docs.nugetorg/docs/release-notes/nuget-2.8#Preview\_NuGet\_Operations\_With\_-whatif

<sup>17</sup> http://semver.org/

<sup>18</sup> http://docs.nuget.org/docs/creating-packages/hosting-your-own-nuget-feeds

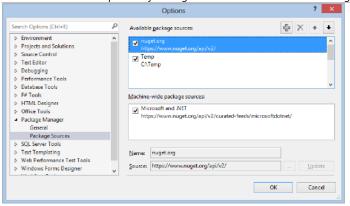


Figure 3- NuGet Package Manager configuration options in Visual Studio 2013

### Using NuGet for Dependency management

NuGet makes it easy to share resources. NuGet has been around for a few years. It is constantly evolving, maturing, and gaining more and more ground<sup>20</sup> in the so ftware development community.

### Version Control Storage Locations for Shared Resources

Should you keep NuGet packages in version control?

For the source of the package, the answer is obviously yes – just as you would for the code of any assembly. In the target projects where the packages are used, there are two approaches:

- Do not add NuGet packages to version control. Instead, use the NuGet Package Restore pattern, which
  we will describe in the next chapter.
- Add NuGet packages to version control like your source code. Configure NuGet to download one version
  of each package and keep it in a location that you can share across the team project.

### Types of resources that can be shared with NuGet

NuGet packages can contain <u>four types of resources</u><sup>21</sup> – assemblies, tools, content (such as images, css, JavaScript, razor views, c# code files) and build (MSBuild targets files that you insert into the project). It is possible to make packages that contain only compiled assemblies or only content files or both. Additionally you can structure <u>each assembly or content for NuGet to be install only for specific .Net framework versions</u><sup>22</sup>. Another important aspect to remember is that NuGet packages can contain useful custom installation/uninstallation and configuration transformation scripts per package and even per framework version.

### Dependency Management Pattern with Central NuGet feed(s)

The main pattern for shared resources and dependency management with NuGet is to use internal and external NuGet feed(s). By default, NuGet has knowledge of the external public NuGet Gallery feed, through which you

<sup>22</sup> http://docs.nuget.org/docs/creating-packages/creating-and-publishing-a-package#Supporting\_Multiple\_NET\_Framework\_Versions\_and\_Profiles



<sup>&</sup>lt;sup>20</sup> http://stats.nuget.org/

 $<sup>^{21} \</sup>quad http://docs.nugetorg/docs/creating-packages/creating-and-publishing-a-package\#From\_a\_convention\_based\_working\_directory$ 

can find standard publicly released packages such as NHibernate, jQuery, Entity Framework, ASP.NET MVC. Using the NuGet tools (such as Package Manager) you can install new NuGet packages to the project or update existing ones. The NuGet Visual Studio Extension will notify you when new versions of installed packages become available. When you install a NuGet package to your project, NuGet will automatically install all dependencies. This pattern has three threads.

- Provide a NuGet feed. It can be either local directory, UNC share or a real NuGet.Server based repository.
   To have a feed that is available to everyone in the team, use a UNC share or NuGet.Server feed. It is very easy to publish a package into a UNC share as you just copy it into the shared directory. Creating a NuGet.Server feed requires a small amount of development work (or at least the setup of cloned nuget.org gallery), and you use the NuGet utility to publish packages. The main advantage of the NuGet server is that it can notify users of new versions and will scale better for searching and browsing, as there are more packages in the
- **PRODUITE ମଧ୍ୟ କଳେ ଓ ଅନ୍ୟାଧି ଓ ଅଧିକର୍ପ କଳି ।** Belief ବୁର୍ମ ହେଉଥିଲି । Ine utility nuget.exe, you can automate the creation and publication of new packages with the help of MSBuild or Team Foundation Build.
- Listen to the feed. In case you plan to store the packages in Version Control, then in each project that uses a package, just one team member should have responsibility for subscribing to the feed, installing the package, and subsequently getting updated versions from the feed. That user should install the package in his own environment, and then check the package into version control, just like any other changes. Other team members get the package contents through version control—they do not subscribe to the feed. This method ensures that everyone in the project is using the same version of the assemblies or other resources from the package. Optionally, you could set permissions that prevent other team members from updating the resources from the package.

You can also decide to avoid storing packages in Version Control and use "Dependency Management Pattern using Package Restore." In this scenario, all development environments would access the feed and download missing packages when needed. Keeping dependencies, and often this may mean binaries out of version control can provide several benefits.

You can set Visual Studio to work with more than one NuGet repository at the same time. You can now add packages from both internal (visible within company) and external repositories. One scenario for an organization might be to mirror "approved" packages from a public feed to a private feed so that developers only work with approved packages. In your internal repository, you can also include packages of internal components that depend on packages from any of the repositories (incl. external such as the NuGet feed) and NuGet will automatically install those dependencies, as you would expect. Additionally when searching for a package, NuGet will look for it in each repository in turn. If you enable package restoration, NuGet uses this same behavior of sequential search in each repository to get dependent packages.



Figure 47– List of available package sources

### The downloaded NuGet packages location

By default, NuGet places all downloaded packages within the solution folder. However, depending on the version control structure and requirements in the project, you might want to place the packages in a different location.



The location where to store the packages (among other options) can be changed using <u>nuget.config</u><sup>23</sup> (Figure 5) and placed at a suitable location in the folder hierarchical structure.

Figure 5- Nuget.config file to configure different repository path

When you add a package to the project, NuGet will add it in a folder named following the pattern "PackageName.Version."

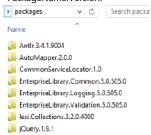


Figure 69– "Packages" folder and its contempendency Management Pattern with NuGet Package Restore

In this pattern, you will not add packages to version control. Instead, each development environment downloads packages from the repository according to the packages.config that you define per project in a solution. Starting with v2.7 NuGet enables Package Restore in Visual Studio by default. By default, you give consent to download automatically missing packages (Tools -> Options -> Package Manager - Figure 8).

Automatic Package Restore<sup>24</sup> will restore missing packages before building any MSBuild projects. This makes sure that NuGet had downloaded all the necessary items (including build targets/properties that are included in the nuget package) and placed them in the correct location. NuGet does this by hooking into Visual Studio build events.

For Team Build 2013 and Visual Studio Online NuGet Package Restore workflow works by default, however for previous versions of <u>Team Build some customization is necessary to integrate using the NuGet Command Line Package Restore</u><sup>25</sup>.

Please note that if you have enabled NuGet Package Restore on solution level before v2.7, then we recommended migrating it to latest "Automatic Package Restore<sup>26</sup>."

http://docs.nuget.org/docs/workflows/migrating-to-automatic-package-restore



Page 13 of 22

<sup>&</sup>lt;sup>23</sup> http://docs.nugetorg/docs/reference/nuget-config-file#NuGet\_config\_extensibility\_point

<sup>24</sup> http://docs.nugetorg/docs/reference/package-restore

<sup>&</sup>lt;sup>25</sup> http://docs.nuget.org/docs/reference/package-restore-with-team-build

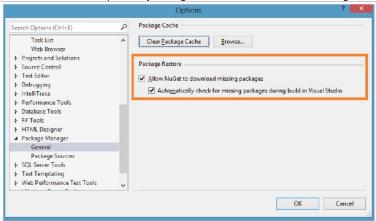


Figure 7 Visual Studio Package Manager Package Restore default settings

### Getting NuGet packages with package restore

Add and update packages using NuGet package manager in the same way as you did before.

Do not commit the binaries into source control.

As before, only one team member needs maintain the packages configurations for projects, and check them into version control (and you might want to restrict write access to the file). When any team member presses F5, the first time, NuGet will automatically download the packages. This will also happen in the build server.

### Pros and Cons regarding NuGet package restore

Like with any feature, there are always advantages and disadvantages to using it and the same applies to NuGet package restore. Keep in mind that although NuGet package restore depends on a network connection to a NuGet feed, NuGet can rely on package cache scenarios to work offline. Also in today's development, environments the network feed to NuGet may be mostly reliable.

### Using NuGet Package restore

### Advantages

- You will use less storage space in TFS Source Control because no you do not need to store binaries in the repository.
- Less data in TFS means less to back up for TFS DR scenarios
- Get operations may be optimized since you are pulling less data over the wire

### Disadvantages

- Developers can be blocked if any of the NuGet repositories is unavailable after a package update.
- TFS Build would fail or be postponed if a NuGet repository is unavailable (especially when the setting to clean workspace before build is turned on) if no cache version of the package was available.

 $<sup>^{27}\</sup> http://docs.nugetorg/docs/workflows/using-nuget-without-committing-packages$ 



Page 14 of 22

### Additional references

- How to use NuGet with TFS Version Control <sup>28</sup>
- Introducing NuGet Package Management for .NET <sup>29</sup>
- Manage Project Libraries with NuGet <sup>30</sup>
- NuGet Documentation 31
- NuGet FAQ 32
- NuGet Gallery 33
- NuGet In Depth: Empowering Open Source on the .NET Platform <sup>34</sup>
- NuGet Videos 35
- Packages in Visual Studio Templates 36
- NuGet Package Restore with Team Foundation Build <sup>37</sup>

<sup>37</sup> http://docs.nuget.org/docs/reference/package-restore-with-team-build



<sup>28</sup> http://loicbaumann.fr/en/2012/02/07/how-to-use-nuget-with-the-tfs-version-control/

<sup>29</sup> http://www.hanselman.com/blog/IntroducingNuGetPackageManagementForNETAnotherPieceOfTheWebStack.aspx

<sup>30</sup> http://msdn.microsoft.com/en-us/magazine/hh547106.aspx

http://docs.nugetorg/docs/start-here/nuget-faq
 http://www.nuget.org/
 http://channel9msdn.com/Events/MIX/MIX11/FRM0

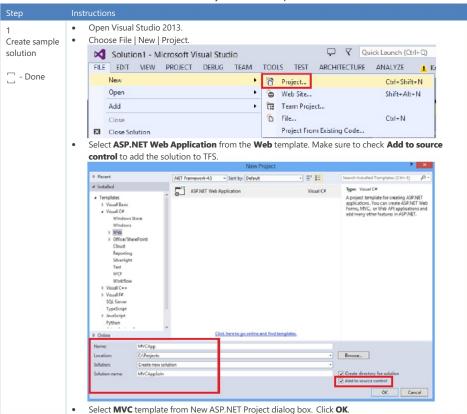
<sup>35</sup> http://docs.nuget.org/docs/start-here/videos

<sup>36</sup> http://docs.nugetorg/docs/reference/packages-in-visual-studio-templates

# Hands-on Lab (HOL) – Using NuGet

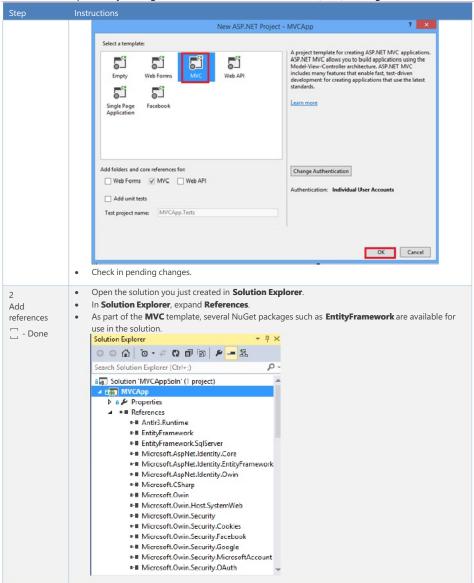
### Exercise 1 – Find and install a NuGet Package

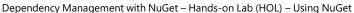
NuGet is a Visual Studio Extension that makes it easy to install and update libraries and tools in Visual Studio.

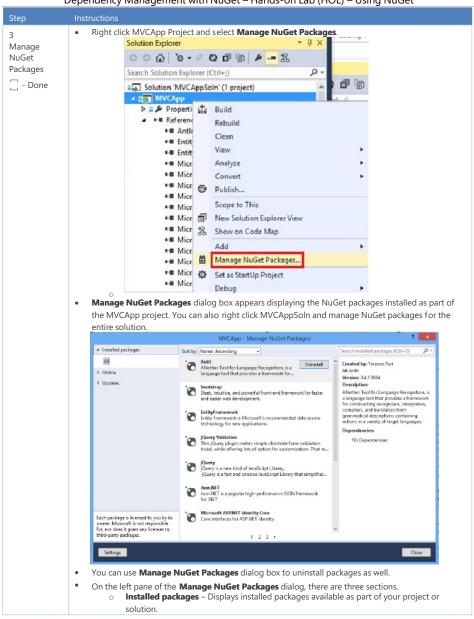




Dependency Management with NuGet – Hands-on Lab (HOL) – Using NuGet

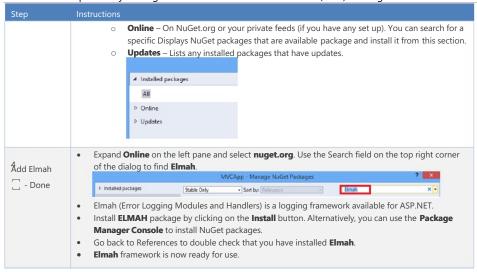






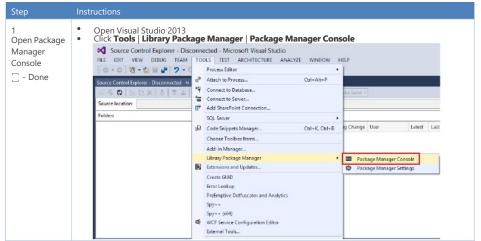


Dependency Management with NuGet - Hands-on Lab (HOL) - Using NuGet

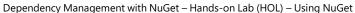


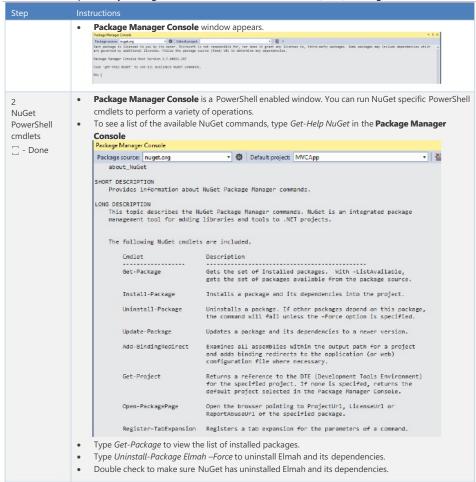
### Exercise 2 – Using Package Manager Console

**Package Manager Console** is a PowerShell enabled window in Visual Studio. PowerShell commands allow you to install packages when you do not have the solution open. You will also need to use it for packages that create commands that require PowerShell.







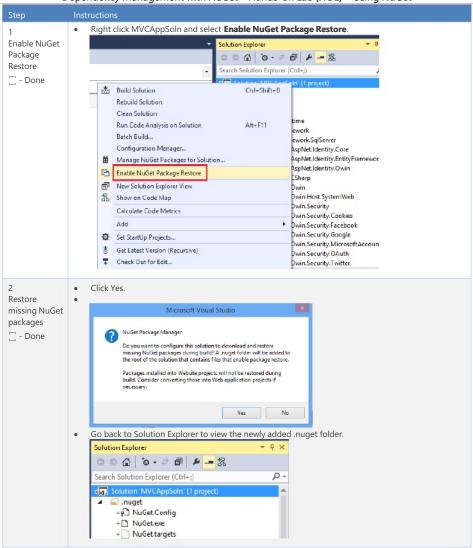


### Exercise 3 – NuGet Package Manager

NuGet has package restoration that on building is capable of downloading all packages that have been installed and configured for the project, but cannot be found from local disk. This does not change the fact that NuGet needs to install packages for the project and installs updates the same way. NuGet enables Package Restore at the solution level.



Dependency Management with NuGet – Hands-on Lab (HOL) – Using NuGet



## In Conclusion

This concludes our adventure into using NuGet. We have touched on basic theory of NuGet, managing shared resources, dependency management, and concluded with a practical Hands-on lab (HOL) exploring NuGet. We hope that you have found this guide useful.

Sincerely

**The Microsoft Visual Studio ALM Rangers** 



