

2012

ALM Rangers Hands-on Lab

IntelliTrace - Symbols Configuration & Build

April 01, 2013

This content was created by the Visual Studio ALM Rangers, a special group with members from the Visual Studio Product Team, Microsoft Services, Microsoft Most Valuable Professionals (MVPs) and Visual Studio Community Leads.

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, it should not be interpreted 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.

* 2012 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.

Table of Contents

[Introduction 4](#_Toc353274780)

[Visual Studio ALM Rangers 4](#_Toc353274781)

[Contributors 4](#_Toc353274782)

[Reviewers 4](#_Toc353274783)

[Prerequisites 4](#_Toc353274784)

[Exercise 1 – Configure the build controller to build Tailspin Toys 5](#_Toc353274785)

[Exercise 2 – Create a Symbol Store file share 7](#_Toc353274786)

[Exercise 3 – Configure a build definition to Index sources and publish symbols 9](#_Toc353274787)

[Exercise 4 – Run the build 12](#_Toc353274788)

[Appendix 14](#_Toc353274789)

[Other ALM Rangers Resources 14](#_Toc353274790)

[Code Reference 14](#_Toc353274791)

[Figures Reference 14](#_Toc353274792)

[Tables Reference 14](#_Toc353274793)

# Introduction

This hands-on lab will show you how to configure the TFS build service to emit source and symbol information, which later can be used with IntelliTrace to perform post-run debugging.

The lab will take approximately 15 minutes to complete.

The hands-on lab is part of a series of labs that will give you a complete end-to-end scenario of using IntelliTrace in production. The series is structured as follows:

1. Learn how to configure TFS build to generate source and symbols information (this lab).
2. Learn now to configure the stand-alone IntelliTrace collector.
3. Learn how to setup Visual Studio to locate symbols when analyzing an IntelliTrace log.

## Visual Studio ALM Rangers

The Visual Studio ALM Rangers are a special group with members from the Visual Studio Product group, Microsoft Services, Microsoft Most Valuable Professionals (MVP) and Visual Studio Community Leads. Their mission is to provide out-of-band solutions to missing features and guidance. A growing Rangers Index is available online[[1]](#footnote-2).

## Contributors

Mathias Olausson

## Reviewers

Jesse Houwing

Anna Galaeva

Giulio Vian

Larry Guger

## Prerequisites

To complete these hands-on-lab walk-through scenarios you need the following environment:

* The “Brian Keller TFS 2012 Demo Virtual Machine”[[2]](#footnote-3).

# Exercise 1 – Configure the build controller to build Tailspin Toys

|  |  |
| --- | --- |
| Objective | In TFS 2012 a build controller is associated with one specific project collection. In this first exercise you will change the association to use the TailspinToysCollection. |

1. Start the virtual machine.
2. Log in to the Virtual Machine as **Adam Barr** using the standard password **P2ssw0rd**.
3. Open the Team Foundation Server administration console from the Start menu and go to the **Build Configuration** section.
4. Stop the build service and click on **properties**.

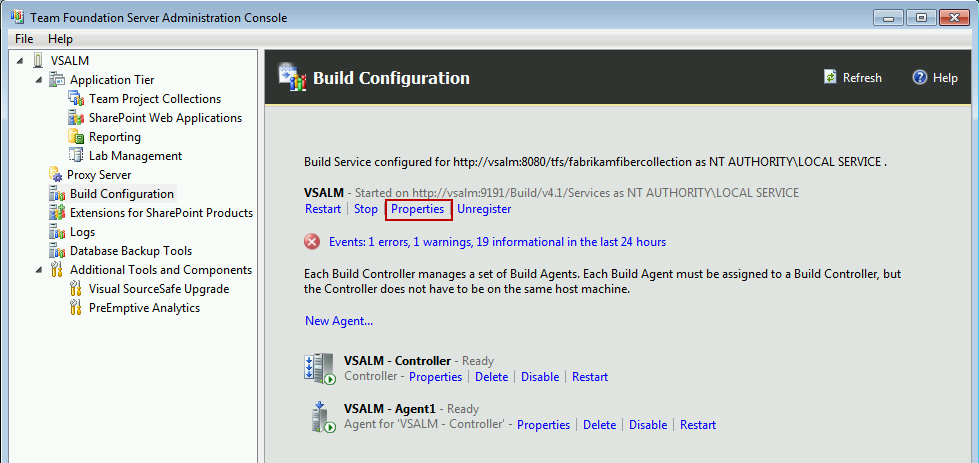


Figure – Set build service properties.

1. Change the collection to the **TailspinToysCollection**.

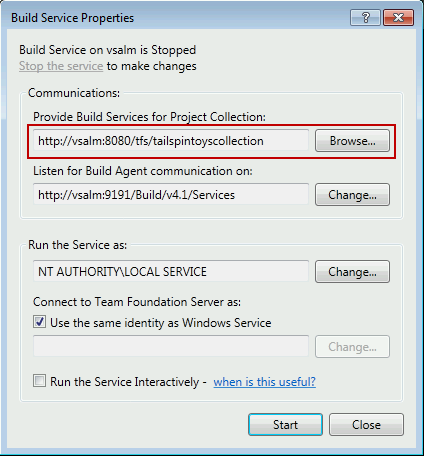


Figure – Configure build service to TailspinToysCollection.

1. Save the changes.
2. Start the build service again and make sure the default controller and agent goes to a **Ready** state.

|  |  |
| --- | --- |
| Review | In this initial exercise you have now seen how a build controller is associated with only one team project collection. In order to use the demo virtual machine with our sample project we needed to change this association to use the TailspinToys collection instead. |

# Exercise 2 – Create a Symbol Store file share

|  |  |
| --- | --- |
| Objective | In this exercise you will create a file share use to store the source and symbol data create by the build process. The symbol information can be published together with the build result but since the use-cases for build output and symbol information is different it often makes more sense to handle them separately. |

1. Create a file share for the symbol store.
   1. Open a Windows Explorer.
   2. Create new folder at **C:\Symbols**.
   3. Share the folder as **Symbols**, we will later refer to it using the UNC path **\\vsalm\symbols**.
   4. Set the appropriate permissions on the file share using the advanced sharing properties of the folder (folder properties🡪Sharing🡪Advanced Sharing). Make sure both the TFS Build service account and the users of IntelliTrace have access to the share as described in the following steps. In this lab Everyone should have read access and the TFS build service is running as LOCAL SERVICE.
      1. The build account needs to have write access to the share.
      2. IntelliTrace users require only read access.

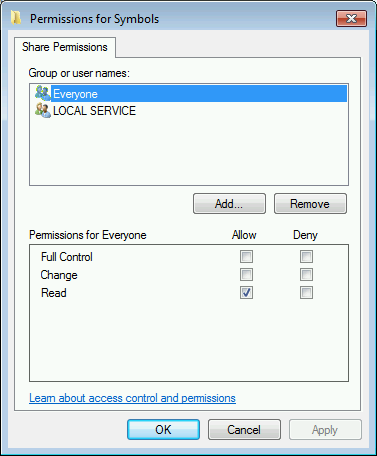


Figure - Assign file share permissions.

* 1. Set the corresponding permissions on the folder.

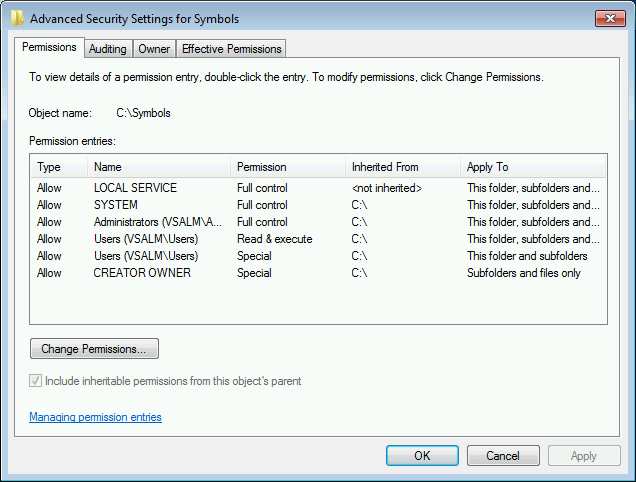


Figure - Assign folder permissions.

**Note:** In a production environment the symbols share is going to be used by multiple people over time. It’s a good idea to create the symbols share on a common file server or create a DNS entry to make the share location machine independent.

|  |  |
| --- | --- |
| Review | In this initial exercise you have now created the file share to store symbol information for later reference by the IntelliTrace debugger. |

# Exercise 3 – Configure a build definition to Index sources and publish symbols

|  |  |
| --- | --- |
| Objective | In this exercise you create and configure a build definition to index sources and publish the symbols to a file share. The built-in build workflow contains the logic to do this for us, which makes it very easy to get started. |

1. Start Visual Studio 2012 and connect to the **Tailspin Toys** team project in the **TailspinToysCollection**.

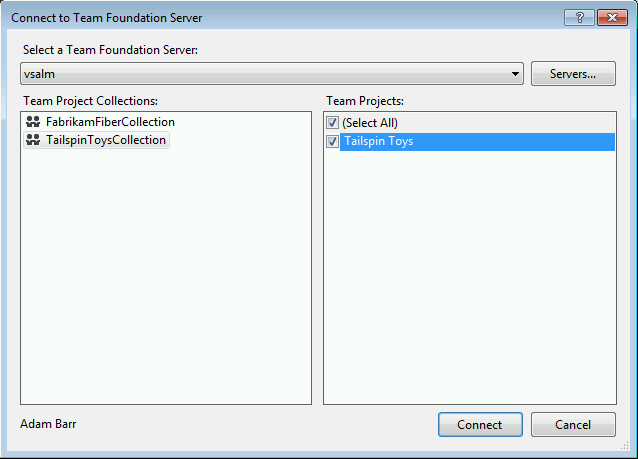


Figure - Connect to the Tailspin Toys team project.

1. Open the Team Explorer and click on the Builds node.
2. Click the **“New Build Definition”** link.
3. Configure the default build process.

|  |  |
| --- | --- |
| General | Name the build definition **Tailspin Toys – Main**. |
| Trigger | Use the default trigger **Manual**.  Note: there are known issues with gated check-in builds and source indexing, see: <http://social.msdn.microsoft.com/Forums/en-US/tfsbuild/thread/d13b0d3a-2b1e-487e-b0d8-aa3de8e3129d> for more information. |
| Workspace | Make sure only the Main branch is mapped to the build workspace. If you have more than the one mapping below make sure you remove the extra lines. |
| Build Defaults | Use the **Default Controller – vsalm** as the build controller.  Use the existing drop folder [**\\vsalm\ffdrops**](file:///\\vsalm\ffdrops) as the staging location. |
| Process | Add the $/Tailspin Toys/Main/TailspinToys.sln solution to the build process and configure to build for Release, Any CPU.    Configure the “Source and Symbol Server Settings” in the Basic settings by setting “Index Sources” to **True** and set the “Path to Publish Symbols” to [**\\vsalm\Symbols**](file:///\\vsalm\Symbols) as created earlier. |
| Retention Policies | Use the default settings.  **Note:** The default settings will remove most of the build information once the retention policy kicks in. For builds in use where you expect the symbol information to be available you should either configure the retention policy to keep symbol information using the settings for each retention policy:    Or prevent the build from being removed by a retention policy by marking the build as “**Retain Indefinately**” (note that you must select an existing build to enable the menu option). |

Table - Build Definition properties.

1. Save the build definition. You should now see the new build definition appear in the Team Explorer.

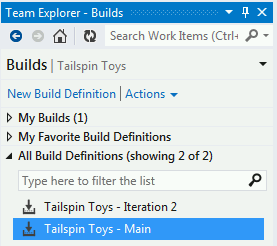


Figure - New build definition.

**Note:** the default behavior in the TFS 2012 build workflow can cause symbol data to be corrupted because concurrent builds are not synchronized when it comes to publishing the symbol data. As long as build producing indexed result are not run concurrently this is not a problem. See <http://msdn.microsoft.com/en-us/library/gg265783.aspx#symbols> for more information about this scenario and how to work around it.

For more information about how publishing symbol data works, see <http://msdn.microsoft.com/en-us/library/hh190722.aspx>.

|  |  |
| --- | --- |
| Review | You have now configured TFS Build to emit symbol information, which later can be used by users who needs to resolve symbols while debugging. |

# Exercise 4 – Run the build

|  |  |
| --- | --- |
| Objective | Next you will run newly defined build and examine the output from the source and symbol indexing. |

1. Locate the Tailspin Toys – Main build definition in the Team Explorer.
2. Right-click and select “Queue new Build”. Use the default settings and click “Queue”.
3. To view the build log, click on the started build in the Team Explorer.



Figure - View build log.

1. Wait for the build to complete.
2. Click “View Log” and scroll down to the Index Source section to confirm the step has been run.

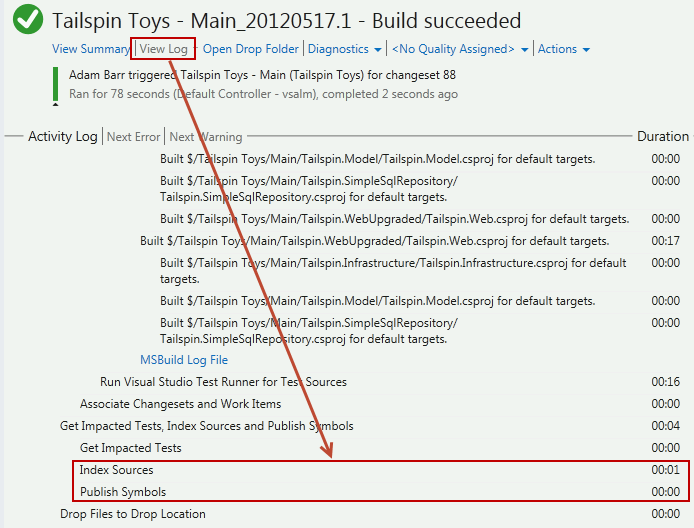


Figure - Index Sources in build log.

1. Open a Windows Explorer and go to **c:\Symbols**. Note how the folder has been updated with the published source and symbol data.

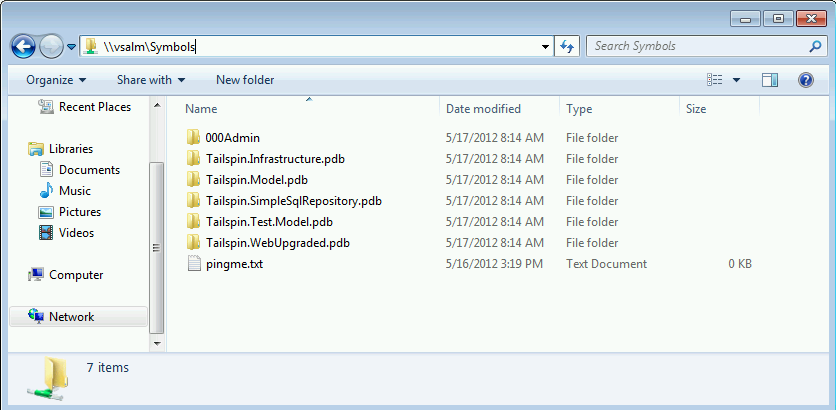


Figure - Symbols share content after build.

|  |  |
| --- | --- |
| Review | You have now configured and run a TFS Build to publish source and symbol data. |

# Appendix

## Other ALM Rangers Resources

Understanding the ALM Rangers – <http://aka.ms/vsarunderstand>

Visual Studio ALM Rangers Home Page – <http://aka.ms/vsarmsdn>

Visual Studio ALM Ranger Solutions – <http://aka.ms/vsarsolutions>

## Code Reference

**No table of figures entries found.**

## Figures Reference

[Figure 1 – Set build service properties. 5](#_Toc353274794)

[Figure 2 – Configure build service to TailspinToysCollection. 6](#_Toc353274795)

[Figure 3 - Assign file share permissions. 7](#_Toc353274796)

[Figure 4 - Assign folder permissions. 8](#_Toc353274797)

[Figure 5 - Connect to the Tailspin Toys team project. 9](#_Toc353274798)

[Figure 6 - New build definition. 11](#_Toc353274799)

[Figure 7 - View build log. 12](#_Toc353274800)

[Figure 8 - Index Sources in build log. 12](#_Toc353274801)

[Figure 9 - Symbols share content after build. 13](#_Toc353274802)

## Tables Reference

[Table 1 - Build Definition properties. 11](#_Toc353274803)

1. <http://aka.ms/vsarindex> [↑](#footnote-ref-2)
2. <http://blogs.msdn.com/b/briankel/archive/2011/09/16/visual-studio-11-application-lifecycle-management-virtual-machine-and-hands-on-labs-demo-scripts.aspx> [↑](#footnote-ref-3)