

2012

ALM Rangers Hands-on Lab

IntelliTrace - Collection for Rich Client (WPF)

April 01, 2013

a

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](#_Toc353274738)

[Visual Studio ALM Rangers 4](#_Toc353274739)

[Contributors 4](#_Toc353274740)

[Reviewers 4](#_Toc353274741)

[Prerequisites 4](#_Toc353274742)

[Exercise 1 – Installing the stand-alone IntelliTrace data collector 5](#_Toc353274743)

[Exercise 2 – Use Visual Studio to create a collection plan for a WPF application 7](#_Toc353274744)

[Exercise 3 – Test the collection profile from a local IntelliTrace session 8](#_Toc353274745)

[Appendix 11](#_Toc353274746)

[Other ALM Rangers Resources 11](#_Toc353274747)

[Code Reference 11](#_Toc353274748)

[Figures Reference 11](#_Toc353274749)

[Tables Reference 11](#_Toc353274750)

# Introduction

This hands-on lab will show you how to setup an IntelliTrace collection profile for a Windows Presentation Foundation (WPF) application.

The lab will take approximately 20 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.
2. Learn now to configure the stand-alone IntelliTrace collector (this lab).
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

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 – Installing the stand-alone IntelliTrace data collector

|  |  |
| --- | --- |
| Objective | In this first exercise you will learn how to deploy the stand-alone IntelliTrace data collector. Knowing how to install the data collector will enable you to quickly start capturing IntelliTrace logs in environment where Visual Studio is not (or cannot) be installed. You will simulate installing the data collector in a production environment by copying the installation file from a Visual Studio installation. |

1. Start the virtual machine.
2. Log in to the Virtual Machine as **Adam Barr** using the standard password **P2ssw0rd**.
3. Create a target folder to hold the IntelliTrace data collector binaries.
   1. Create a new folder called **c:\iTrace**.
4. Create a folder to hold IntelliTrace log files.
   1. Create a new folder called **c:\iTraceLogs**.
5. Locate the stand-alone IntelliTrace data collector under the Visual Studio 2012 installation.
   1. Open a Windows Explorer and navigate to “**C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\CommonExtensions\Microsoft\IntelliTrace\11.0.0”**.

**Note:** in a production scenario you will download just the stand-alone collector to the production environment,

1. Copy the file **IntelliTraceCollection.cab** to **c:\iTrace**.
2. Extract the IntelliTrace cabinet using PowerShell.
   1. Start a Windows PowerShell command console by typing Windows PowerShell in the Windows “Search programs and files” box.

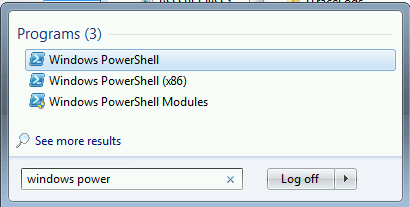


Figure - Start Windows PowerShell

* 1. Set the working folder to c:\iTrace with the following command in the PowerShell console:

CD c:\iTrace

* 1. Extract the content in the IntelliTrace cabinet with the following command:

Expand IntelliTraceCollection.cab -F:\* .

|  |  |
| --- | --- |
| Review | You have now successfully installed the stand-alone IntelliTrace data collector. |

# Exercise 2 – Use Visual Studio to create a collection plan for a WPF application

|  |  |
| --- | --- |
| Objective | In this exercise you will create a collection plan to collect relevant IntelliTrace events for a Windows Presentation Foundation application. IntelliTrace data collection is very flexible but it comes with a price, so to avoid excessive logging (which will affect performance as well as storage) you should create different collection plans to use for different scenarios. |

1. Open a Windows Explorer and navigate to the c:\iTrace folder.
2. By default the stand-alone IntelliTrace data collection contains two collection plans.

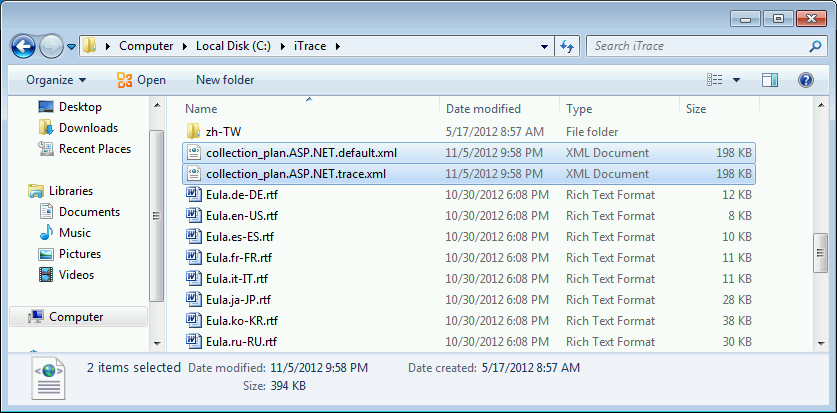


Figure - Default IntelliTrace collection plans.

1. Copy the collection\_plan.ASP.NET.default.xml to collection\_plan.WPF.xml.
2. Open the collection\_plan.WPF.xml file in Visual Studio and search for WPF to find the IntelliTrace events specific to Windows Presentation Foundation.



Figure - WPF events in the IntelliTrace collection plan.

1. Enable all WPF events by setting the <DiagnosticEventSpecification enabled="**true**"> on each event.
   1. Note that the **enabled** attribute may be missing in which case you will need to add it.
2. Save the collection\_plan.WPF.xml.

|  |  |
| --- | --- |
| Review | In this exercise you have learnt how to create a customized IntelliTrace data collection plan. |

# Exercise 3 – Test the collection profile from a local IntelliTrace session

|  |  |
| --- | --- |
| Objective | In this exercise you will use Visual Studio to test the collection profile created in the previous exercise. We will use an existing build to run the IntelliTrace collection using the collection plan from the previous step. |

1. Open a Windows command prompt and type the following command to start the stand-alone IntelliTrace collector:

c:\iTrace\IntellitraceSC.exe launch /cp:c:\iTrace\collection\_plan.WPF.xml /f:c:\iTraceLogs\TailspinToysAdmin.iTrace "C:\drops\Tailspin Toys - Iteration 2\Tailspin Toys - Iteration 2\_20100318.6\Tailspin.Admin.App.exe"

1. The IntelliTrace collector will initiate the data collection and launch the WPF application from C:\drops\Tailspin Toys - Iteration 2\Tailspin Toys - Iteration 2\_20100318.6\Tailspin.Admin.App.exe.
2. Click through the user interface using the mouse and keyboard. Navigating with the up/down arrows will eventually cause the application to fail:

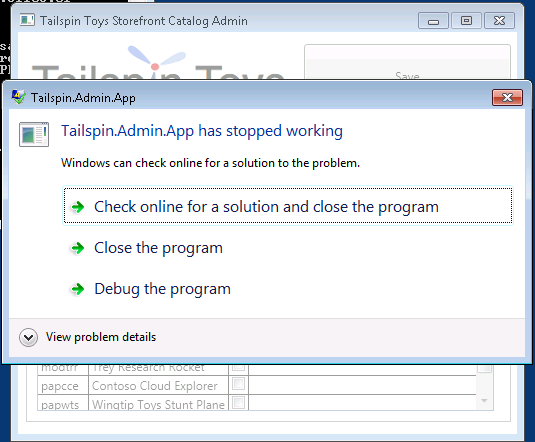


Figure - Exception from the Tailspin Toys admin client.

1. Close the Tailspin Toys admin application.
2. Note how an IntelliTrace logfile has been created in **c:\iTraceLogs**.

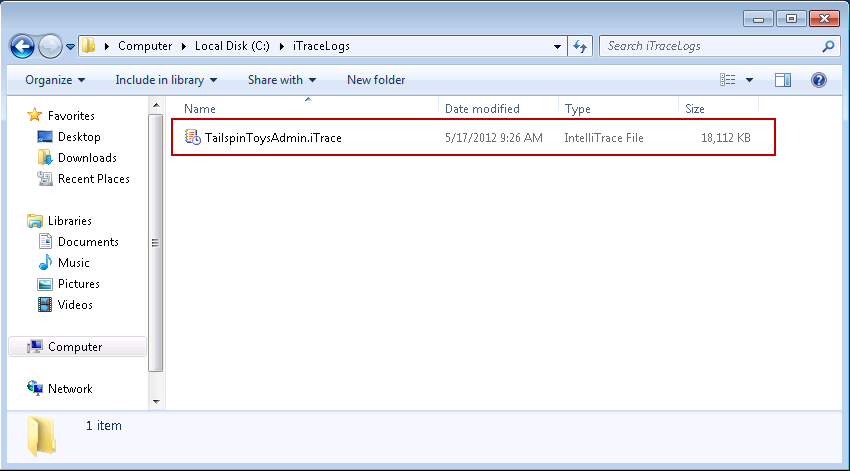


Figure - IntelliTrace log file.

1. Double-click on the TailspinToysAdmin.iTrace file. The file will now be loaded in Visual Studio.
2. Double-click on the exception to load the IntelliTrace log in the debugger. Ignore any messages regarding failures to load symbols, is covered in the “5024 - No Symbols Found Resolution (Client side)” hands-on lab.

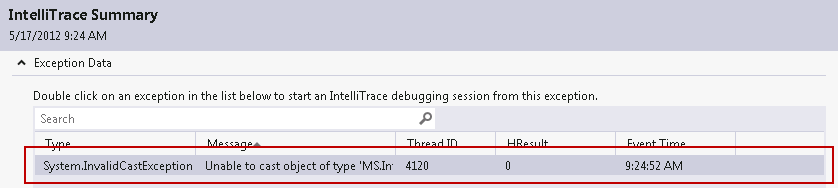


Figure - IntelliTrace log with Exception.

1. Filter the IntelliTrace events to only show Exceptions and Gestures:

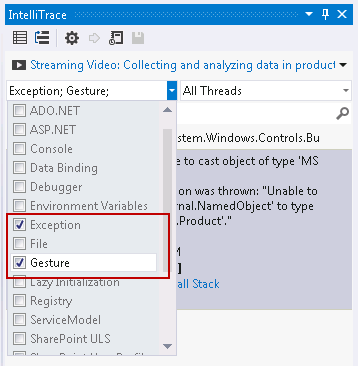


Figure - Filter IntelliTrace events.

1. Notice how the IntelliTrace log now only displays the selected event types.

|  |  |
| --- | --- |
| Review | You have learnt how to use the stand-alone IntelliTrace data collector with a custom collection plan to capture IntelliTrace information that is relevant to your application. |

# 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 - Start Windows PowerShell 5](#_Toc353274751)

[Figure 2 - Default IntelliTrace collection plans. 7](#_Toc353274752)

[Figure 3 - WPF events in the IntelliTrace collection plan. 7](#_Toc353274753)

[Figure 4 - Exception from the Tailspin Toys admin client. 8](#_Toc353274754)

[Figure 5 - IntelliTrace log file. 9](#_Toc353274755)

[Figure 6 - IntelliTrace log with Exception. 9](#_Toc353274756)

[Figure 7 - Filter IntelliTrace events. 9](#_Toc353274757)

## Tables Reference

**No table of figures entries found.**

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)