|  |
| --- |
| **C:\Users\fez\Downloads\VS2012.ALMRangers.Logo.NoTrademark.Transparent.Purple.1416x475.png** |
| **Microsoft Test Manager Visual Studio 2012** |
| **Test Impact Analysis**  **Hands-On Lab (HOL)** |
|  |
| Wednesday, July 18, 2012 |
| **Visual Studio ALM Rangers**  Richard Fennell, Richard Albrecht |
| **Microsoft Corporation** |

|  |
| --- |
| **Visual Studio ALM Rangers**  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 Valued 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 5](#_Toc322683109)

[Overview 5](#_Toc322683110)

[What is Test Impact Analysis 5](#_Toc322683111)

[Visual Studio ALM Rangers 5](#_Toc322683112)

[Prerequisites 5](#_Toc322683113)

[Paths 6](#_Toc322683114)

[Suggested Focus and Time Map 7](#_Toc322683115)

[Exercise 1 - Using an Automated Build with Test Impact Analysis for Manual Tests 8](#_Toc322683116)

[Task 1 - Create a new WinForm Application to test 8](#_Toc322683117)

[Task 2 – Create an automated build 12](#_Toc322683118)

[Task 3 – Creating the Manual Tests 17](#_Toc322683119)

[Task 4 – Setting up the gathering the Test Impact for manual test runs 20](#_Toc322683120)

[Task 5 - Gathering the baseline Manual Test data 23](#_Toc322683121)

[Task 6 –Impacting the Manual Test 26](#_Toc322683122)

[Task 7 –Viewing Impacted Tests in MTM 28](#_Toc322683123)

[References 31](#_Toc322683124)

[Visual Studio ALM Rangers Site 31](#_Toc322683125)

**Table of Figures**

[Figure 1 - Creating a new project 8](#_Toc330321621)

[Figure 2 - Adding the project to source control 9](#_Toc330321622)

[Figure 3 - The sample form 10](#_Toc330321623)

[Figure 4 - Check in files 11](#_Toc330321624)

[Figure 5 - Create a new build 12](#_Toc330321625)

[Figure 6 - Setting the trigger 13](#_Toc330321626)

[Figure 7 - Workspace settings 13](#_Toc330321627)

[Figure 8 - Creating a drops folder share 14](#_Toc330321628)

[Figure 9 - Setting the drop location 14](#_Toc330321629)

[Figure 10 - Enable Test Impact 15](#_Toc330321630)

[Figure 11 - Queue a build 16](#_Toc330321631)

[Figure 12 - Completed build 16](#_Toc330321632)

[Figure 13 - Select Home Button 17](#_Toc330321633)

[Figure 14 - Change Project 17](#_Toc330321634)

[Figure 15 - Connect to team project 18](#_Toc330321635)

[Figure 16 - Create a test plan 18](#_Toc330321636)

[Figure 17 - Add a new test 19](#_Toc330321637)

[Figure 18 - Create a new test 20](#_Toc330321638)

[Figure 19 - Test plan properties 21](#_Toc330321639)

[Figure 20 - Enabling test impact 21](#_Toc330321640)

[Figure 21 - Configure Diagnostic Data Adapter - Modules 22](#_Toc330321641)

[Figure 22 - Configure Diagnostic Data Adapter - Advanced Option 22](#_Toc330321642)

[Figure 23 - Enable ASP.NET Client Proxy for IntelliTrace and Test Impact 23](#_Toc330321643)

[Figure 24 - Run tests 24](#_Toc330321644)

[Figure 25 - Selecting a build 25](#_Toc330321645)

[Figure 26 - Starting a manual test 25](#_Toc330321646)

[Figure 27 - Running a manual test 26](#_Toc330321647)

[Figure 28 - Tests passed 26](#_Toc330321648)

[Figure 29 - Build in progress 27](#_Toc330321649)

[Figure 30 - Completed build showing impacted tests 28](#_Toc330321650)

[Figure 31 - Recommended tests tab 29](#_Toc330321651)

[Figure 32 - Selecting a build 29](#_Toc330321652)

[Figure 33 - Recommended Tests 30](#_Toc330321653)

[Figure 34 - Resetting active flag 30](#_Toc330321654)

[Figure 35 - User prompted to re-run a test 31](#_Toc330321655)

**Table of Tables**

[Table 1 – Recommend Software and Hardware for Hands-On-Lab 6](#_Toc321140418)

[Table 2 - Suggested focus and time map 7](#_Toc321140419)

# Introduction

## Overview

|  |  |
| --- | --- |
| NoteNote | **Note** |
| This Hands-on Lab was initially created for Visual Studio 11 Beta. If you are using a newer version of Visual Studio, you may need to follow slightly different steps to those prescribed. | |

### What is Test Impact Analysis

Though it is preferable to run all test available for a new build of a solution, running a full suite of tests, whether they be manual or automated, can be time consuming.

To provide an alternative to running the entire test suite or having to decide which tests to run yourself, since Visual Studio 2010, Test Impact Analysis is part of the Visual Studio 2012 Premium and Ultimate SKU.

Test Impact Analysis is a related concept to test code coverage. Test coverage tools tell you what percentage of the code base is covered by tests. By knowing which tests cover which blocks of production code, it is possible to say which tests need to be run if a given block of production code is edited. This technology can be applied to manual tests, as well as to unit tests, as long as suitable diagnostic data is gathered when a manual test is first run.

WARNING

A really common mistake is to try to make use of test impact analysis before a test has been successfully run at least one. Visual Studio/TFS cannot advise of test impact without the data stored from a successful test run. This applies equally to both manual and unit tests.

Remember also that test impact data is not collected when you file a bug, irrespective of whether the test is marked as passed or failed.

## 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 Valued Professionals (MVP) and Visual Studio Community Leads. Their mission is to provide out-of-band solutions to missing features and guidance.

This guide is intended for Microsoft “200-300 level” users of Team Foundation Server. They are intermediate to advanced users of Team Foundation Server and have in-depth understanding of the product features in a real-world environment. Parts of this guide might be useful to novices and experts, but they are not the intended audience for this guide.

## Prerequisites

To complete the basic and advanced hands-on-lab walk-through scenarios you need the following environment:

* Visual Studio ALM Rangers Base Virtual Image

… or

* A single server (physical or virtual) environment that has the following software installed and configured:

| Software | Version | Note |
| --- | --- | --- |
| Operating System | Microsoft Windows 7  Microsoft Windows Server 2008 Microsoft Windows Server 2008 R2 |  |
| IIS | IIS 7.0 IIS 7.5 | Shipped as part of operating system. |
| SQL Server | Microsoft SQL Server 2008 SP2  Microsoft SQL Server 2008 R2  Microsoft SQL Server 2012 | Enterprise edition recommended. |
| Microsoft Visual Studio Team Foundation Server | Team Foundation Server 2012 | Configuration: Basic or Standard |
| Microsoft Visual Studio Ultimate | Visual Studio 2012 |  |
| Microsoft Visual Studio Team Build Services | Team Foundation Server 2012 | 1 Build Controller + 1 Build Agent |

Table – Recommend Software and Hardware for Hands-On-Lab

## Paths

The HOL refers to the working directory as **<HOL\_PATH>** in this document, which by default refers to the following path on Rangers VMs: **C:\HOL\MTM**

## Suggested Focus and Time Map

If you are intending to follow the Hands-on Lab (HOL) step by step, use these times as a guideline. If, however, you are intending to investigate each step in detail to double the times as a bare minimum.

|  |  |  |
| --- | --- | --- |
| Topic | Duration in minutes | Page |
| Task 1 - Create a new WinForm Application to test | 10 | 8 |
| Task 2 – Create an automated build | 10 | 12 |
| Task 3 – Creating the Manual Tests | 15 | 17 |
| Task 4 – Setting up the gathering the Test Impact for manual test runs | 5 | 20 |
| Task 5 – Gathering the baseline Manual Test data | 10 | 23 |
| Task 6 – Impacting the Manual Test | 5 | 26 |
| Task 7 – Viewing Impacted Tests in MTM | 5 | 28 |
| TOTAL | **60 min** |  |

Table – Suggested focus and time map

*We wish you a pleasant and interesting journey!*

# Exercise 1 - Using an Automated Build with Test Impact Analysis for Manual Tests

HinweisOBJECTIVE

In this exercise, we show how an automated build can return a list of manual tests that are impacted by a change in the production code base

### Task 1 - Create a new WinForm Application to test

First we need to create a simple application to run our tests against, and place this code under source control in TFS.

1. Logon as **Administrator** using password: **P2ssw0rd**
2. Start Visual Studio
3. If this is the first time launching Visual Studio for this user, you will be prompted to **Choose Default Environment Settings.** Choose **Visual C# Development Settings** and click the **Start Visual Studio** button
4. Make sure you are connected to team project on your TFS server and the Fabrikam Fiber Team Project[[1]](#footnote-2).
5. **Create** a simple **Windows Forms Application** (**click** “**File”** in the menu | **select “New”** | **select “Project”**).
6. **Add** the solution to **source control** (**click** on **“Windows Forms Application”** within the Visual C# area | **enter** the **name “TestImpact”** | **enter** the **location “C:\HOL\MTM” | enter** the **solution name “TestImpact”** |select checkbox **“Add to source control”** | **click** button **“OK”**):

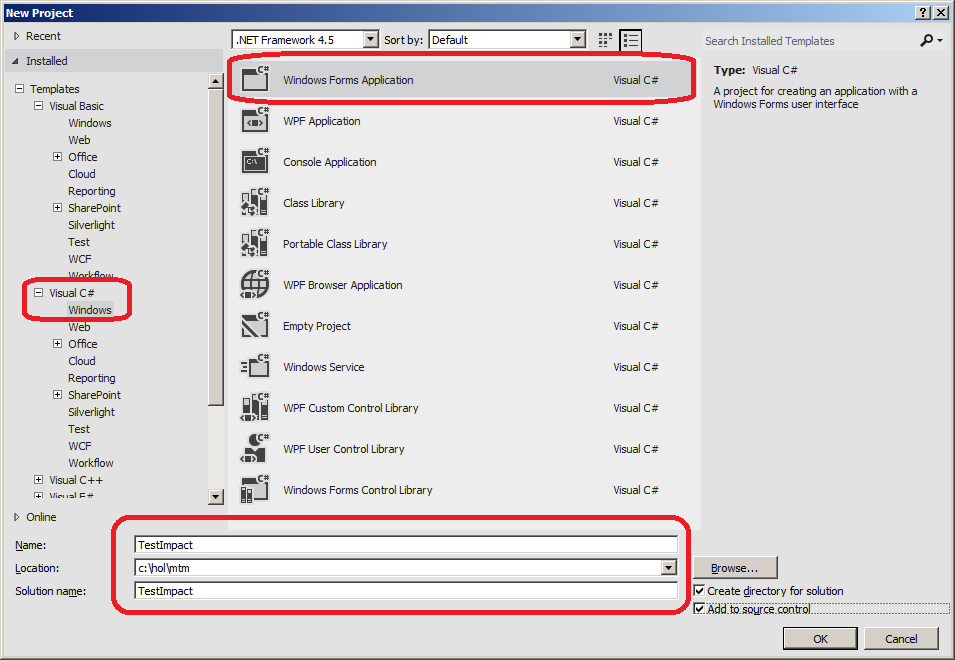


Figure - Creating a new project

1. As add solution to source control has been selected you will be prompted to provide a location in TFS.Add the solution to the Source Control into a **new folder** called **“TestImpact”** under the “**Dev”** branch in the existing Team Project **“FabrikamFiber”**

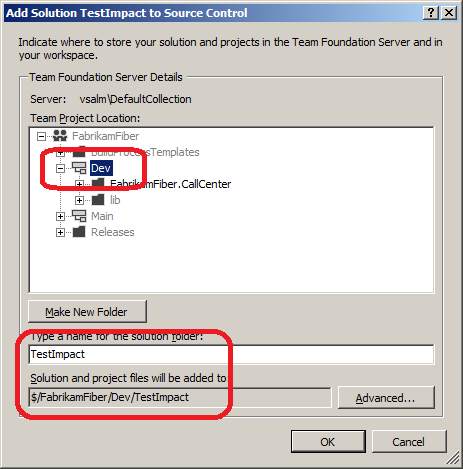


Figure - Adding the project to source control

1. Once this is complete you will be able to edit the project.
2. **Form1** should be open in the IDE
3. From the Visual Studio toolbox add two buttons and a label to the form

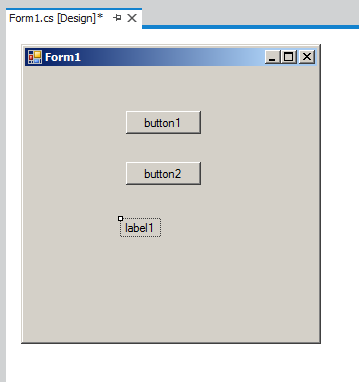


Figure - The sample form

1. Select the **button1** and add the following code to its click event

private void button1\_Click(object sender, EventArgs e)

{

this.label1.Text = "Button 1 pressed";

}

1. Repeat this process for **button2**, making the new text “Button 2 pressed”
2. Build the project in Visual Studio to make sure you have made no typing errors
3. Check the edited code into TFS by opening the **“Pending Changes”** hub in Team Explorer[[2]](#footnote-3). Check that all your projects files are listed in the pending changes and press **“Check In”** to check them in.

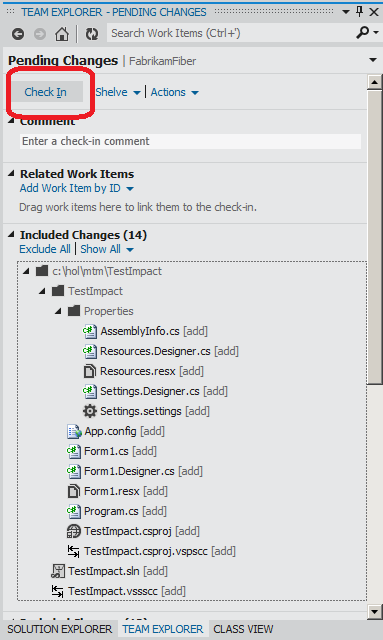


Figure - Check in files

### Task 2 – Create an automated build

An automated build must be created to run the test impact analysis.

1. In Visual Studio open the **Team Explorer**, using the drop down, select **Builds[[3]](#footnote-4)**

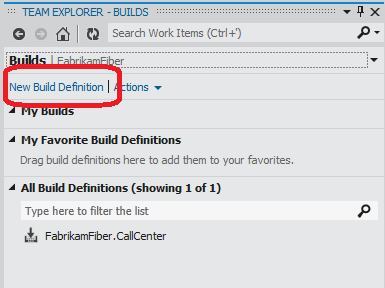


Figure - Create a new build

1. Select **New Build Definition.** Accept the default settings, the build should default to the name of **TestImpact.**
2. On the **Trigger** tab set the trigger to be **Continuous Integration,** this means the build will occur whenever there is a checking in the future.

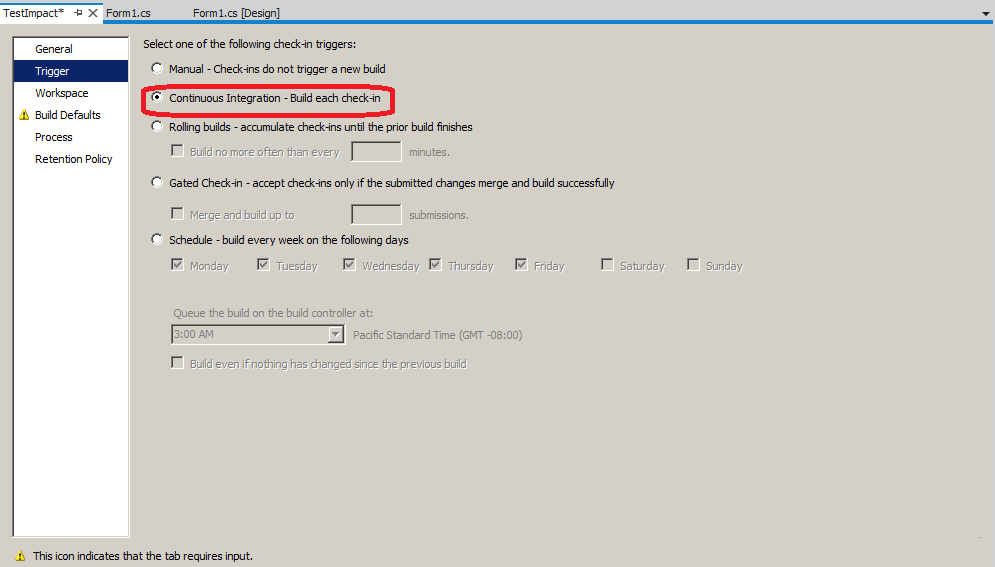


Figure - Setting the trigger

1. On the **Workspace** tab make sure that only the newly checked in path is listed in the workflow

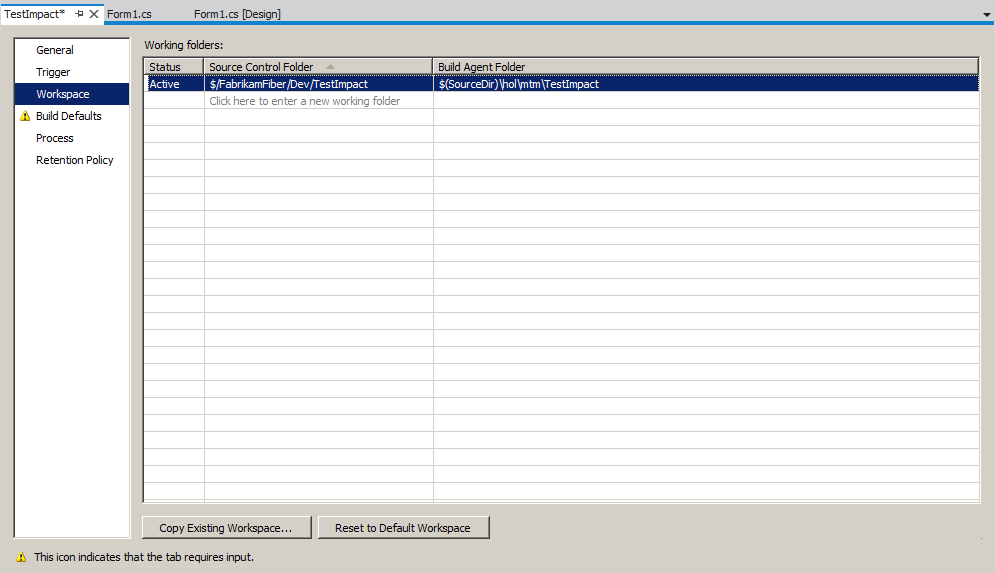


Figure - Workspace settings

|  |  |
| --- | --- |
| NoteNote | **Note** |
| This build will require a UNC folder share to act as a drops location for the files created during the build. This should be present on the HOL VM. If it does not  Create a folder e.g. c:\Drops  Right click on the folder and select sharing  Add **Everyone** to the list of people the folder is shared with **read/write** access    Figure - Creating a drops folder share | |

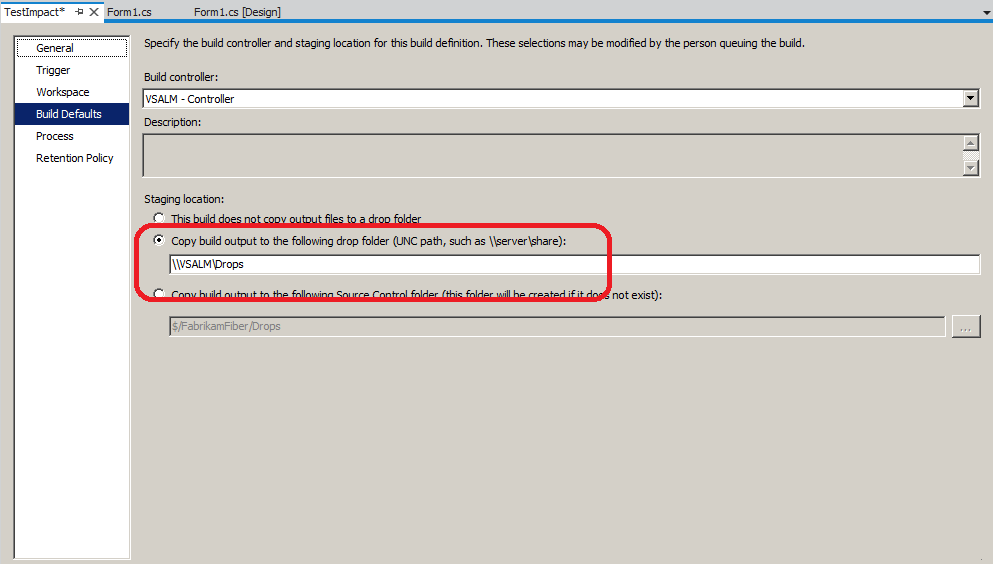
The **Drop** location on the **Build Default** tab needs to be set to a UNC folder e.g. **\\VSALM\Drops**  


Figure - Setting the drop location

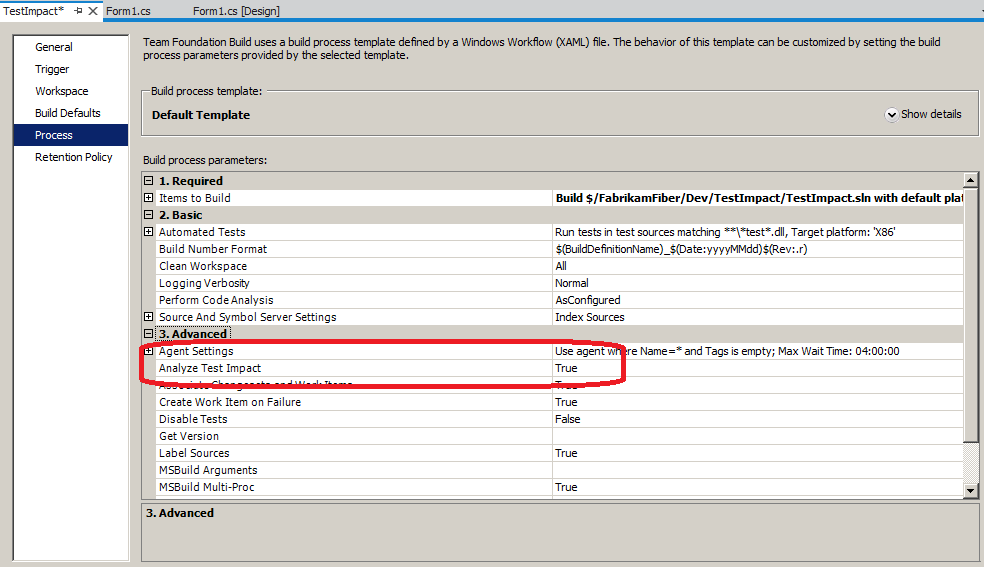
Note on the **Process** tab that **the Analyzer Test Impact** option is enabled by default.  


Figure - Enable Test Impact

1. Save the new build definition.
2. In the future the build will trigger automatically when a check in occurs, but for this first run, as the code is already in TFS and we want to check the build is working we will trigger it manually. To queue a build select the **Build** in **Team Explorer**, right clicking and selecting **Queue New Build**

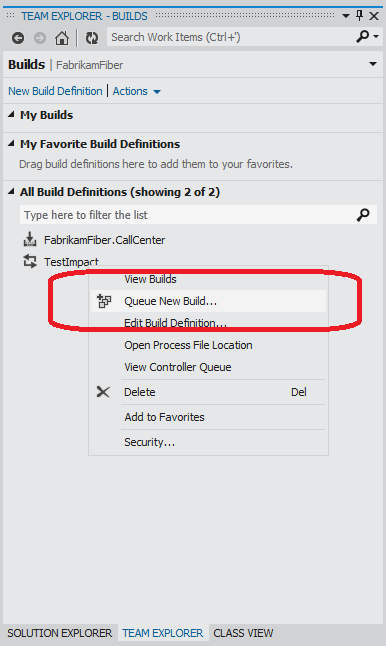


Figure - Queue a build

1. The build should complete and the files placed in the drops location

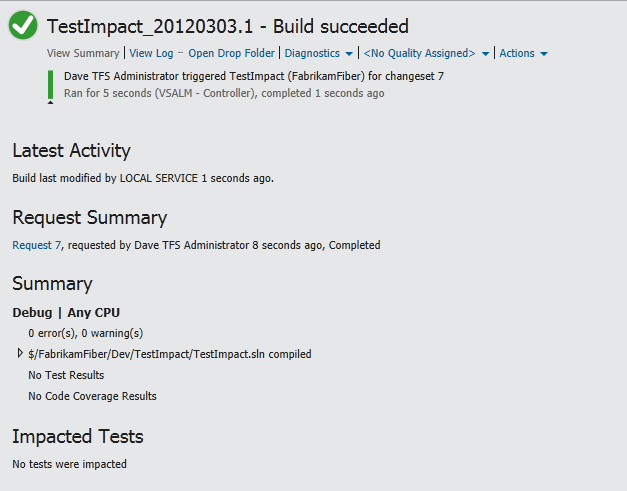


Figure - Completed build

### Task 3 – Creating the Manual Tests

We need to create manual tests that exercise both the buttons on the form.

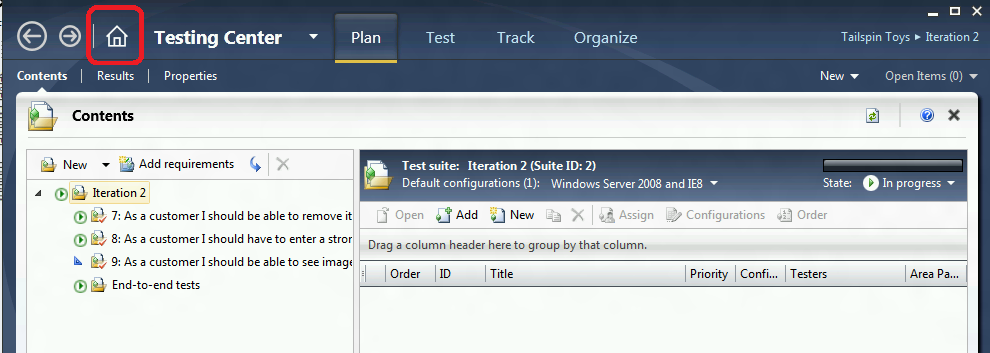
1. Open **Microsoft Test Manager** (MTM). If MTM has been used previously it will return you to the point you exited from. If you are not on the **Project Selection** dialog shown in step 2
   * Click the Home button  
     

Figure - Select Home Button

* + Click on **Change Project**

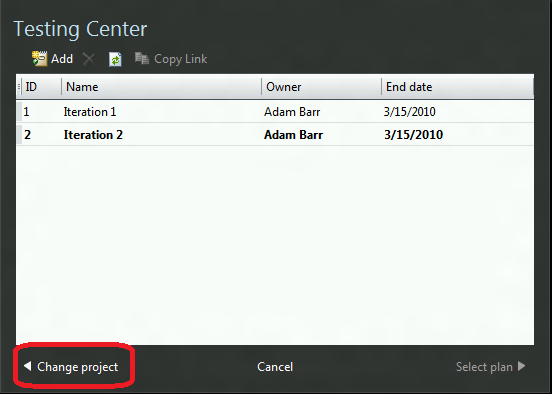


Figure - Change Project

* + You should be on the dialog detailed in step 2

1. Select the FabrikamFiber team project and press the **Connect Now** button. If you were previously connected to another Team Project you will be warned that open items will be closed.

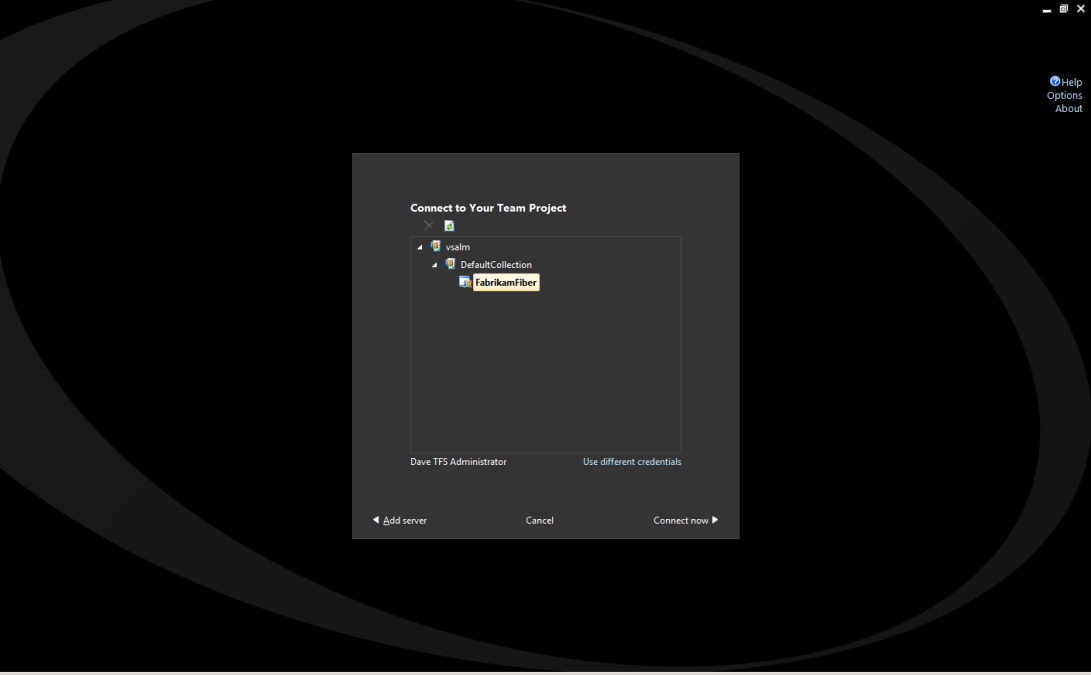


Figure - Connect to team project

1. Add a new Test Plan called **Test Impact,** and press **Select Plan**

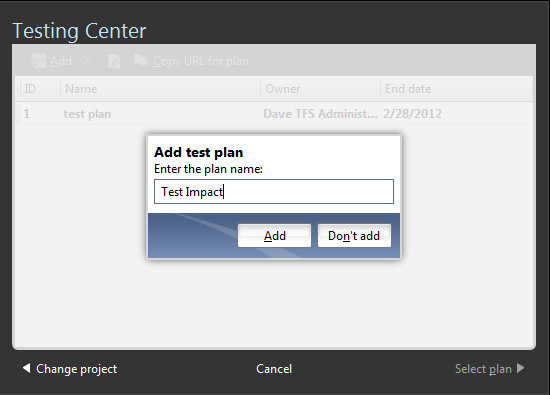


Figure - Create a test plan

1. You will now be in MTM on the **Plan** tab. Add a **New** test using the New button

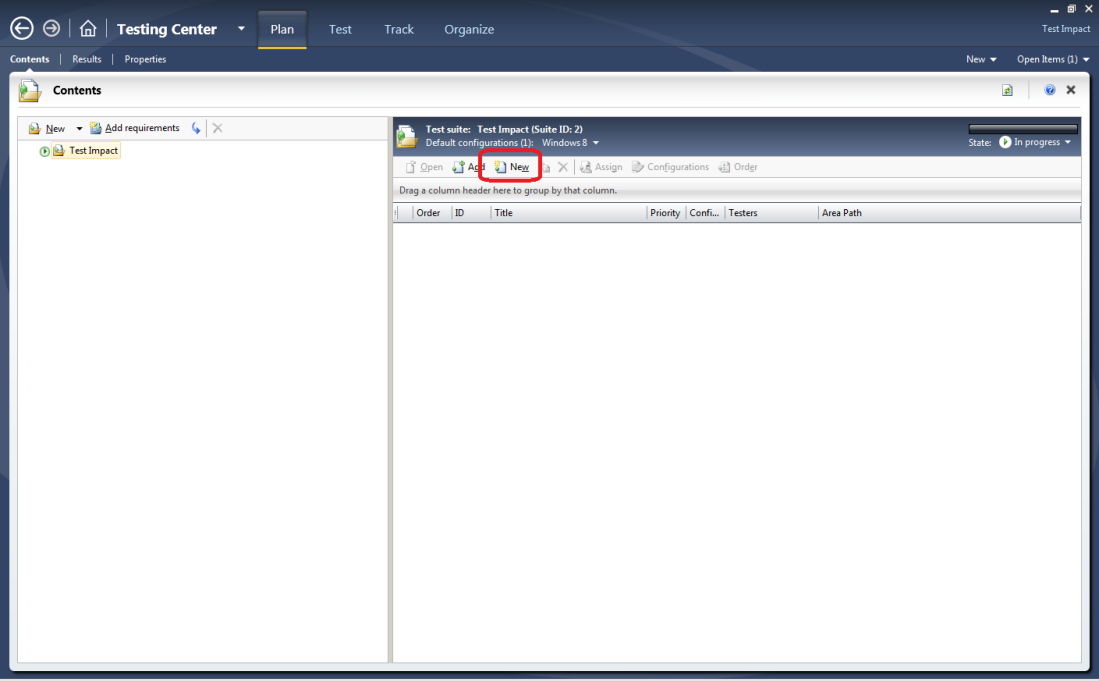


Figure - Add a new test

1. When the test work item opens set the title as “Button 1 Test” and enter the following steps

* Add a step
  + Action - Load Application
  + Expected result – leave blank
* Add a second step with the following settings
  + Action – “Press Button 1”
  + Expected result – “Message should say button 1 pressed”
* Finally add a step
  + Action – Unload Application
  + Expected result – leave blank

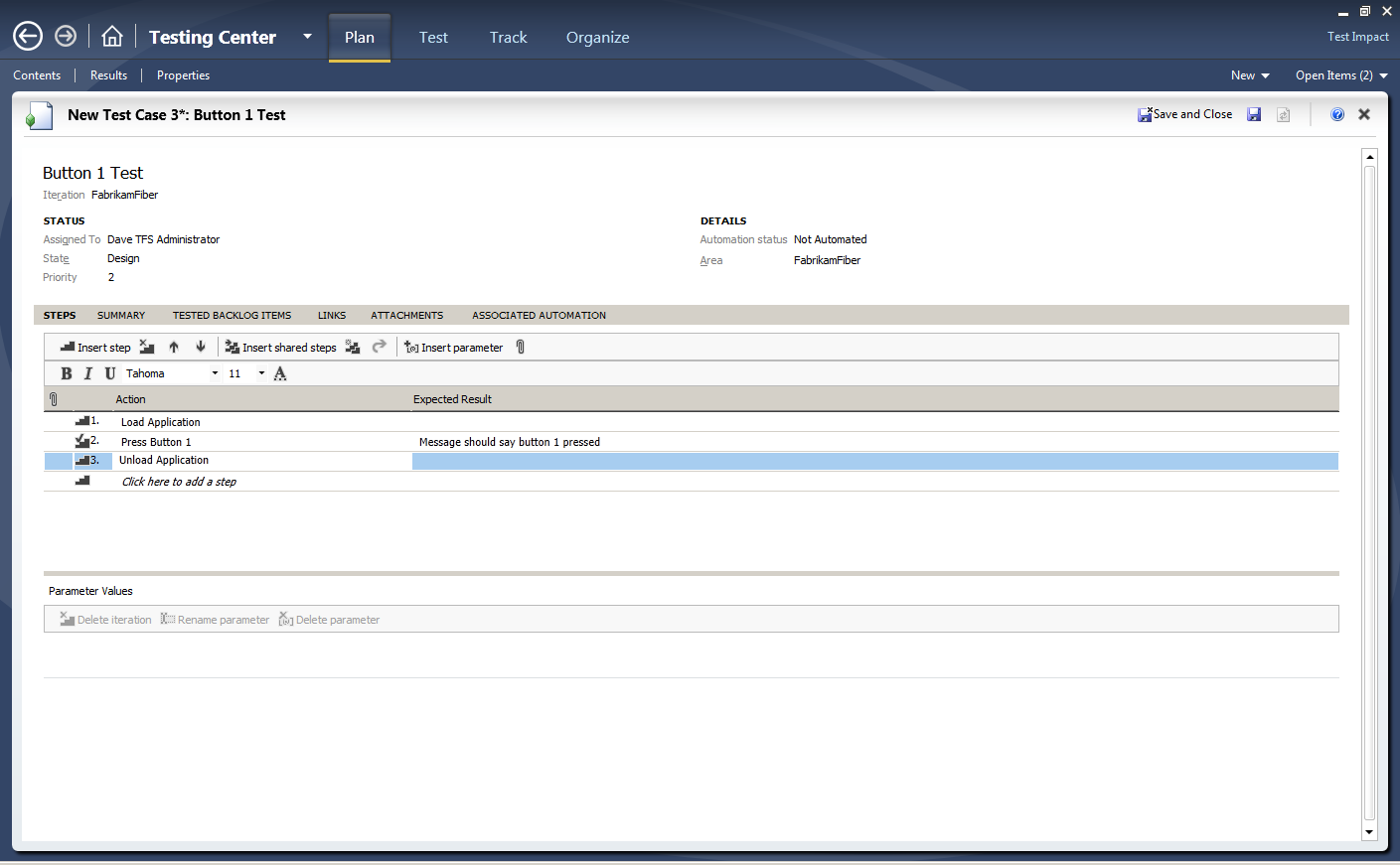


Figure - Create a new test

1. Save and close the test
2. Repeat steps 5 and 6 with the equivalent test for Button 2

|  |  |
| --- | --- |
| NoteNote | **Note** |
| You could also use the Create copy and add to suite found on the right click menu.  If you do this remember you will need to edit the tests title and test steps | |

### Task 4 – Setting up the gathering the Test Impact for manual test runs

1. In MTM, on the **Plan** tab, select the **Properties**
2. In the Manual run settings select the **Local Test Run** and select the **Open** link

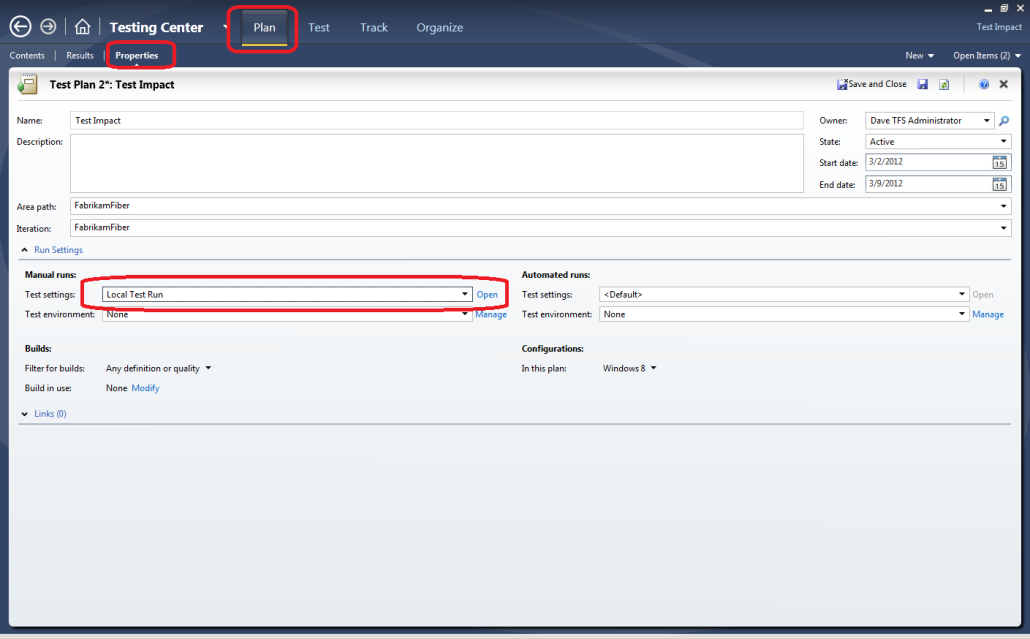


Figure - Test plan properties

1. Make sure that the **Test Impact** diagnostic data collector is enable

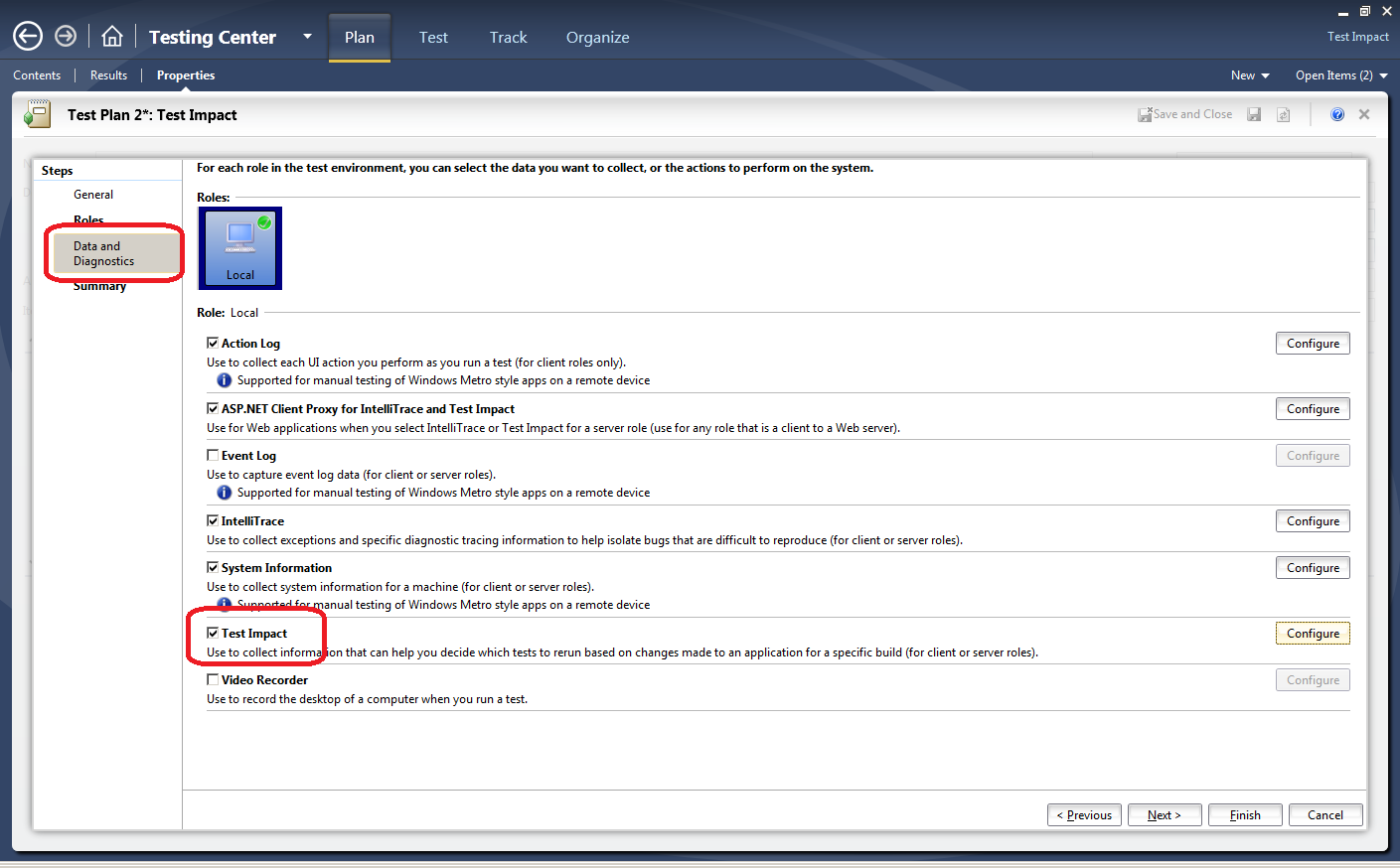


Figure - Enabling test impact

1. Press the **Finish** button to save the changes to the test settings
2. Finally press **Save and Close** to save the Test plan properties

|  |  |
| --- | --- |
| NoteNote | **Note** |
| For this HOL you do not need to edit these options, but you do have the option to configure what will be monitored by the collector i.e. select which assemblies and/or processes to monitor or ignore    Figure - Configure Diagnostic Data Adapter - Modules  Also on the **Advanced** tab of this dialog you are warned that if you wish to use Test Impact with an ASP.NET application on IIS then you must, as well as checking the checkbox in this dialog, also enable the **ASP.NET client proxy for IntelliTrace and test impact** collector.    Figure - Configure Diagnostic Data Adapter - Advanced Option  The **ASP.NET client proxy for IntelliTrace and test impact** can be found in the same list as the **Test Impact** collector. Though there is a configuration button for the client proxy data collector, no configuration is available or required.    Figure - Enable ASP.NET Client Proxy for IntelliTrace and Test Impact | |

### Task 5 - Gathering the baseline Manual Test data

To use Test Impact Analysis you need to run each test at least once without any errors against a past build

|  |  |
| --- | --- |
| NoteNote | **Note** |
| Remember that test impact data is not collected when you file a bug, irrespective of whether the test is marked as passed or failed. | |

1. IN MTM switch the **Test** tab. You should see your two test in the list
2. Highlight both tests and press the **Run with Options** button; we need to select this option to associate our test run with the correct build.

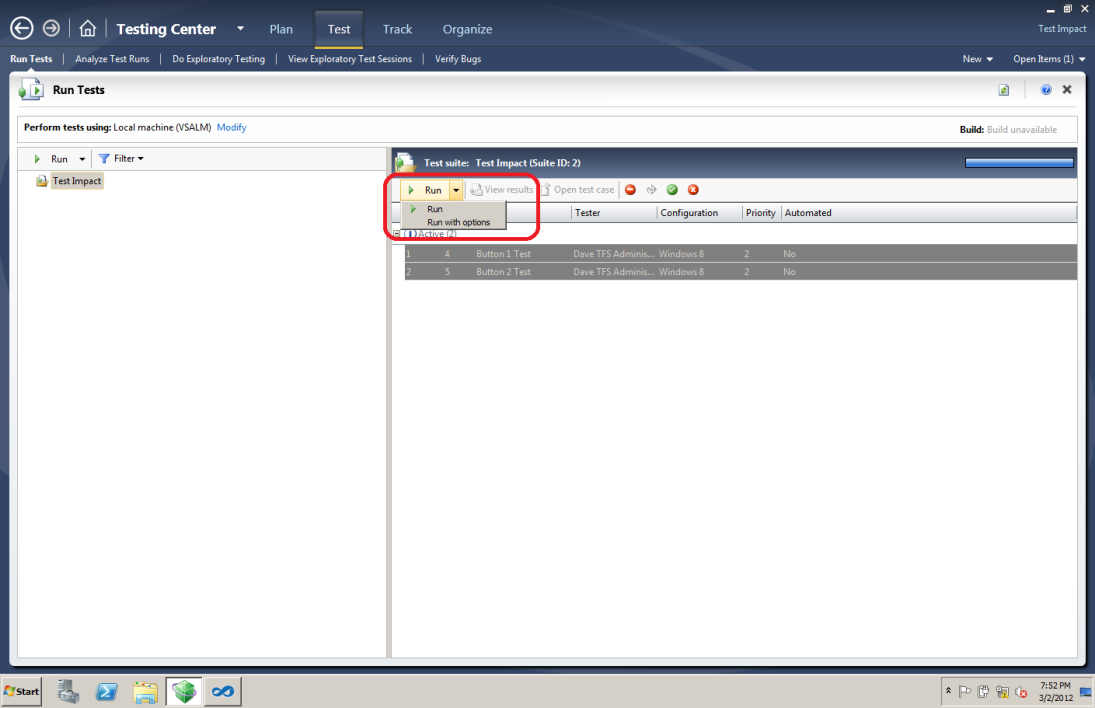


Figure - Run tests

|  |  |
| --- | --- |
| NoteNote | **Note** |
| On a production system, the association of a test run and a build will usually be done in the **Plan**, **Properties** section of MTM. It would not usually be the choice of the tester when they run their tests. | |

1. Select the previously run build and press the Run button. If the build is not shown in the list close the dialog and press the **refresh** button on the plan tab then select **Run with options** again.

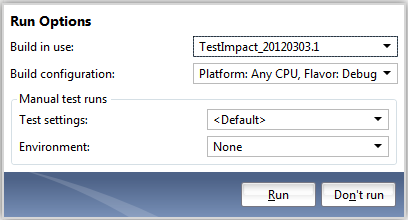


Figure - Selecting a build

1. MTM will change to testing mode
2. Open the drops location, as created in the build, to find the EXE to test e.g. **\\vsalm\drops\TestImpact\TestImpact\_20120303.1**

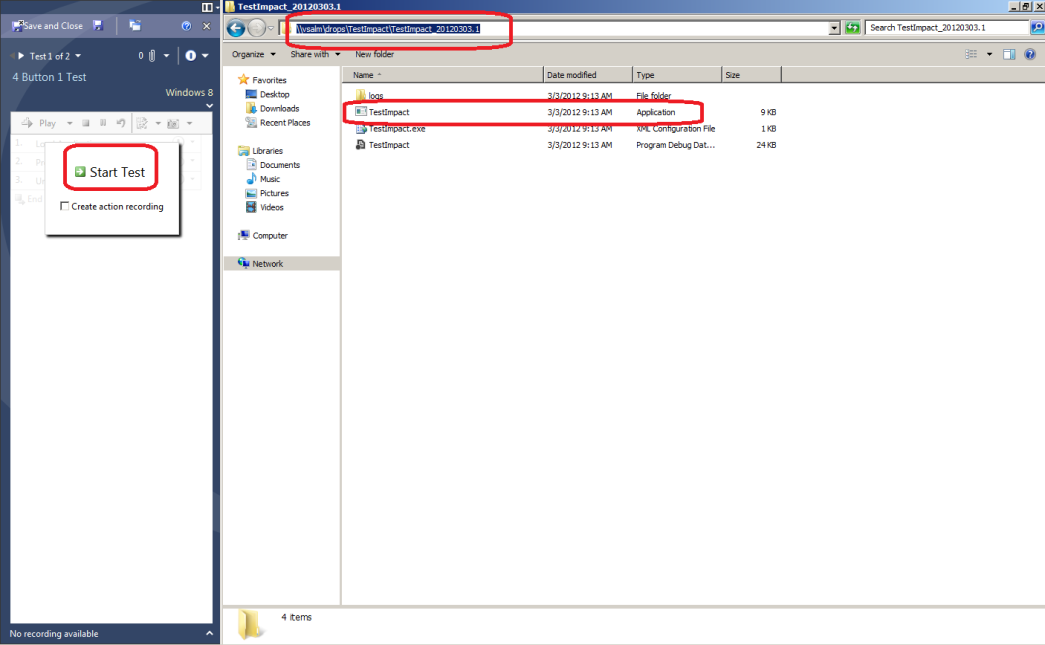


Figure - Starting a manual test

1. Press the **Start Test** button. There is no need to check the ‘Create action recording’ checkbox as we are not covering action recording in this HOL.
2. For Test 1, follow prompts in the test, marking each step as complete as it is done

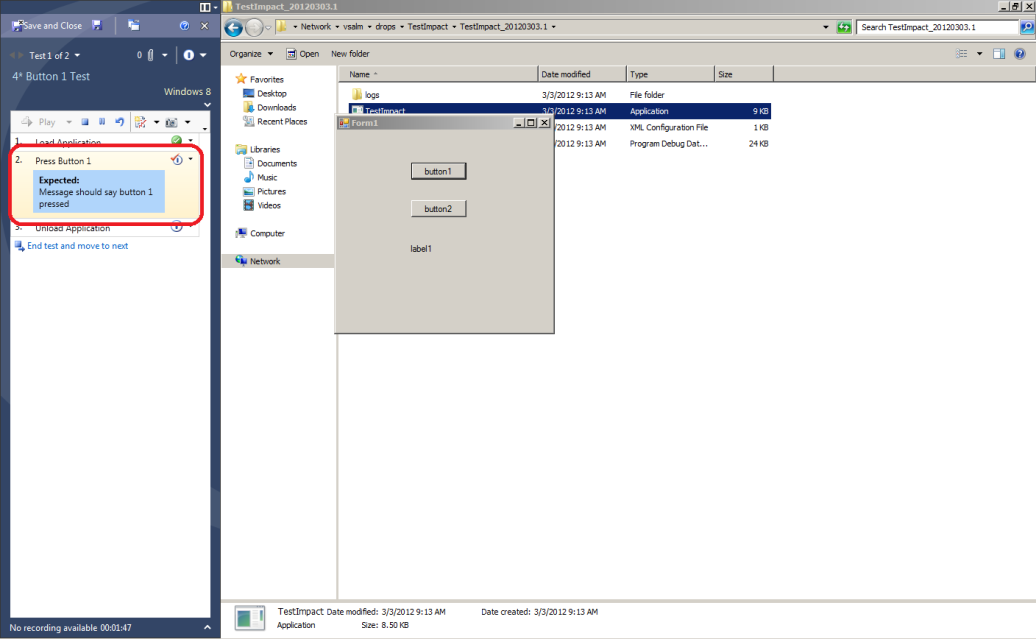


Figure - Running a manual test

1. When all the test are completed for Test 1 click the **End test and move to next** link (under the last test step)
2. Repeat steps 6/7 for Test 2, pressing the **End test** link when test 2 is completed
3. With both test competed press the **Save and Close** button at the top left of the screen
4. On the Test tab both tests should be marked as passed.

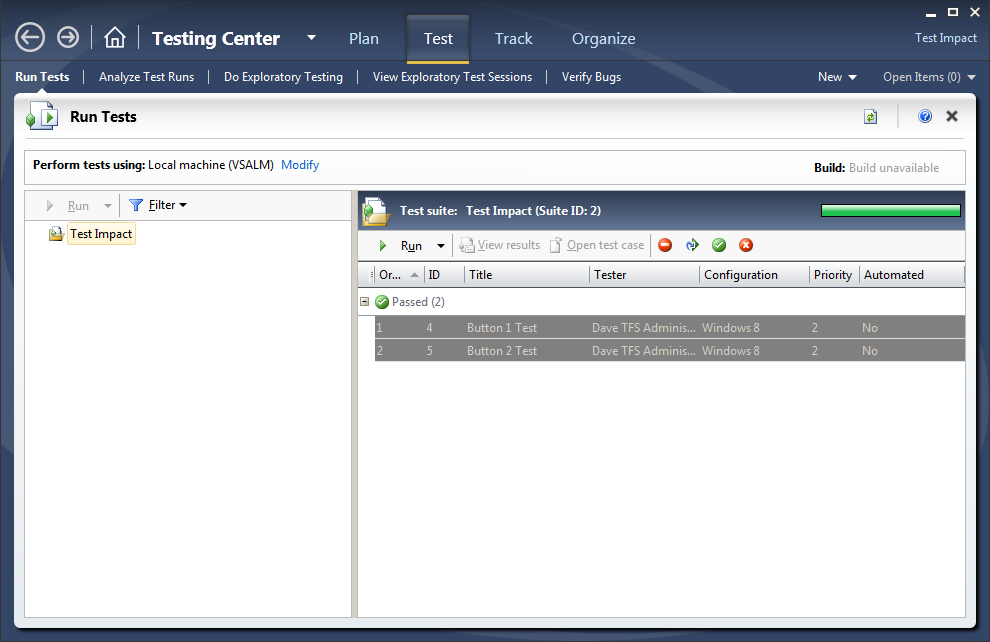


Figure - Tests passed

### Task 6 –Impacting the Manual Test

The sample application will now be altered to impact one of the manual tests but not the other.

1. In Visual Studio open the sample application solution
2. Refactor the click event for the Button2 on Form1 so the way the text message is generated is changed. This should have the effect of impacting test 2 but not test 1.

private void button2\_Click(object sender, EventArgs e)

{

this.label1.Text = GetButton2Message();

}

private string GetButton2Message()

{

return "Button 2 Pressed message from another method";

}

1. Build the solution within Visual Studio, to make sure there are no silly typing errors.
2. Select the Team Explorer -> Pending Changes hub and check in the pending change to **Form1.cs** into TFS. You can accept the default of not entering a comment or a work item to associate the change, but this is not considered best practice for production code.
3. As the automated build is set to Continuous Integration the build should automatically run. Selecting the Team Explorer -> Build hub should see the new build running.

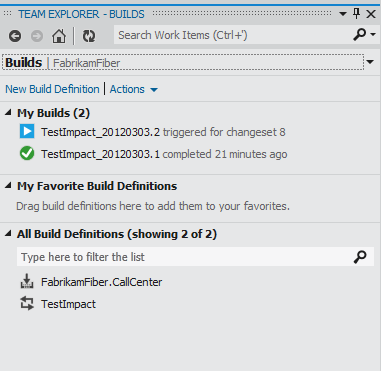


Figure - Build in progress

1. You can click on the active build to see the progress of the build while it is running and to see the details of the build when it is finished. If you scroll to the bottom of the completed build summary report you should see the impacted tests

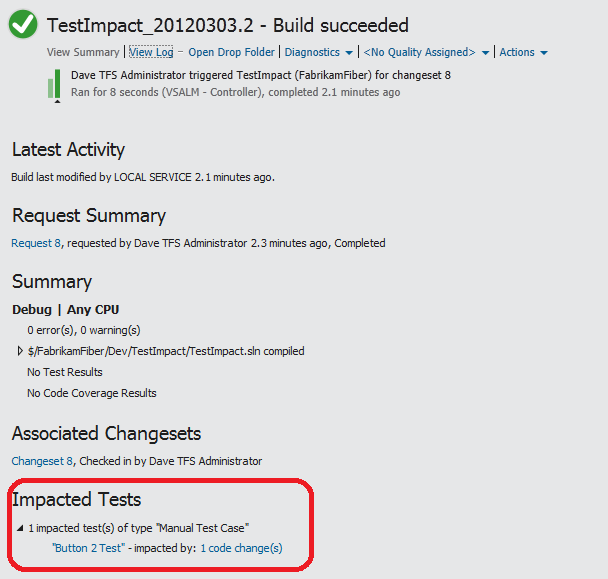


Figure - Completed build showing impacted tests

1. You can click the test link to open the Test case work item in Visual Studio to see the details of the test

### Task 7 –Viewing Impacted Tests in MTM

The tester can see which tests are impacted from within MTM

1. Open MTM and select the **Track** tab, then select **Recommended Tests**

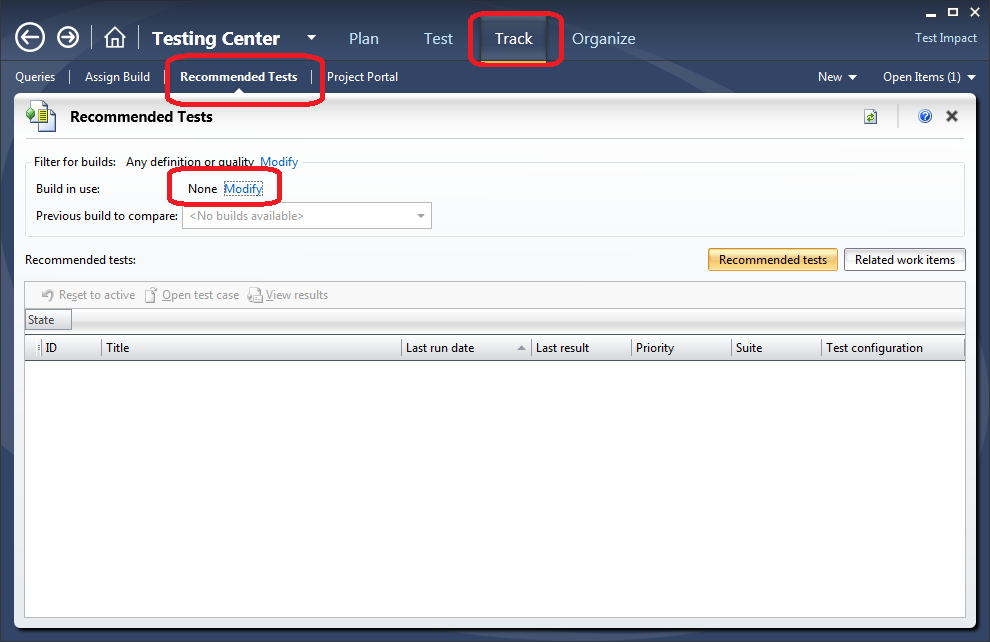


Figure - Recommended tests tab

1. Click the **Modify** build in use link and you are taken to the **Assign Builds** tab.

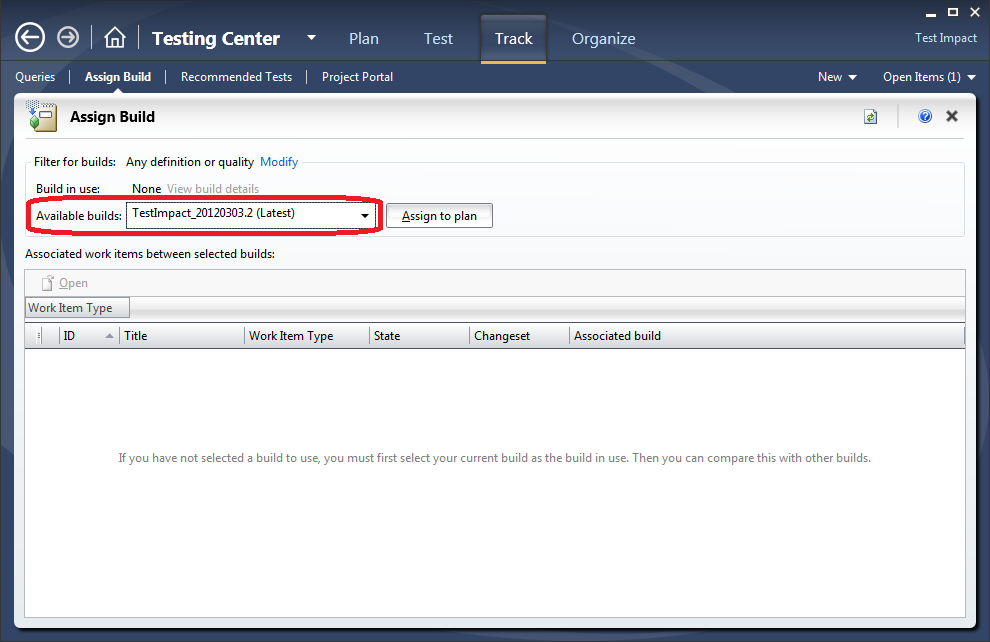


Figure - Selecting a build

1. Make sure the latest build is selected. If it is not then use the **Available Builds** drop down to select the required build and the **Assign to plan** button to select it.
2. Return to the **Recommended Tests** page
3. Make sure the build you ran the manual test against is select in the **previous build to compare** drop down
4. The impacted tests should be listed in the lower part of the screen

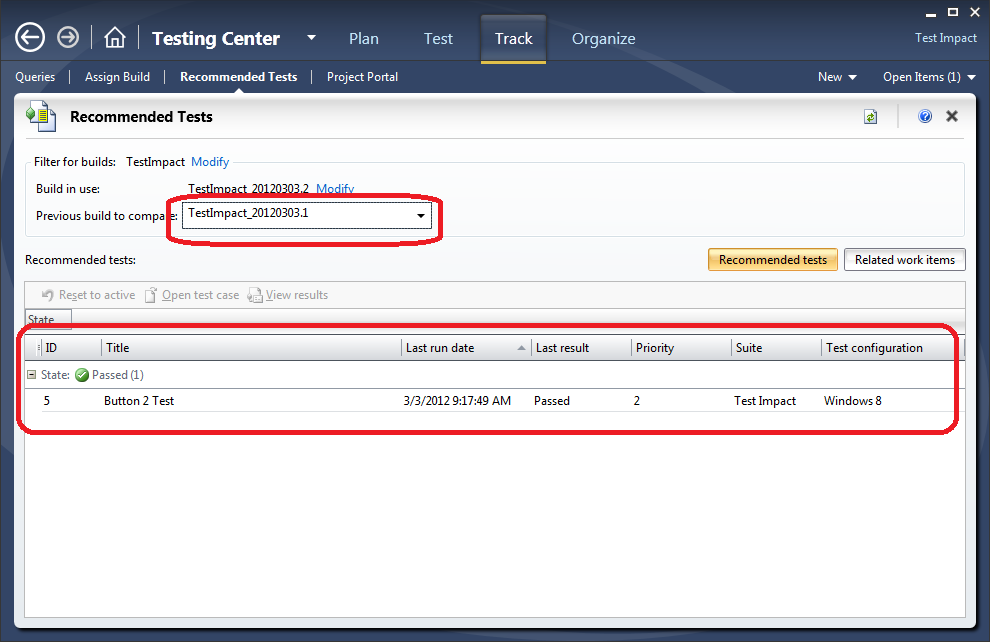


Figure - Recommended Tests

1. If you highlight the impacted test you can right click (or the use the menu bar options) to reset the test to active so it appears on the test suite as test to run.

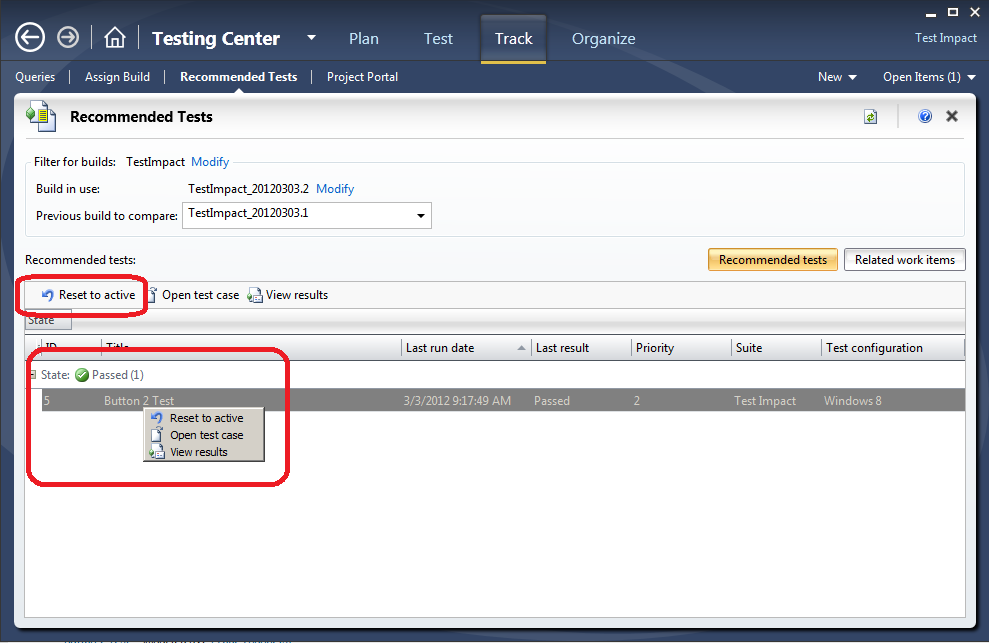


Figure - Resetting active flag

1. If you return to the Test tab you should now see that the button 2 test has been set as active, prompting the tester to re-run it

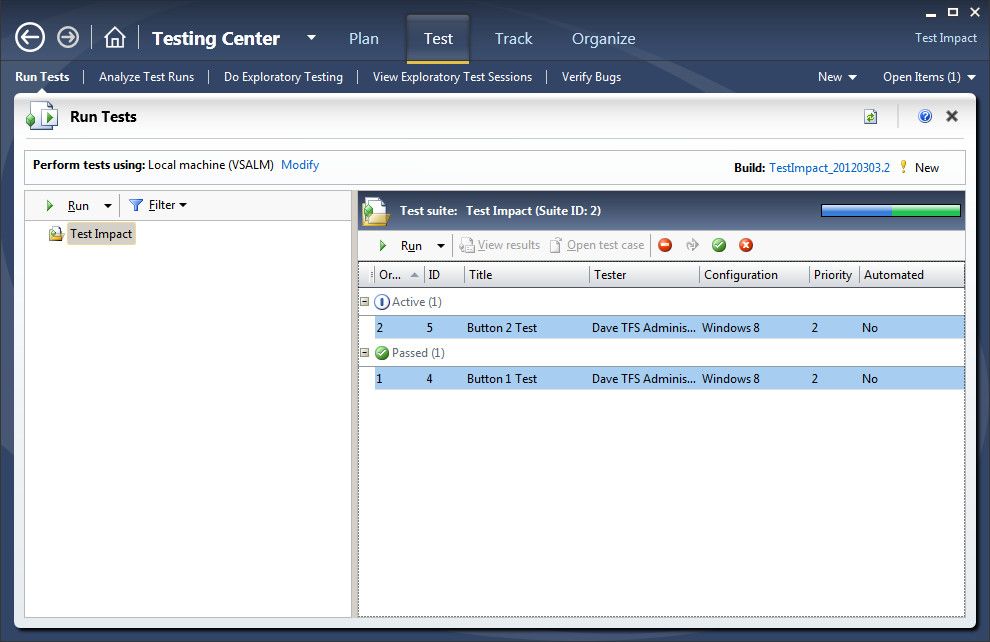


Figure - User prompted to re-run a test

HinweisREVIEW

We created a simple WinForms application, checked it into Team Foundation Server and created an automated build. We have then create manual tests, edit the application, allowed TFS to rebuild it then seen how we are shown which manual tests are impacted.

The most important point to note is that test impact analysis will only work one a test has been base-lined i.e. been run once successfully

# References

<http://msdn.microsoft.com/en-us/library/dd286743(v=vs.110)>

<http://msdn.microsoft.com/en-us/library/dd286586(v=vs.110)>

## Visual Studio ALM Rangers Site

<http://www.tinyurl.com/almrangers>

1. This is the only point in this HOL that Fabrikam Fiber is used. The remainder of the HOL uses a small sample application to demonstrate the features. Hence this exercise could be done against any Team Foundation Server 2012 server and associated team build setup. [↑](#footnote-ref-2)
2. You can use the Ctrl 0 (zero) then P keyboard shortcut here [↑](#footnote-ref-3)
3. Or use Ctrl 0 (zero) and then B [↑](#footnote-ref-4)