Hands-on Lab

Lab 6.1: (Advanced) Adding Windows Azure and Windows Phone 8 to the Trey Research Environment

Table of Contents

Objectives

This lab is an advanced version of how to set up the Trey Research environment. It is considered advanced because it differs from the standard lab in that it contains the steps for adding the Windows Phone 8 application and the Windows Azure services.

Note: You cannot use the Brian Keller VM with this lab.

You install the initial version of the Trey Research application and create the initial version of the Trey Research release pipeline. You can read about this version of the pipeline in Chapter 2 of Building a Release Pipeline with Team Foundation Server 2012. During this hands-on lab, you will:

- Install the components that comprise the Trey Research application.
- Learn how the mobile client and the Windows Azure Windows Communication Foundation (WCF) service work together.
- Install the test artifacts. Some of these artifacts will be used in later labs.

- Use the Team Project Build Process default template to construct a simple release pipeline that includes continuous integration.

The Advance Trey Research application is located in the Lab06\Source folder.

Prerequisites

The prerequisite for completing this is that you have already installed the components listed in the Introduction.

- Visual Studio 2012
- Team Foundation Server 2012
- Team Foundation Build Controller 2012
- Team Foundation Server Build Controller 2012
- Test Controller 2012
- Microsoft Test Manager 2012
- Windows Azure SDK
- Windows Phone 8 SDK

Time

You should be able to complete this lab in approximately 40 minutes.

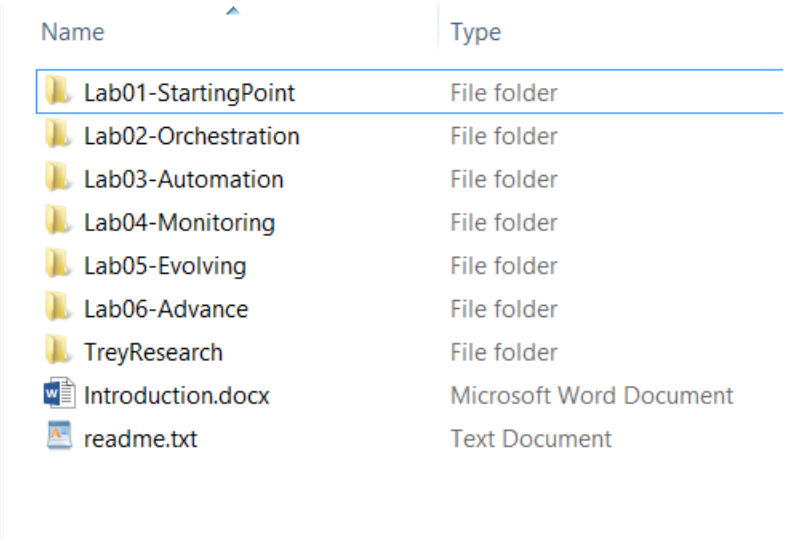
Exercise 1: Installing the Trey Research Application

In this exercise you install the Trey Research application, deploy the Windows Communication Foundation (WCF) service, deploy the Windows Presentation Foundation (WPF) service, and deploy the Windows Phone 8 client.

Task 1: Install the Trey Research Solution

In this task you create a TFS team project, map it to a local directory, and populate the directory with the Trey Research solution files.

1. Create a working directory where all the lab work and projects will be placed. In this lab, the default directory is **C:\HOL**.
2. Unzip the ReleasePipeline_HOL.zip file to a working directory. It should contain 6 lab folders and an Introduction.docx file. The following screenshot illustrates this.



Task 2: Replace the Basic TreyResearch Solution with the Advanced TreyResearch Solution

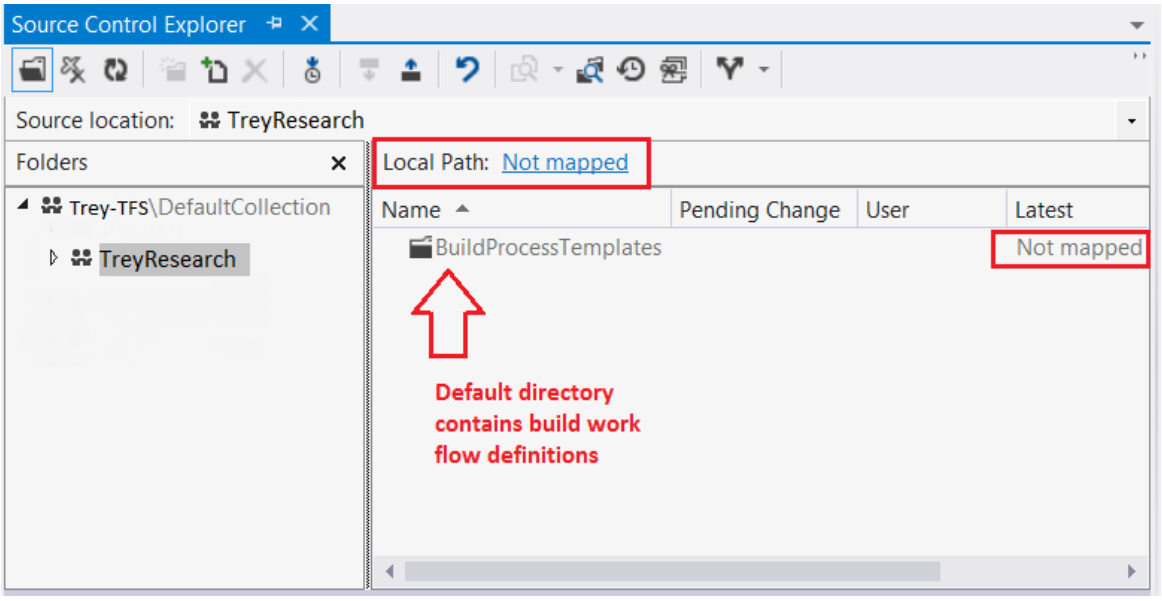
In this task you'll delete the basic **TreyResearch** solution and replace it with the advanced **TreyResearch** solution.

1. In File Manager go to C:\HOL\TreyResearch and delete all items in this directory.
2. Go to C:\HOL\Lab06\Source\TreyResearch folder.
3. Copy everything in this folder and paste it into C:\HOL\TreyResearch.

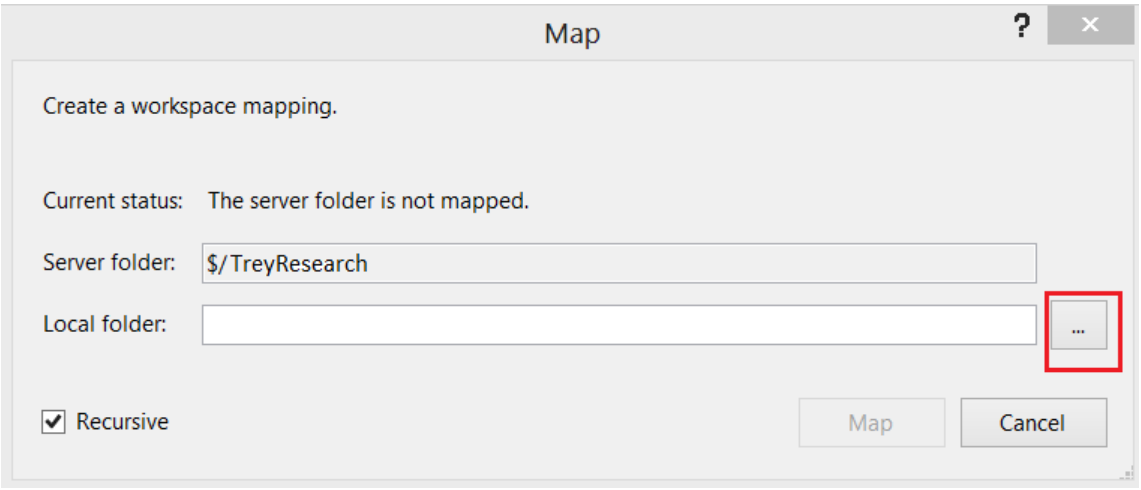
Task 3: Create the TreyResearch Team Project

In this task you create the TreyResearch team project.

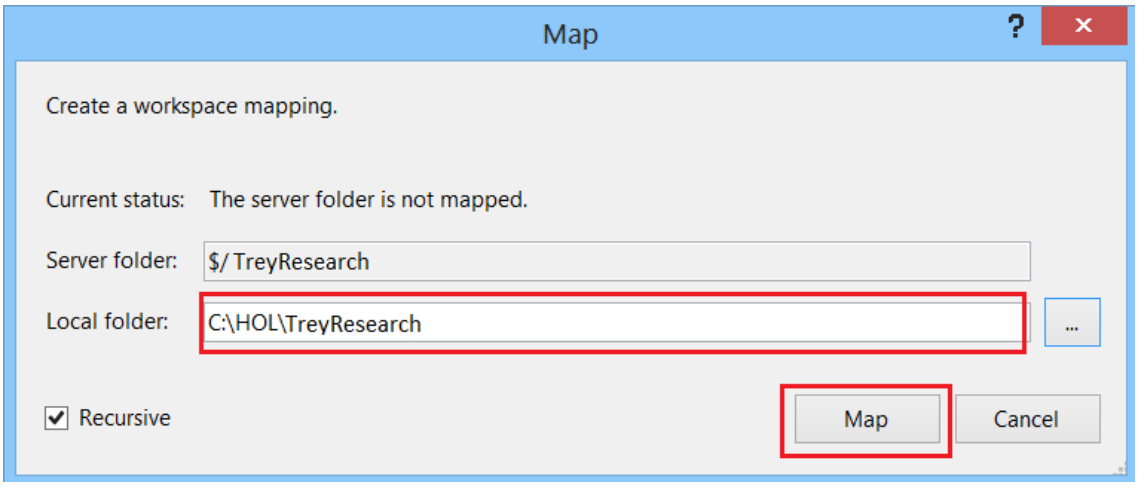
1. Open Microsoft Visual Studio.
2. Click **File**. Point to **New**. Select **Team Project**. Create a new team project and name it **TreyResearch**. Click **Next**.
3. Select the **MSF for Agile Software Development 6.x** process template.
4. Accept all the default options. Click **Finish**.
5. Select **Source Control Explorer**. Select the **TreyResearch** project. The default team project has a single **BuildProcessTemplates** folder. The **Local Path** is not mapped, as is shown in the following screenshot.



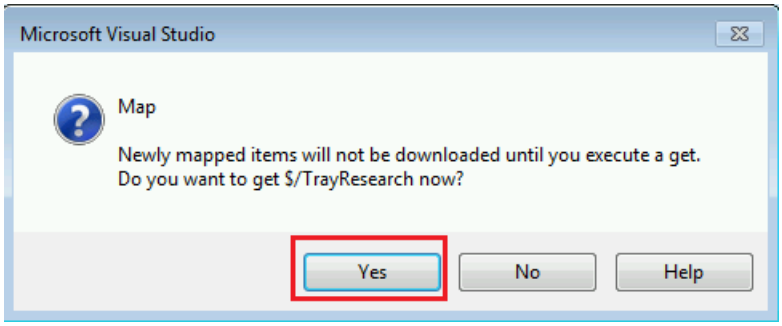
6. Click **Not mapped**. The **Map** dialog box appears.



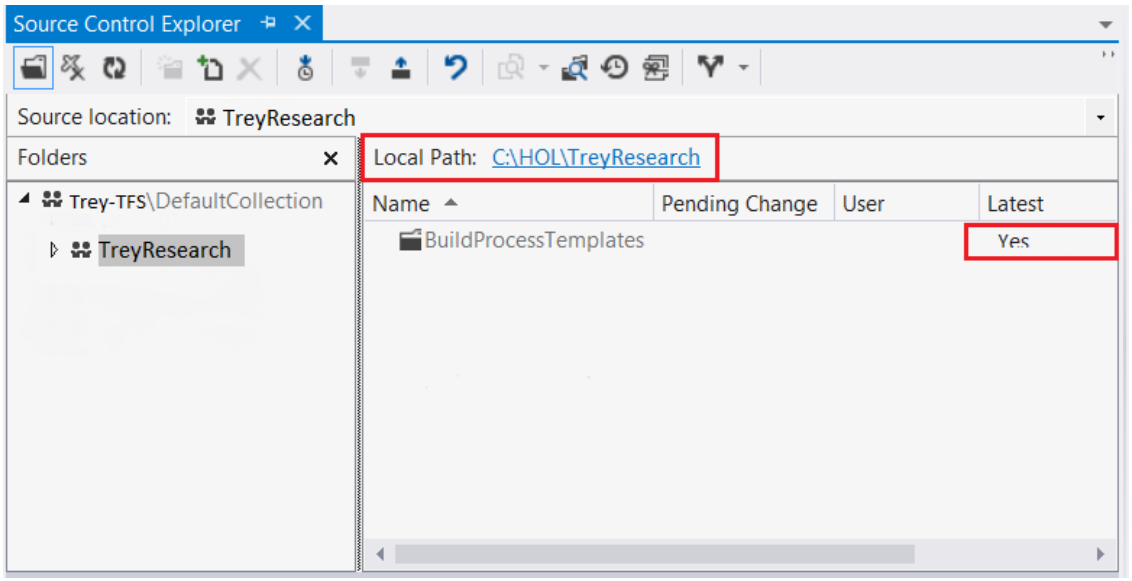
7. Click the **ellipses (…)** and navigate to the local directory you want to map to the team project. Browse to the existing C:\HOL\TreyReseach and click **Map**. This launches the **Map** dialog. Click **Map**.



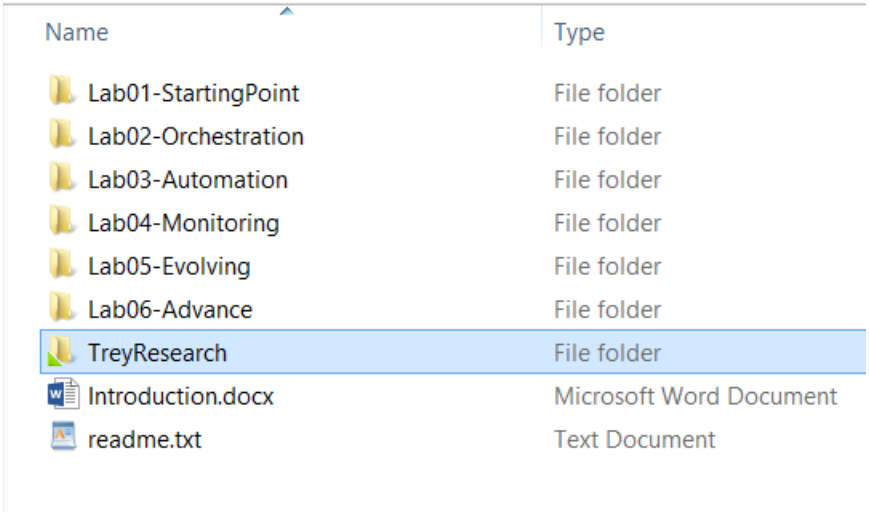
8. This launches the **Map** validation box, which asks if you want to get $/TreyResearch now? Click **Yes**.



9. The Solution Explorer view should now look like the following screenshot.



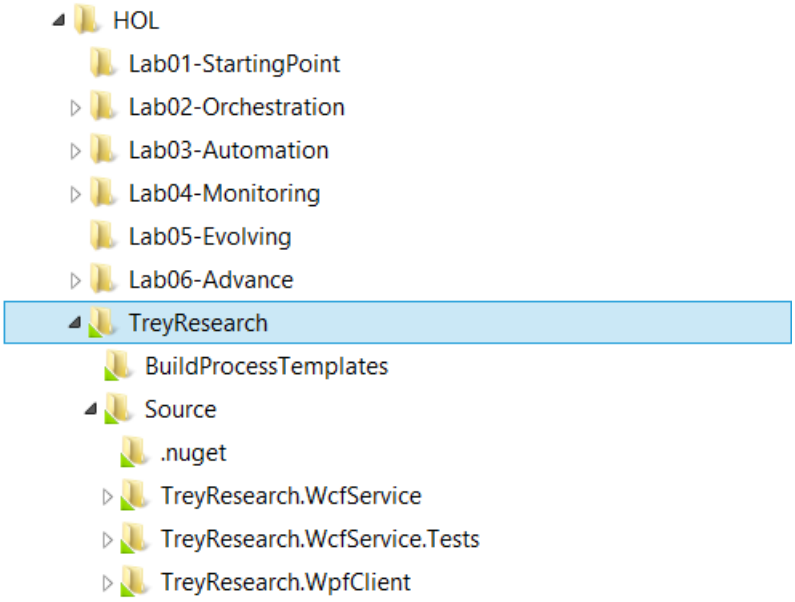
10. The local work space should look like the following screenshot. Note the green triangle on the **TreyResearch** folder. The green triangle means that the folder is now mapped to a TFS team project.



Task 4: Add the Trey Research Solution to the TreyResearch Team Project

In this task you add the Trey Research solution to the TreyResearch team project.

1. In Source Control Explorer, navigate to the team project, **TreyResearch**.
2. Right-click on the **project** and select **Add Items to Folder**.
3. Select the **Source** file folder and click **Finish**. The files are added to the project.
4. Check in the files. Right-click the **Source** file folder and select **Check In Pending Changes**.
5. In Team Explorer the **Pending Changes** dialog box appears. Click **Check In**. The check-In confirmation message appears. Click **Ok**.
6. The **TreyResearch** solution is now a part of the team project. The following screenshot shows the file hierarchy. The green triangles mean that the files are under source control.



Task 5: Build the Trey Research Solution

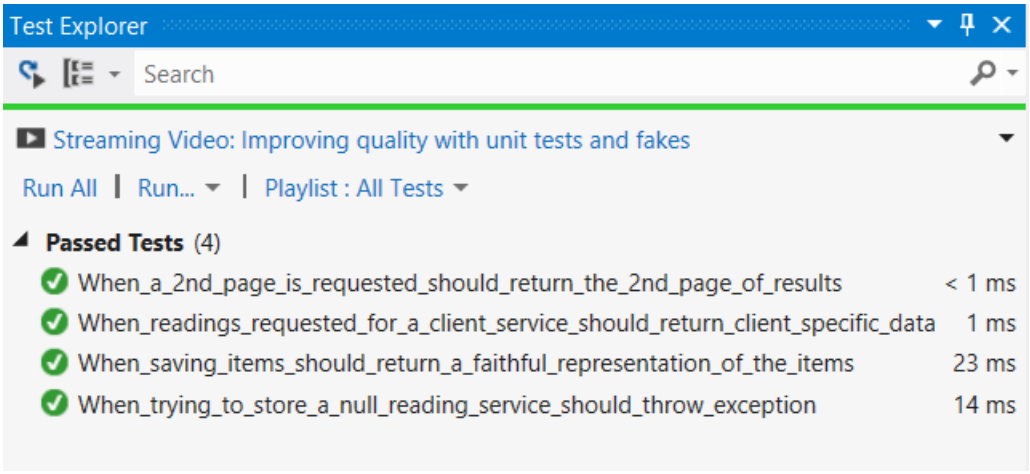
In this task you compile the Trey Research solution and test it. Because the lab use NuGet, make sure that it is enabled.

1. In Visual Studio, open **TreyResearch.sln**.
2. Click the **Tools** menu. Click **Options**. Click the **Package Manager** tab.
3. Make sure that **Allow NuGet to download missing packages during build** is selected.
4. On the menu bar, click **Build**. Click **Build Solution**. The solution should build successfully.

Task 6: Run the Trey Research Unit Tests

In this task you'll run the unit tests that are associated with the Trey Research solution.

1. Open Test Explorer. On the menu bar, click **Test**. Point to **Windows**. Click **Test Explorer**.
2. Click **Run All**. Visual Studio builds the solution and runs the unit tests. After they complete, the unit tests should be listed under **Passed Tests**.



Exercise 2: Publishing the Trey Research Solution

In this exercise you publish the Trey Research WCF service to Windows Azure.

For more information about how to publish cloud services to Windows Azure from Visual Studio, see Publishing Cloud Services to Windows Azure from Visual Studio.

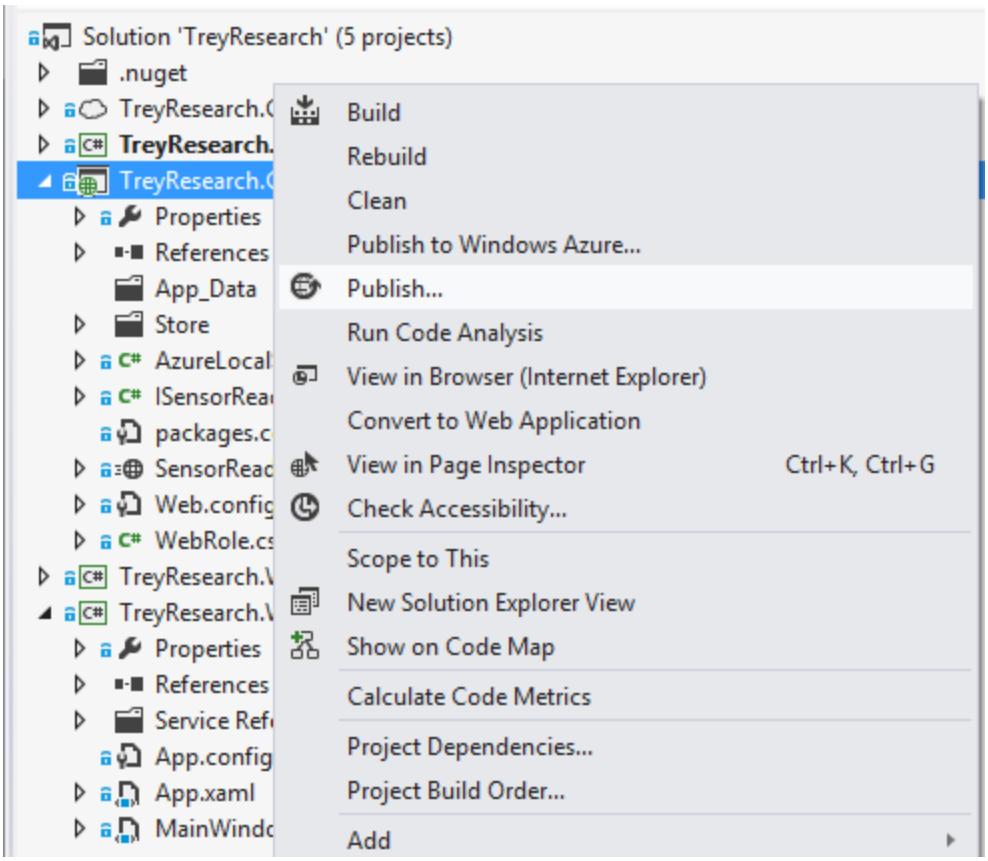
You should already have set up a Windows Azure VM. If you haven't, see the Introduction document, which has links to information about Windows Azure. You will need the following information:

- The Service URL of the endpoint address, which is: http://*YOUR_AZURE_DNS_SERVICE_NAME_GOES_HERE*.cloudapp.net.
- The log on credentials for your Windows Azure VM.

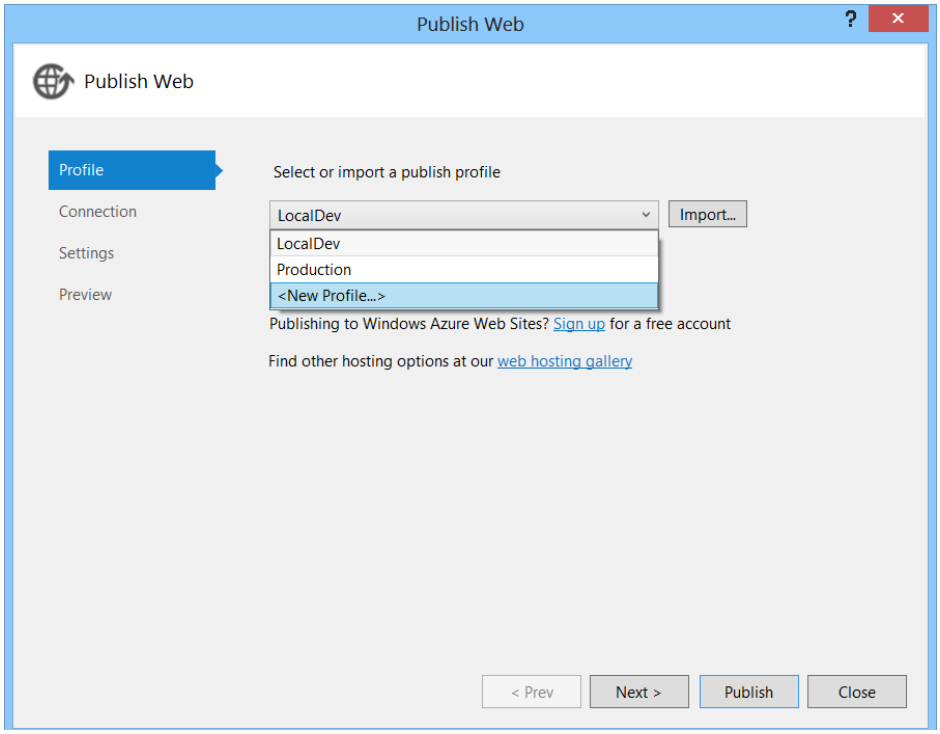
Task 1: Publishing the Web Role to the Windows Azure VM

In this task you publish the web role to the Windows Azure VM. You first create a profile, then publish the web role, and then validate that the WCF service was installed.

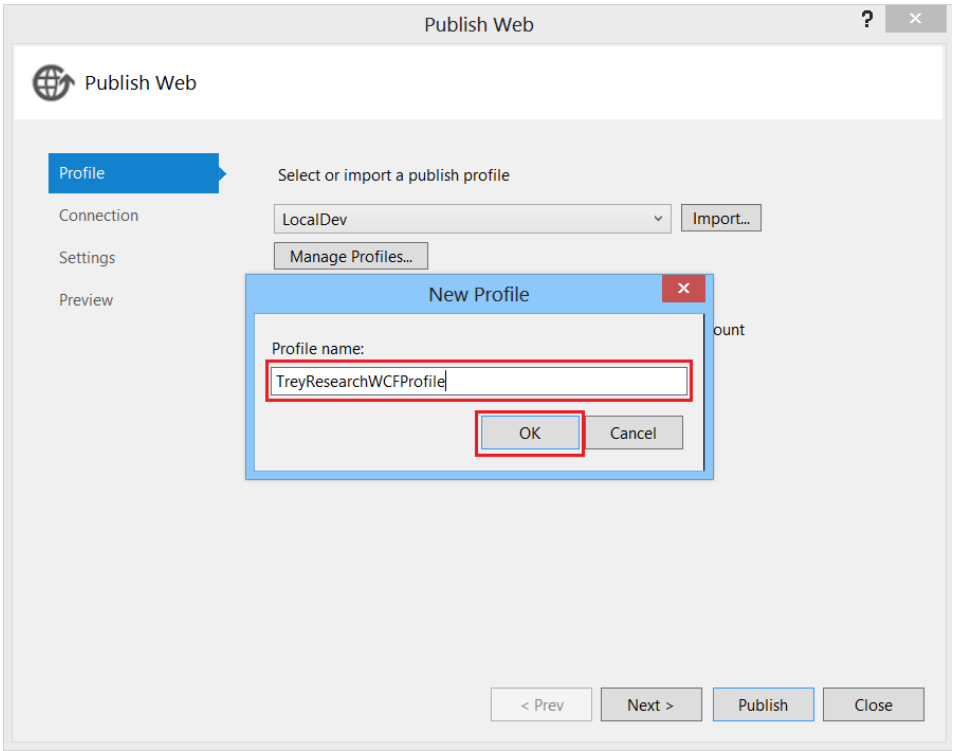
1. In Visual Studio, right-click **TreyResearch.Cloud.WcfService.WebRole**. In the context menu, click **Publish**.



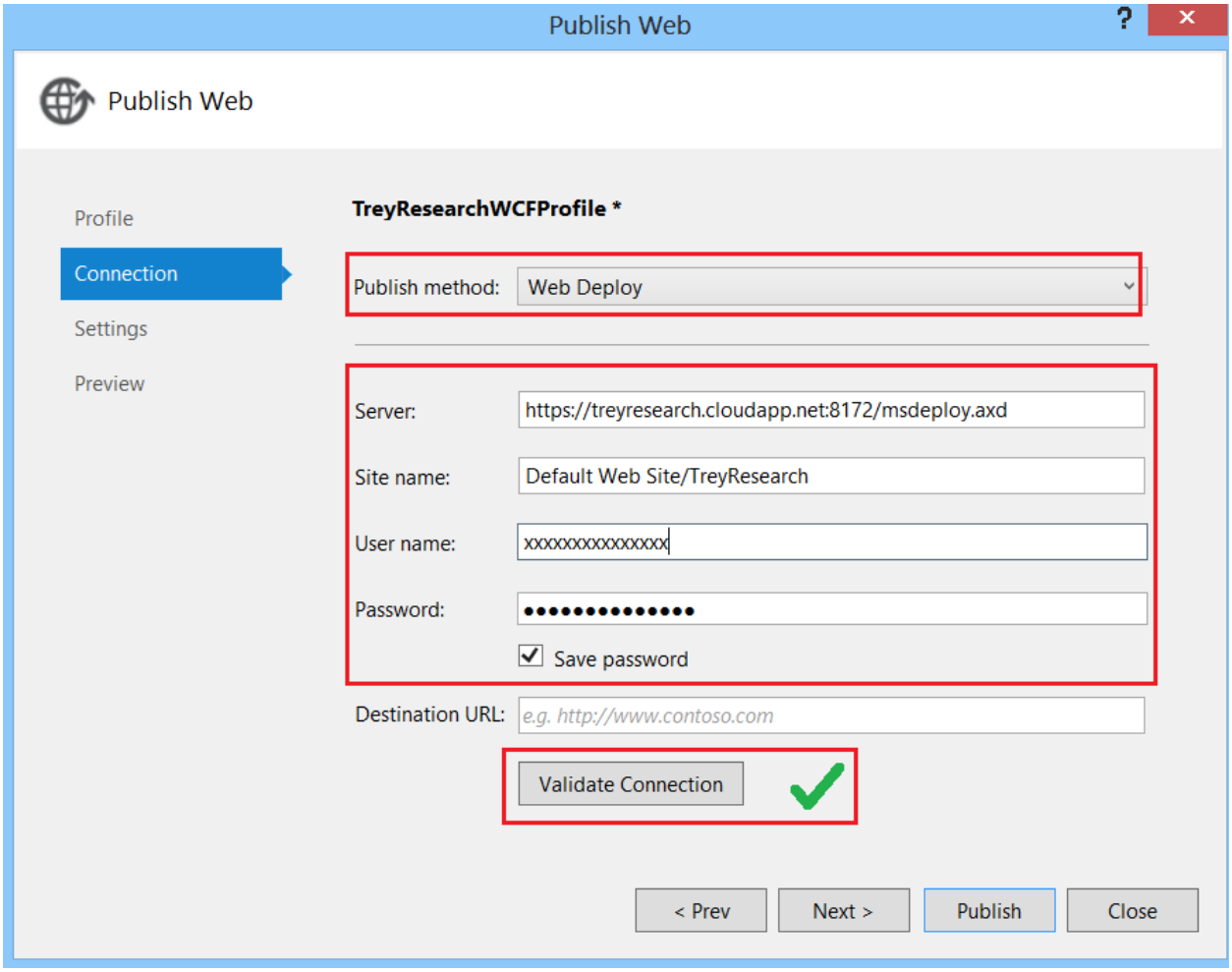
2. The **Publish Web** wizard appears. Click the **Profile** tab. From the **Select or import a publish profile** drop-down box, select **<New Profile>**.



3. The **New Profile** dialog box appears. Enter the profile name **TreyResearchWCFProfile**. Click **OK**.



4. The **Connection** tab opens. Fill in the fields highlighted in the following screenshot with your Windows Azure settings.



5. Make sure that the **Publish method** is set to **Web Deploy**. If it isn't, select it from the drop-down box.
6. In the **Server** box, enter https://*VM_DNS_NAME*:8172/msdeploy.axd. Use the DNS name you used when you created the Windows Azure VM.
7. In the **Site name** box, enter **Default Web Site/TreyResearch**.
8. In the **User name** box, enter the **user name** you used when you created the VM.
9. In the **Password** box, enter the **password** you used when you created the VM.
10. Click **Validate Connection** to validate that you can connect to the server using the settings you entered. A green check mark appears if the connection is valid.
11. Click **Publish**.

Task 2: Validating the WCF Service on Windows Azure

In this task you make sure that the WCF service is working.

1. Open a browser and go to the URL for the WCF service. In this example, the URL is http://treyresearch.cloudapp.net/treyresearch/SensorReadingService.svc. The **SensorReadingService Service** page should appear.

SensorReadingService Service

You have created a service.

To test this service, you will need to create a client and use it to call the service. You can do this using the svcutil.exe tool from the command line with the following syntax:

```
svcutil.exe http:// TreyResearch .cloudapp.net/SensorReadingService.svc?wsdl
```

You can also access the service description as a single file:

```
http://TreyResearch .cloudapp.net/SensorReadingService.svc?singleWsdl
```

This will generate a configuration file and a code file that contains the client class. Add the two files to your client application and use the generated client class to call the Service. For example:

C#

```
class Test
{
    static void Main()
    {
        SensorReadingServiceClient client = new SensorReadingServiceClient();

        // Use the 'client' variable to call operations on the service.

        // Always close the client.
        client.Close();
    }
}
```

Visual Basic

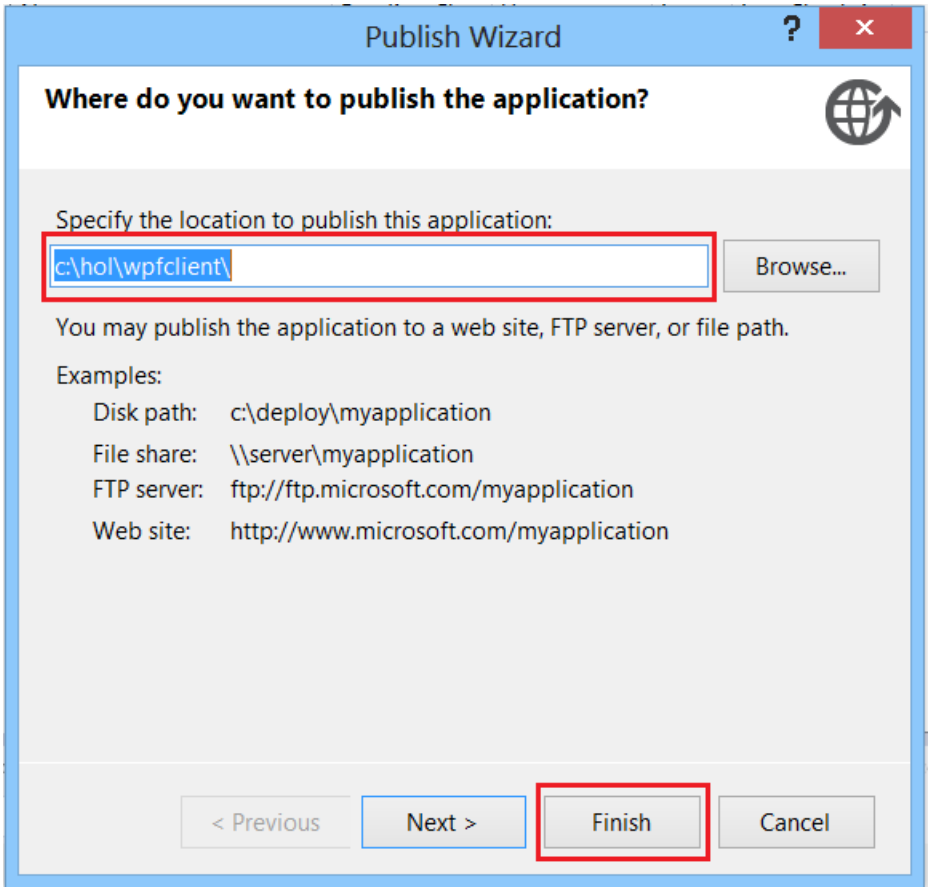
```
Class Test
    Shared Sub Main()
        Dim client As SensorReadingServiceClient = New SensorReadingServiceClient()
        ' Use the 'client' variable to call operations on the service.

        ' Always close the client.
        client.Close()
    End Sub
End Class
```

Task 3: Deploying the WPF Application

In this task you use Visual Studio to deploy the WPF application. You can deploy it to your local machine.

> **Note:** Make sure that the endpoints in the App.Config file are updated to the WCF Service that you deployed in Task 1, and that the **TreyResearch.WpfClient** service reference **SensorReadingServiceRef** is configured for the correct WCF service.

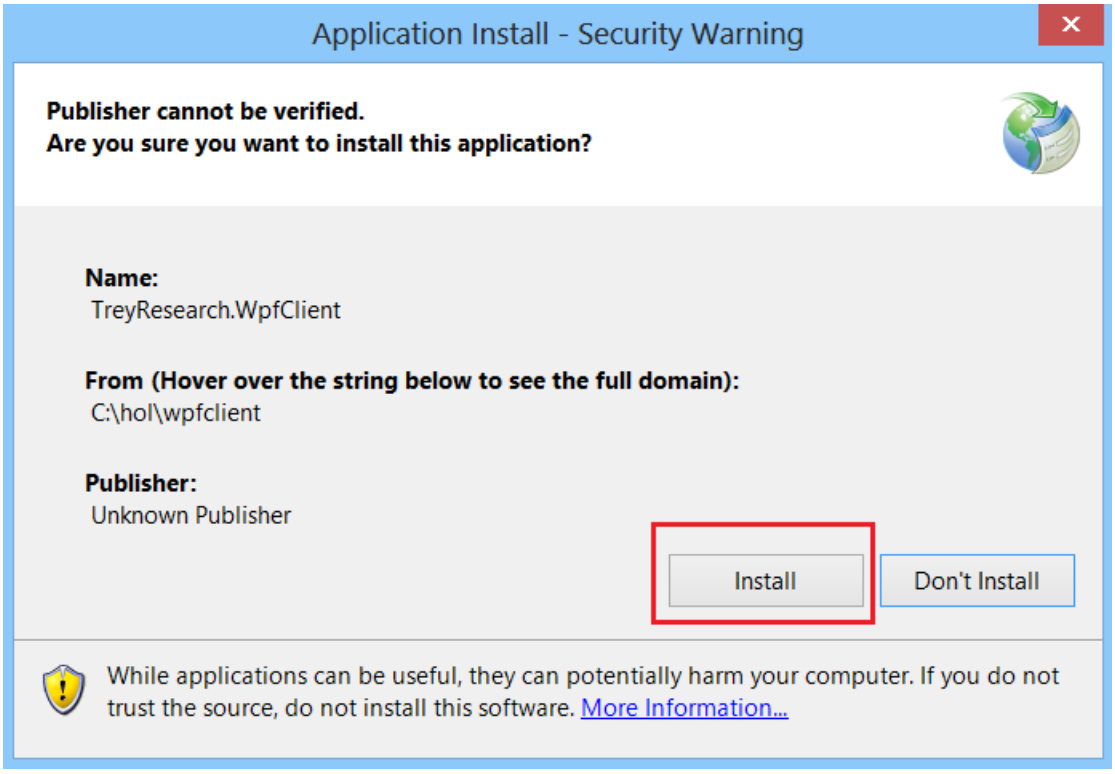
1. Right-click **TreyResearch.WpfClient** project. Click **Set as StartUp Project**.
2. Right-click **TreyResearch.WpfClient** and select **Publish**. The **Publish Wizard** appears.



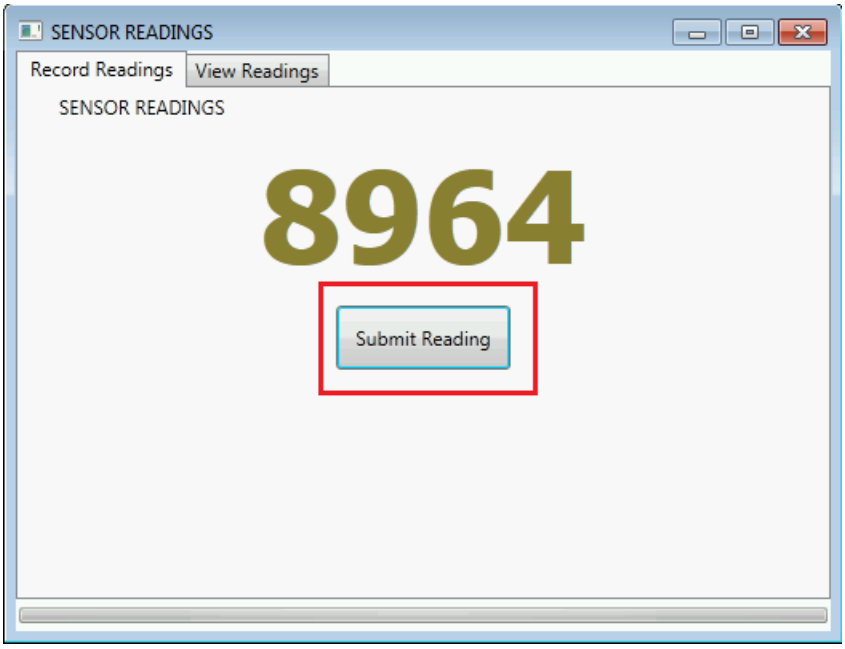
3. In the **Specify the location to publish this application** box, either enter or navigate to the location where the WPF application will be published. Click **Finish**. Visual Studio builds the project and publishes it to the location.

4. The folder at the designated location opens. Click on either **Setup.exe** or **TreyResearch.WpfClient.application**. The **Application Install** dialog box opens. Click **Install**.

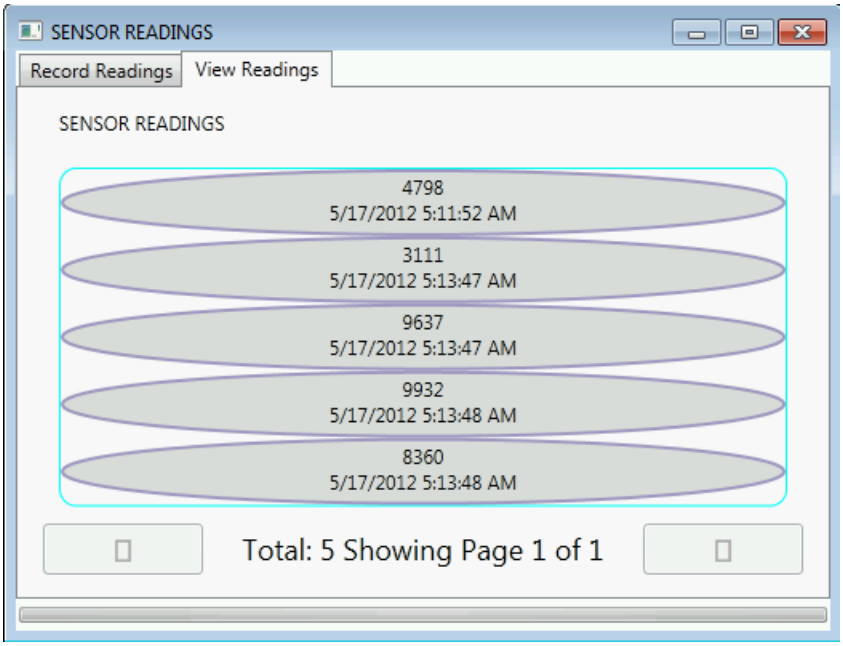
Note: The security warnings occur because there is no certificate for this application.



5. The application is installed and then launches. You should see the Trey Research application's **Sensor Readings** window.



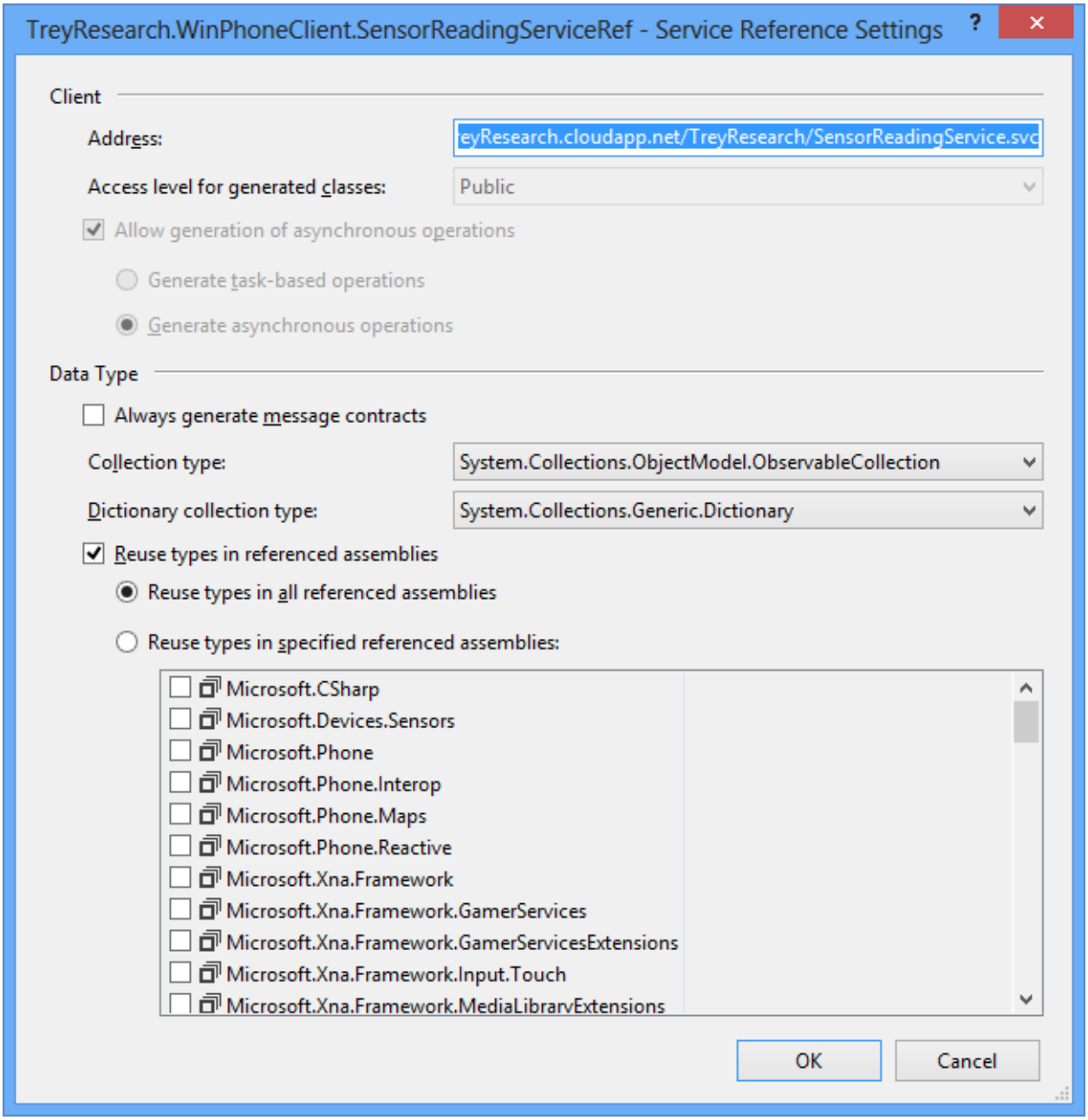
6. To test it, click **Submit Reading** several times. You are submitting sensor data to the WCF service. Click the **View Readings** tab to read from the WCF service. The following screenshot is an example of what you should see.



Task 4: Deploy the Windows Phone 8 Client

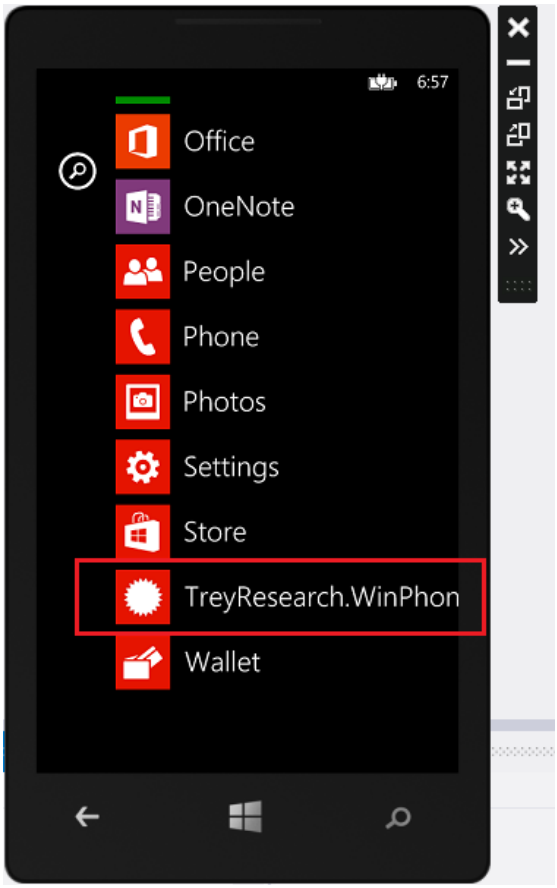
The Windows Phone 8 client is an optional component because it requires either Windows 8 or Windows Server 2012. The computer must also support Hyper-V. For more details about working with the Windows Phone 8 emulator, see [How to deploy and run a Windows Phone app.]

1. Open **Visual Studio** as an administrator.
2. Open **TreyResearch.sln**.
3. Set **TreyResearch.WinPhoneClient** as the startup project.
4. Navigate to **TreyResearch.WinPhoneClient.** Navigate to **Service Reference**. Right-click **SensorReadingServiceRef** and select **Configure Service Reference**.
5. Update the **Address** field with the new WCF service that you deployed in Exercise 2, Task 2.

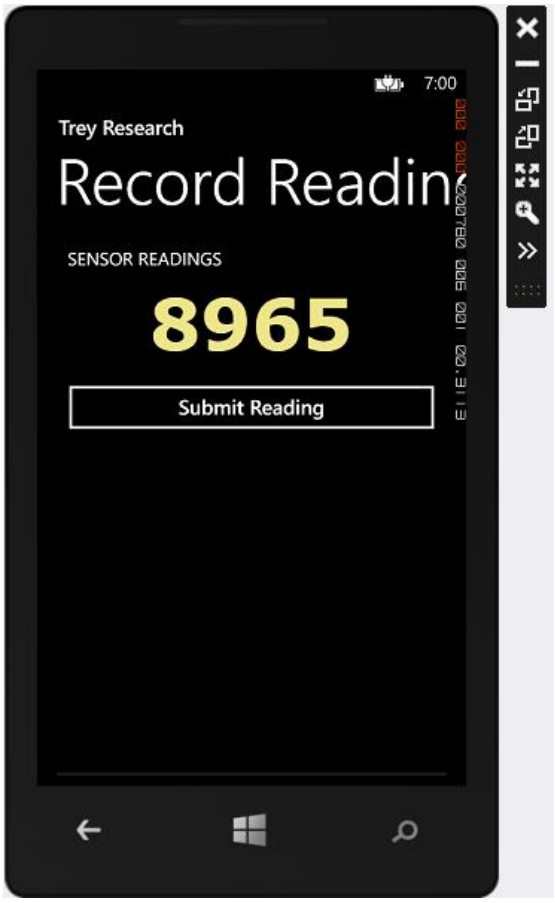


Note: Examine the **ServiceReferences.ClientConfig** files under **TreyResearch.WinPhoneClient** and make sure that the client endpoint address points to the WCF service that you deployed.

6. Save all changes. Press **F5**. The application opens in the emulator. Select **TreyResearch.WinPhoneClient**.



7. You should see a record reading. The following screen shot is an example. A display such as this means that the application works with the **TreyResearch.Cloud.WcfService** that you deployed earlier.

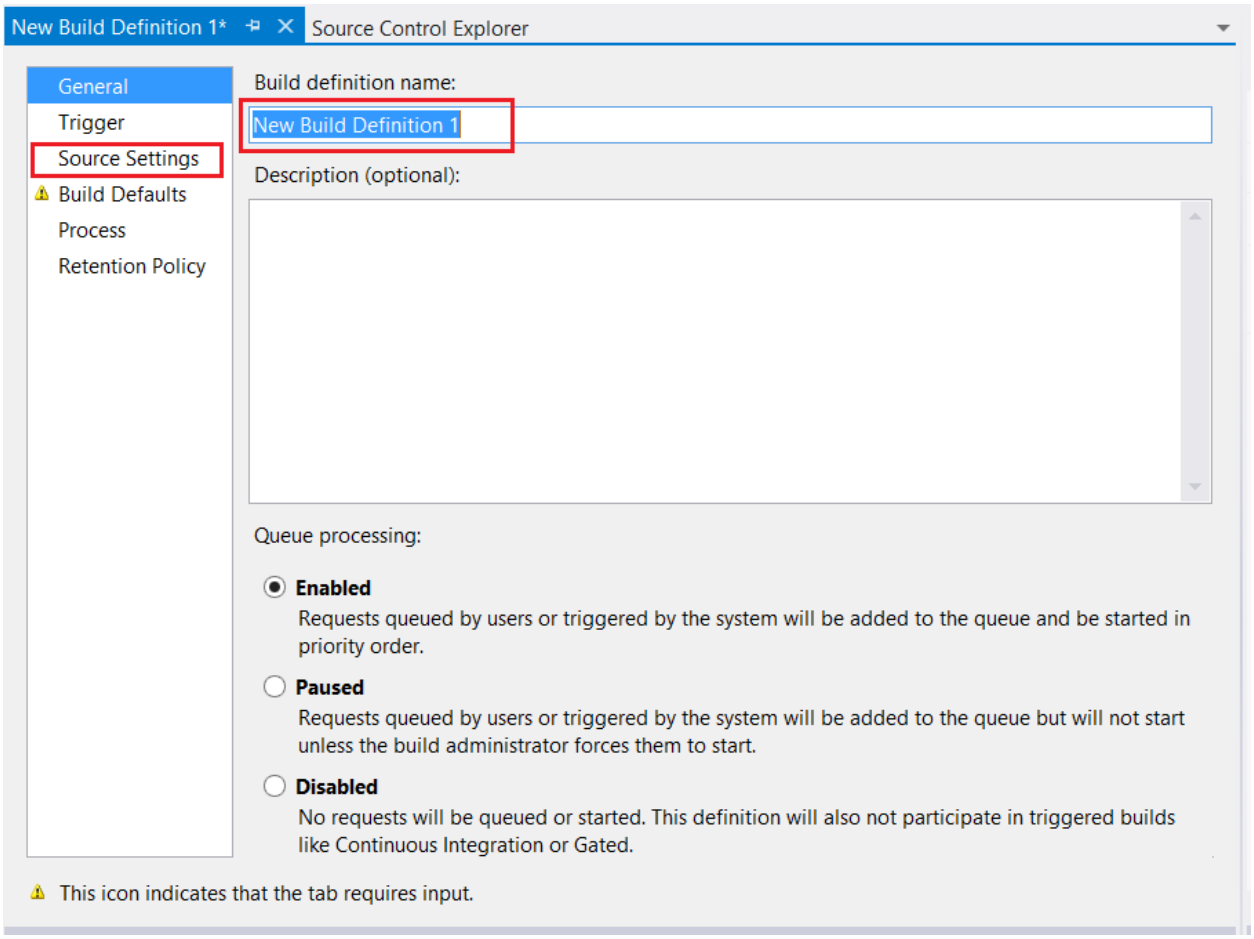


Exercise 3: Creating the Build definition for the Trey Research Pipeline

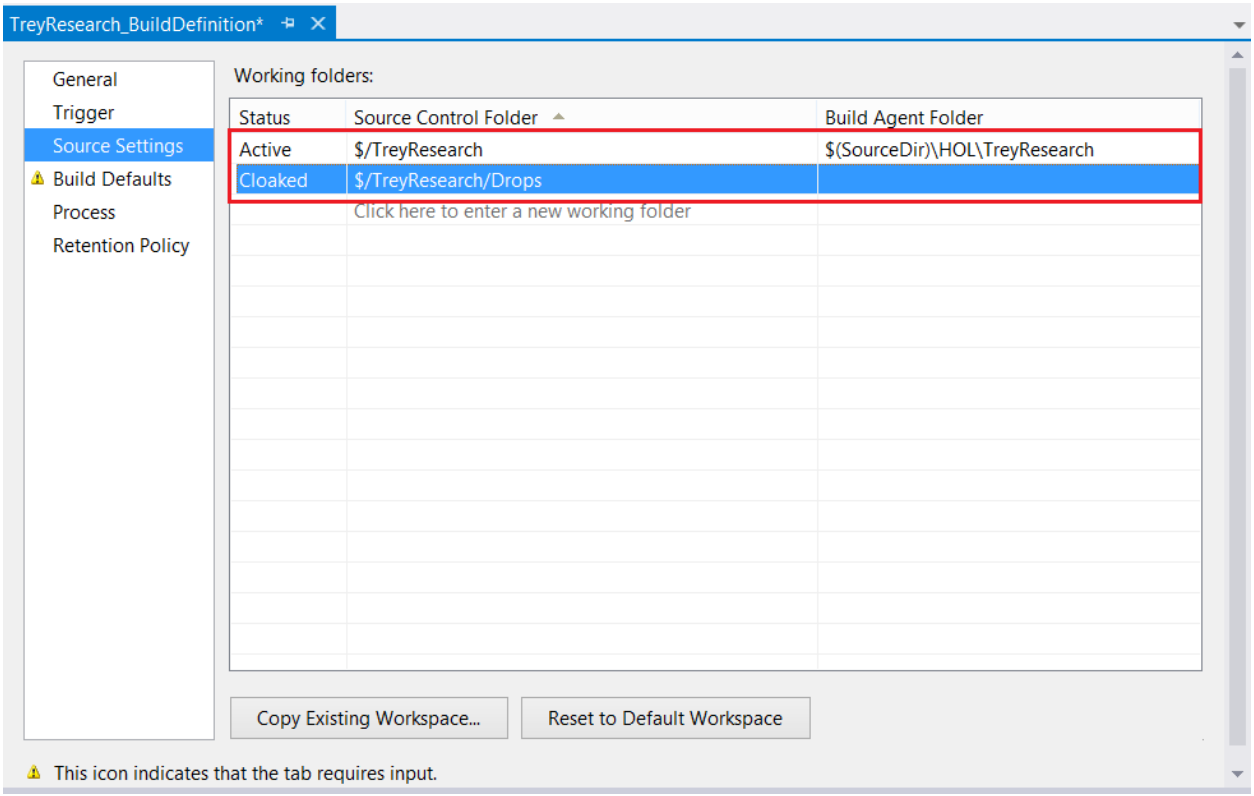
In the exercise you create a build definition for the Trey Research pipeline, which supports continuous integration. For more information about build definitions, see How To: Create a Build Definition.

Task 1: Create the Basic Build definition

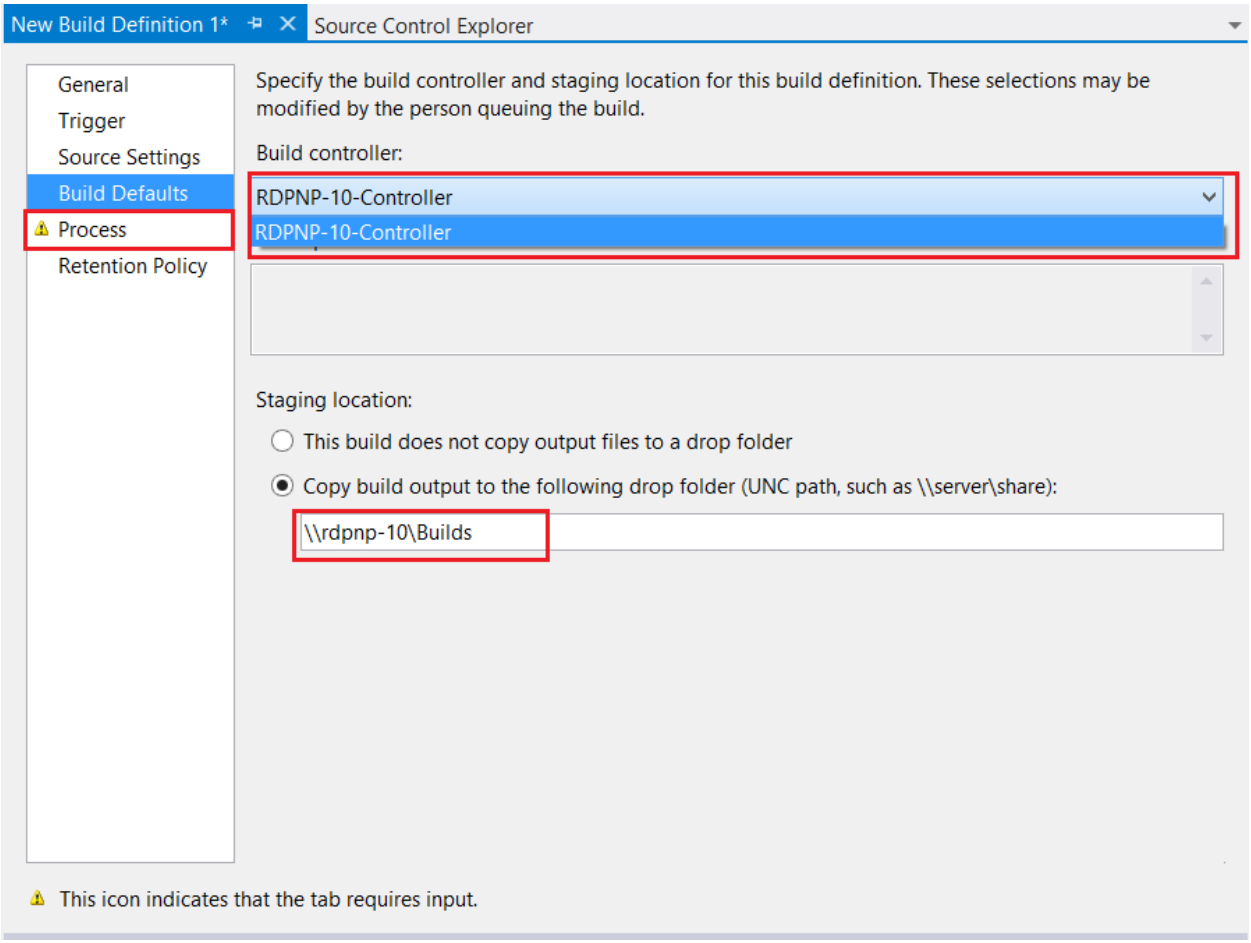
1. In Team Explorer, click **Builds**.
2. Click **New Build Definition**.
3. In the **General** tab, in the **Build definition name** text box, enter a name for the build. Click **Source Settings**.



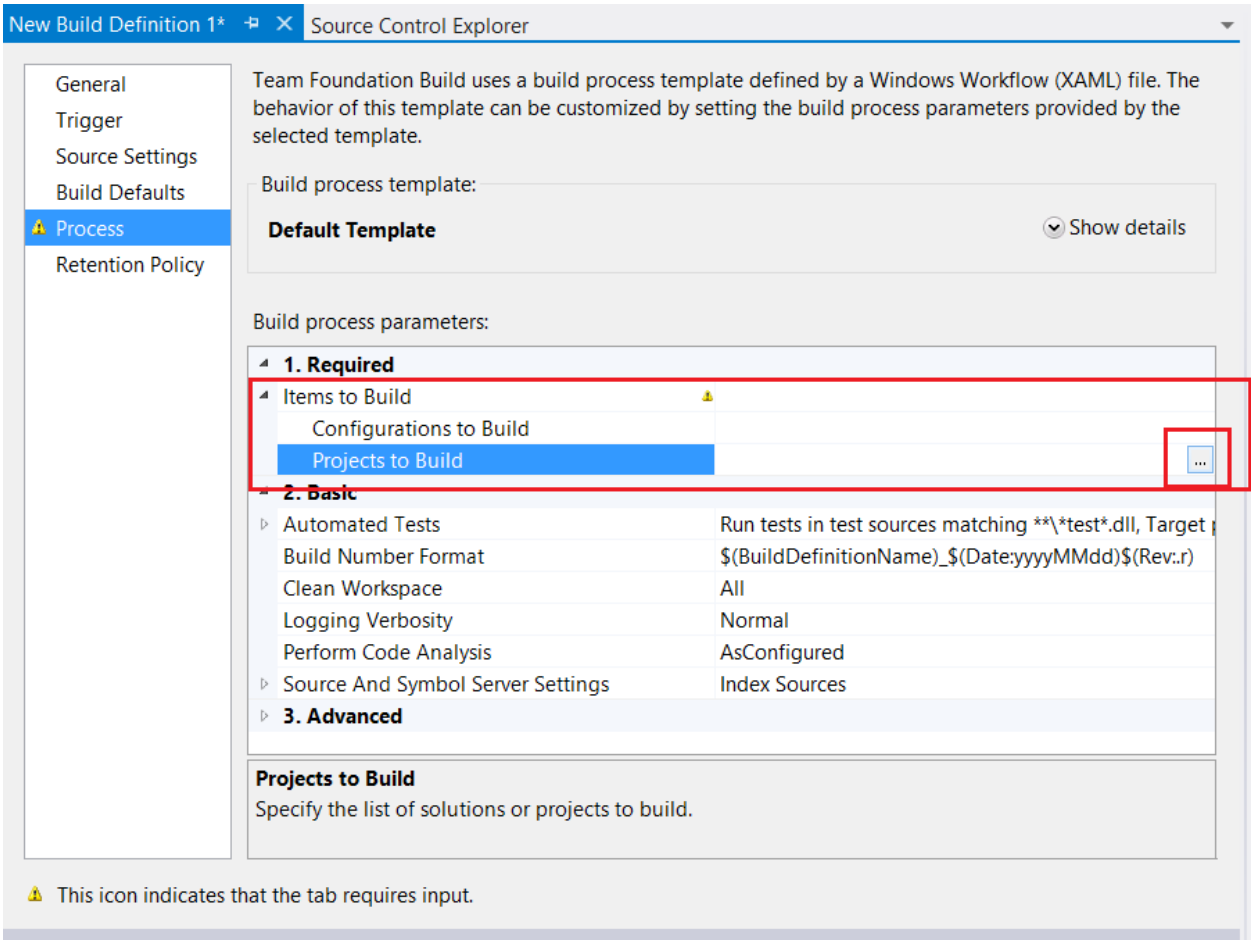
4. In **Source Settings**, check to see that the **Working folders** fields match the highlighted fields shown in the following screenshot. Click the **Build Defaults** tab.



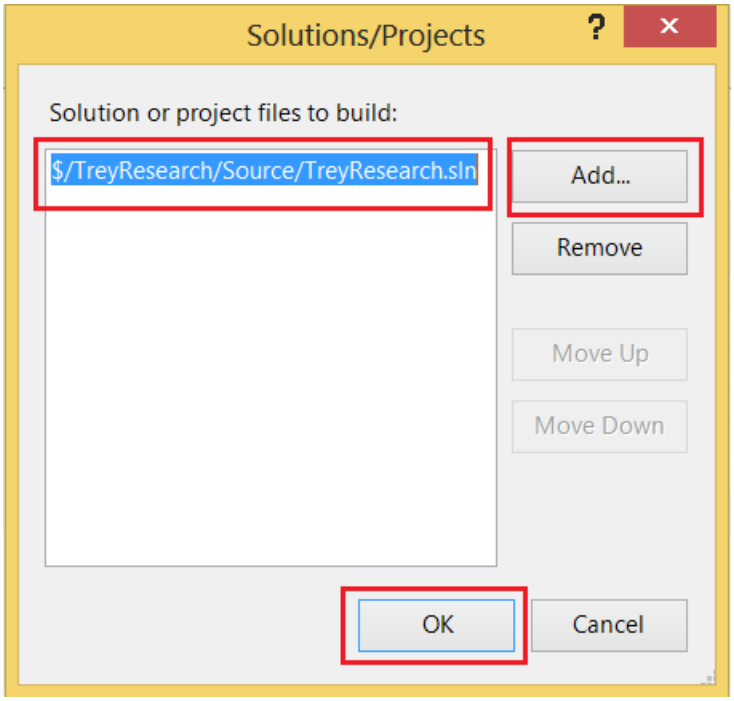
5. In **Build Defaults**, select the build controller used with TFS from the **Build controller** combo box. Under **Staging location**, enter the build directory in **the Copy build output to the following drop folder** text box. Select the **Process** tab.



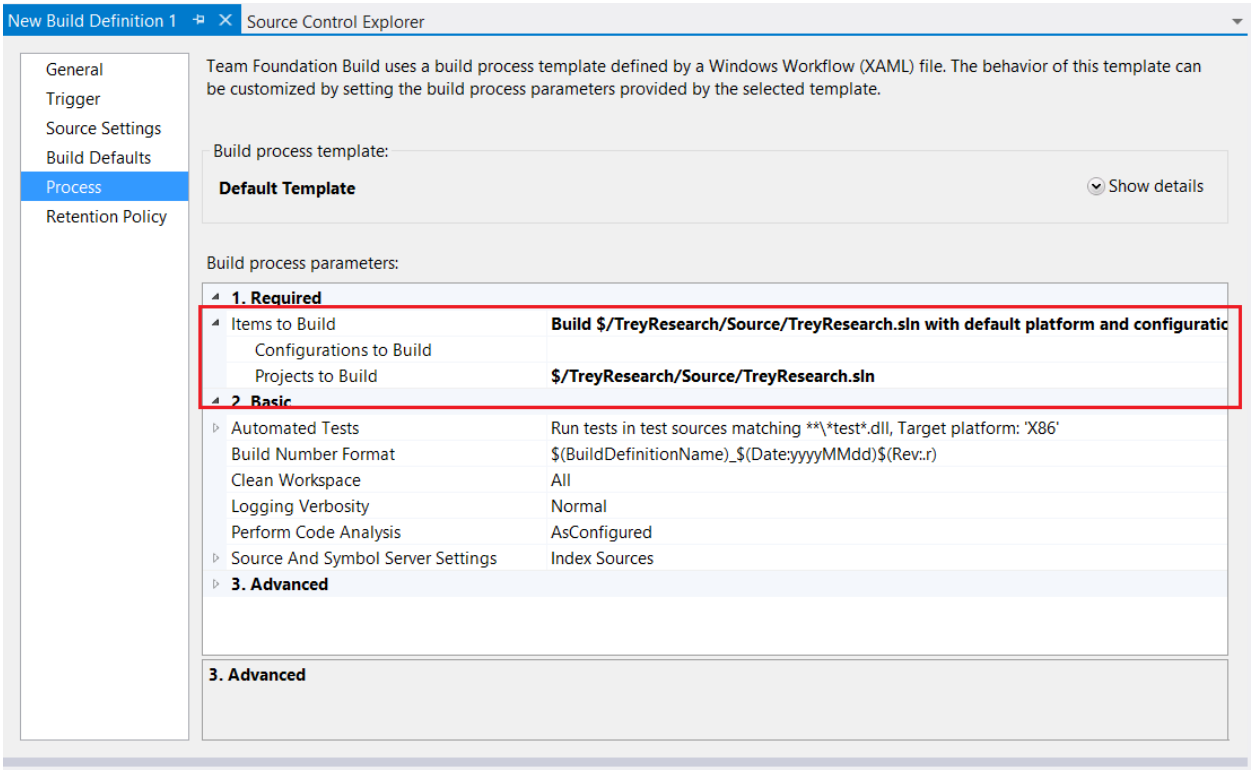
6. In the **Process** tab, under **Build process parameters**. Expand the **Items to Build** node. In the **Projects to Build** field, click the ellipses **(…)**. **The Solutions/Projects** dialog box opens.



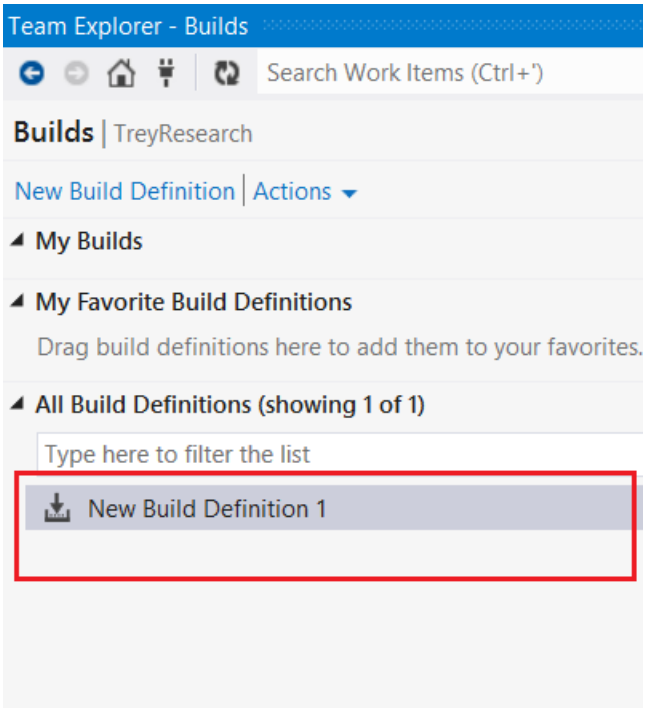
7. Click **Add** and browse to **TreyResearch.sln**. Click **OK**.



8. The **Process** tab should look like the following screenshot. Save the build.



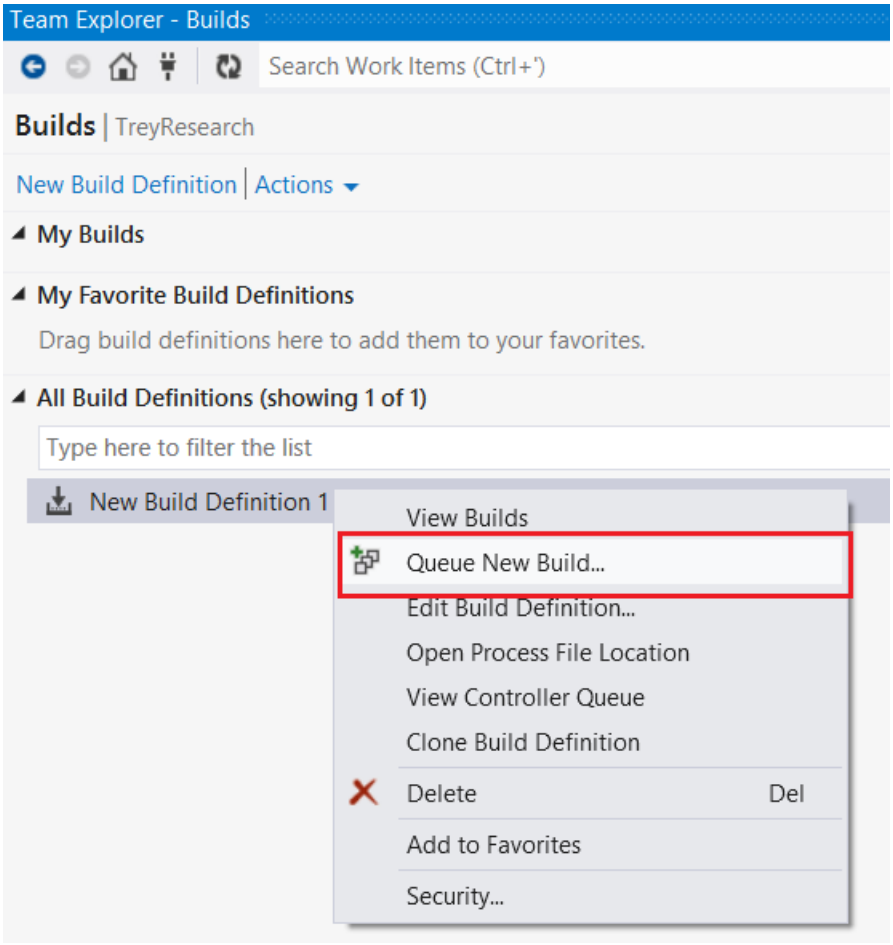
9. The new build definition will appear in the Team Explorer **Build** menu.



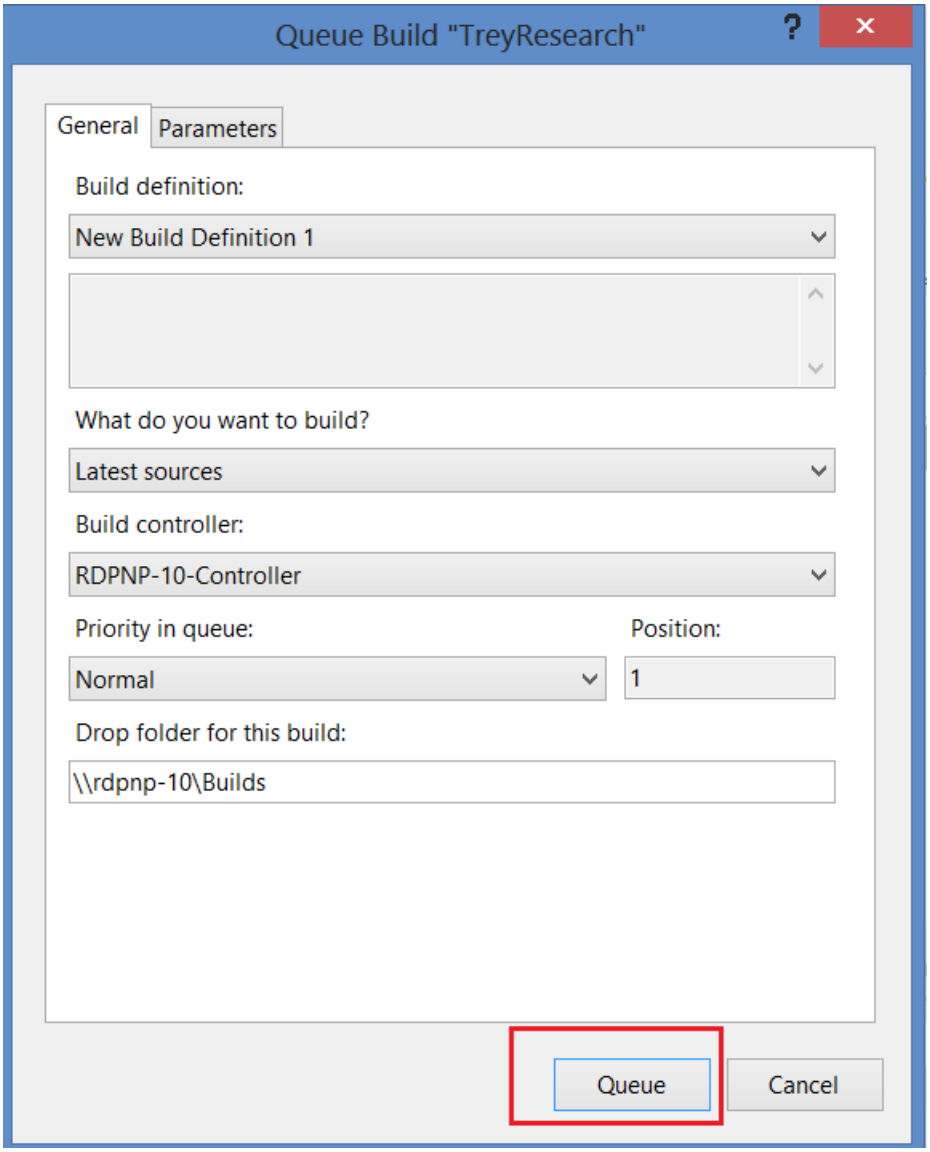
Task 2: Run the Basic Build Definition

In this task you run the build to make sure it works properly.

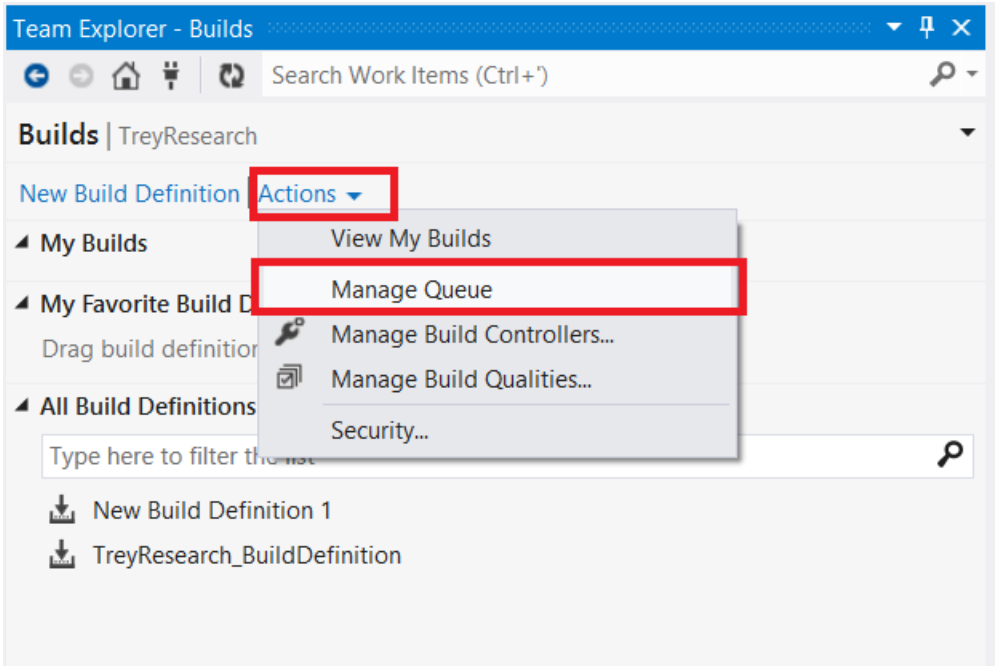
1. Queue the build by right-clicking on the new build definition. Select **Queue New Build**. This launches the **Queue Build** dialog box.



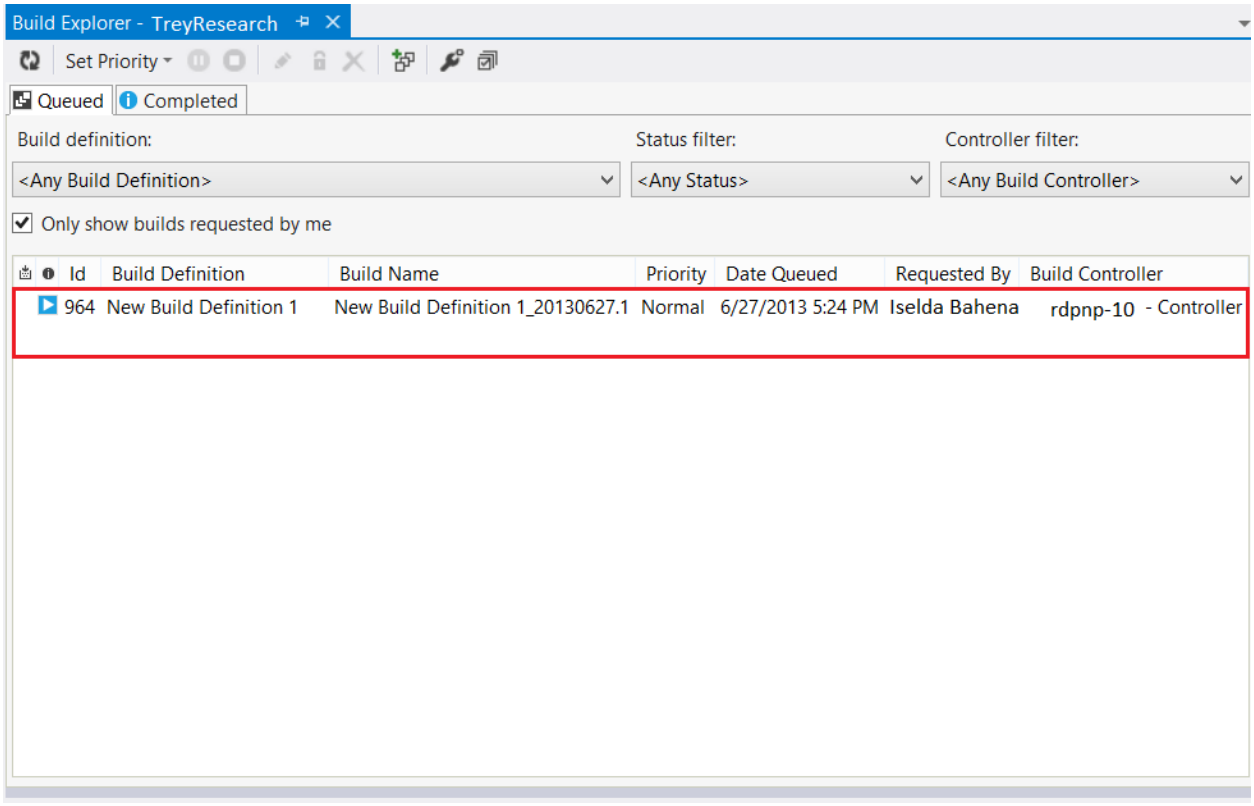
2. Leave the defaults as they are. Click **Queue**.



3. To watch the build, launch Build Explorer. Click **Actions**. Click **Manage Queue**.



4. In Build Explorer you can see both active and completed builds. The following screenshot is an example.



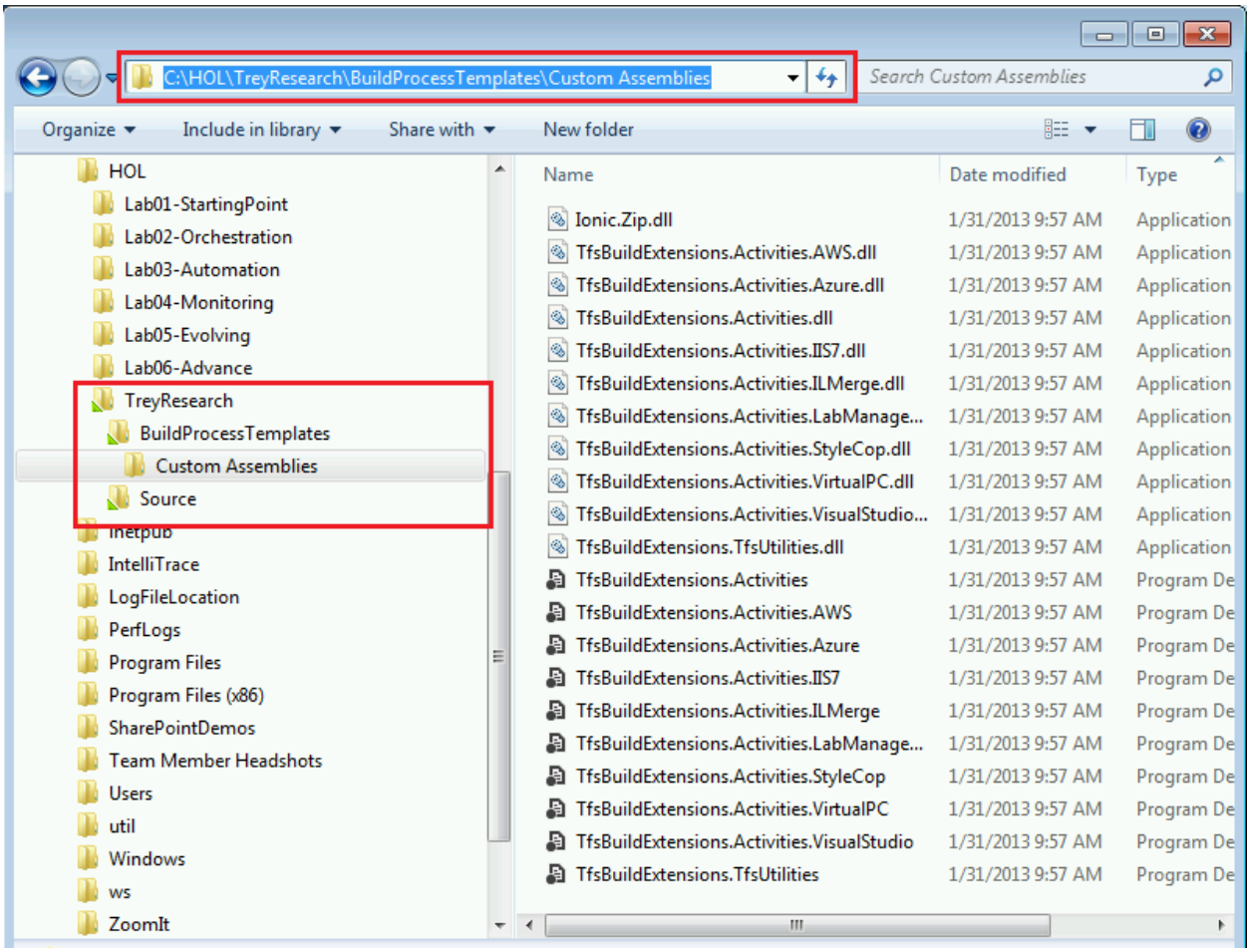
5. When the build is done it appears in both Build Explorer and the **Build** menu.

Exercise 4: Install the TFS Build Extensions

In this exercise you install the TFS Build Extensions, which you will use in the next labs. The TFS Build Extensions are on CodePlex . To use them, follow these steps.

Task 1: Copy files to Custom Assemblies folder.

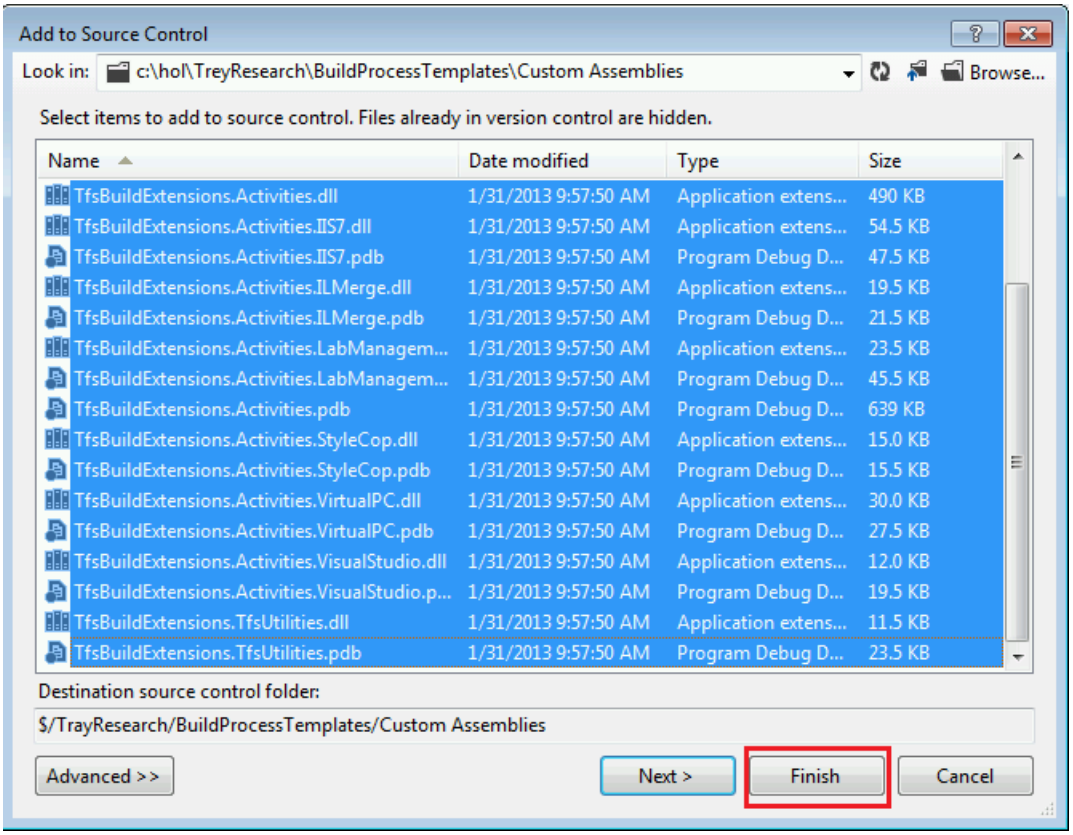
1. Unzip the file TfsBuildExtensions.zip you downloaded in Introduction document.
2. Go to your **TreyResearch** folder and locate the **BuildProcessTemplates** folder. This was created when you created the team project.
3. Create a new folder under the **BuildProcessTemplates** and name it **Custom Assemblies**.
4. Copy all the assemblies located in the unzipped folder **C:\<yourlocation>\ TfsBuildExtensions January 2013\TfsBuildExtensions January 2013\Code Activities\VS2012** to the new **Custom Assemblies** folder.
5. The following screenshot shows how the new folder should appear.



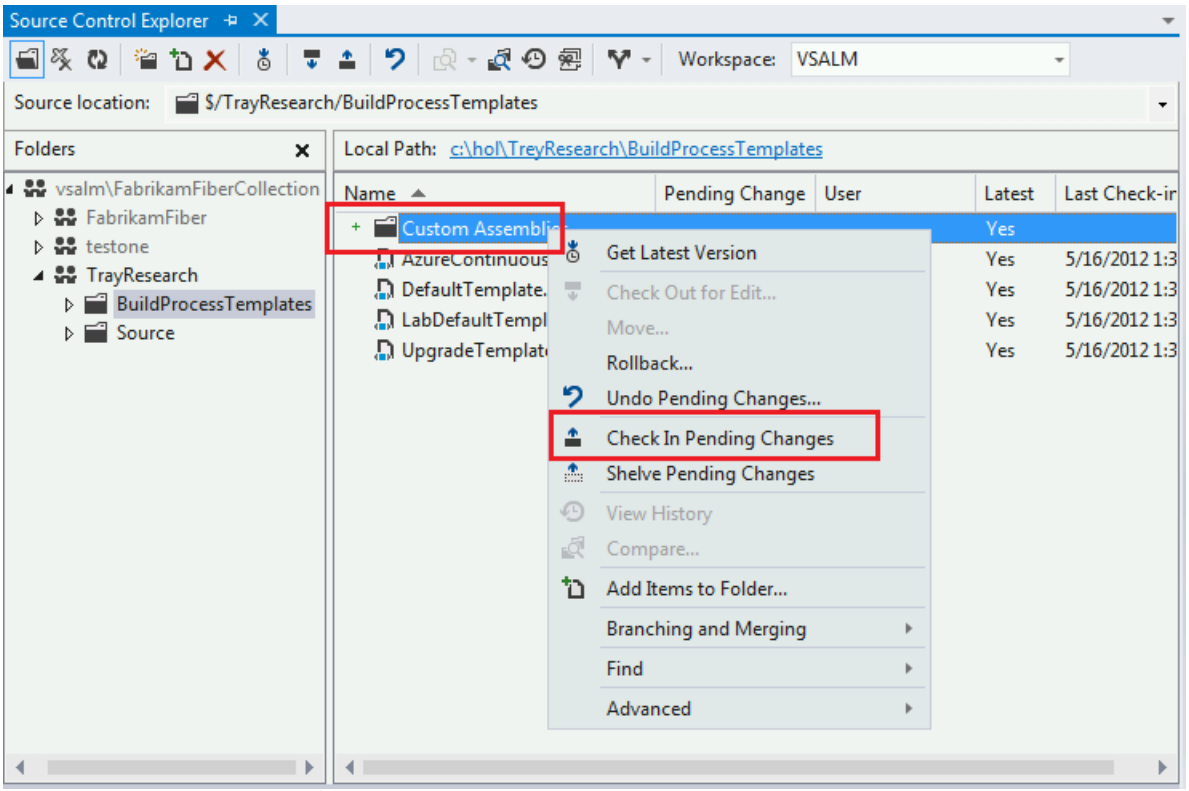
Task 2: Add Custom Assemblies Files to TFS

In this task, you add the custom assemblies files to your TreyResearch Solution.

1. In Solution Explorer, navigate to the team project, **TreyResearch**.
2. Right-click on the **BuildProcessTemplates** folder and select **Add Items to Folder**. The **Add to Source Control** dialog comes up.
3. Select the **Custom Assemblies** folder. All the files in all the assemblies appear. Select them all. Click **Finish**.



4. To check in the files, right-click the **Custom Assemblies** folder and select **Check In Pending Changes**.

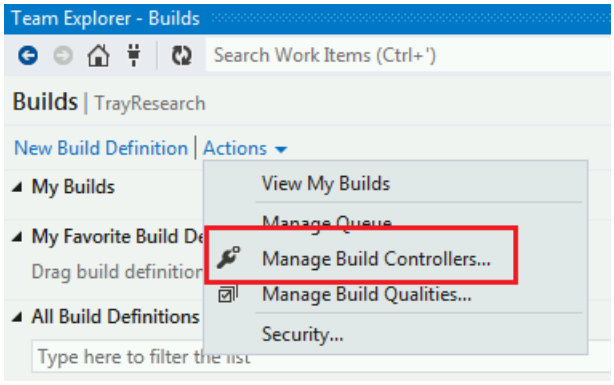


5. In Team Explorer the **Pending Changes** dialog box appears. Click **Check In**. The check-In confirmation message appears. Click **Ok**.

Task 3: Check that Version Control for the Build Controller is Set to the Custom Assemblies

In this task you make sure that the build controller points to the custom assemblies.

1. In Team Explorer, click **Builds**.
2. Under the **Actions** drop down menu, select **Manage Build Controller**s. The Build Manager dialog box appears.



3. In the Manage Build Controllers, click on the Properties… button.

4. In the **Properties** dialog box, make sure the **Version control path to custom assemblies** points to the custom assemblies folder that you checked in Task 2.

Exercise 5: Setting Up the Target Environments

In this exercise you set up the target environments so that they can host the WCF services and so that they can be managed by the pipeline.

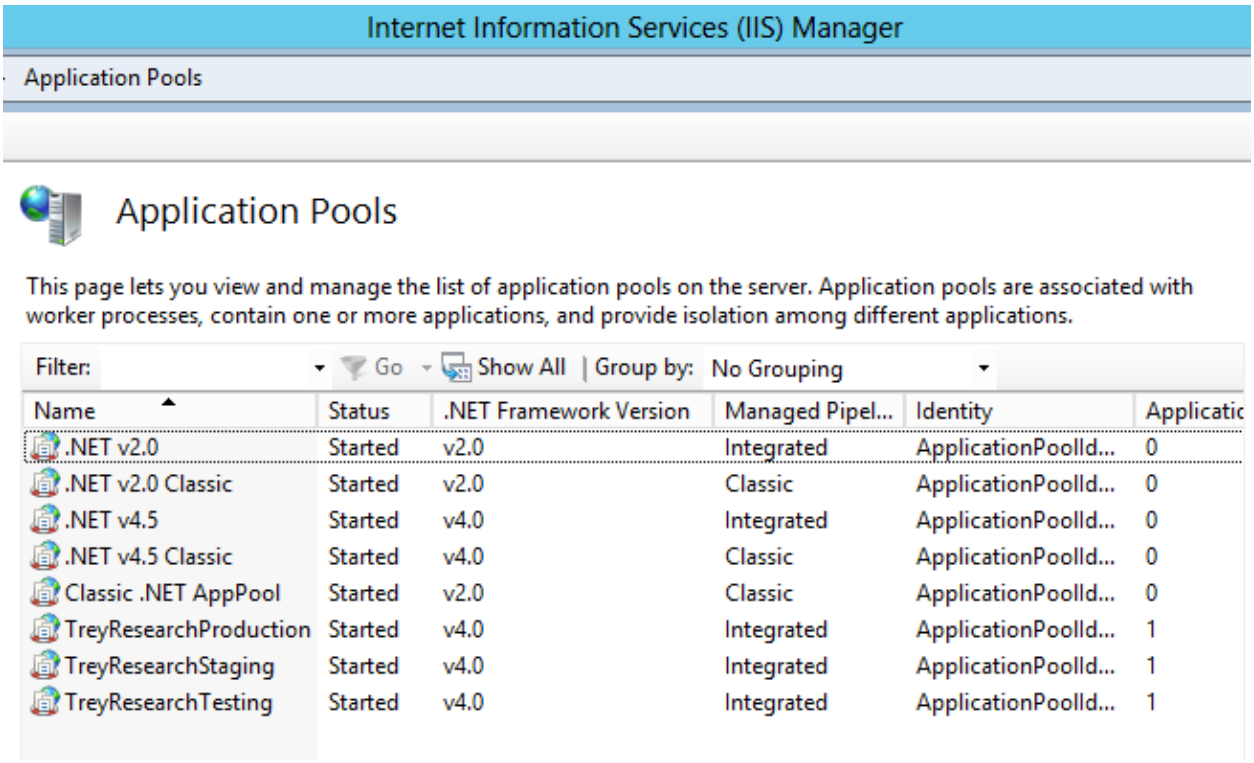
The pipeline has four environments: development, testing, staging, and production. Because the development environment is isolated and only exists on development machines, it doesn't host any services. However, the three other environments host the WCF service, so each requires an IIS website.

Typically, these three websites would reside on three different IIS servers across different computers. For simplicity, this lab creates three different websites on the same IIS server.

Task 1: Set up the Web Server

In this task you set up the computer that acts as the web server. This task assumes that IIS is already installed.

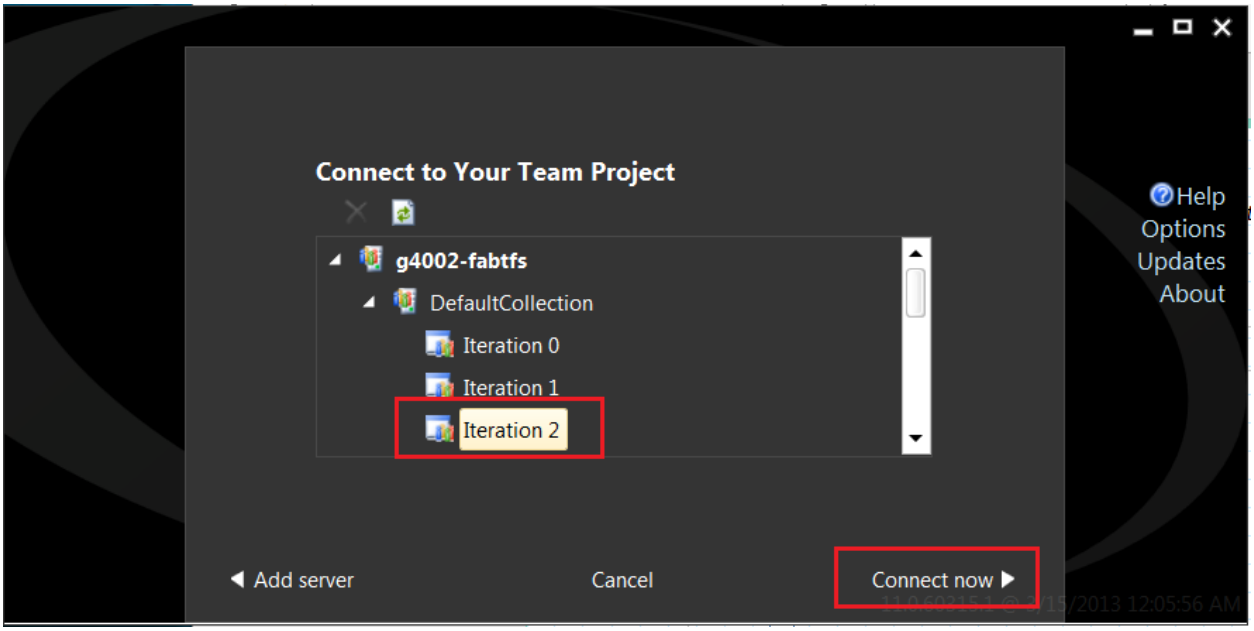
1. Make sure that all the roles and features required to host WCF 4.5 services are installed. The minimum requirements are that everything listed under the following sections is setup under the roles and features settings. http://technet.microsoft.com/en-us/library/cc754523(v=WS.10).aspx.



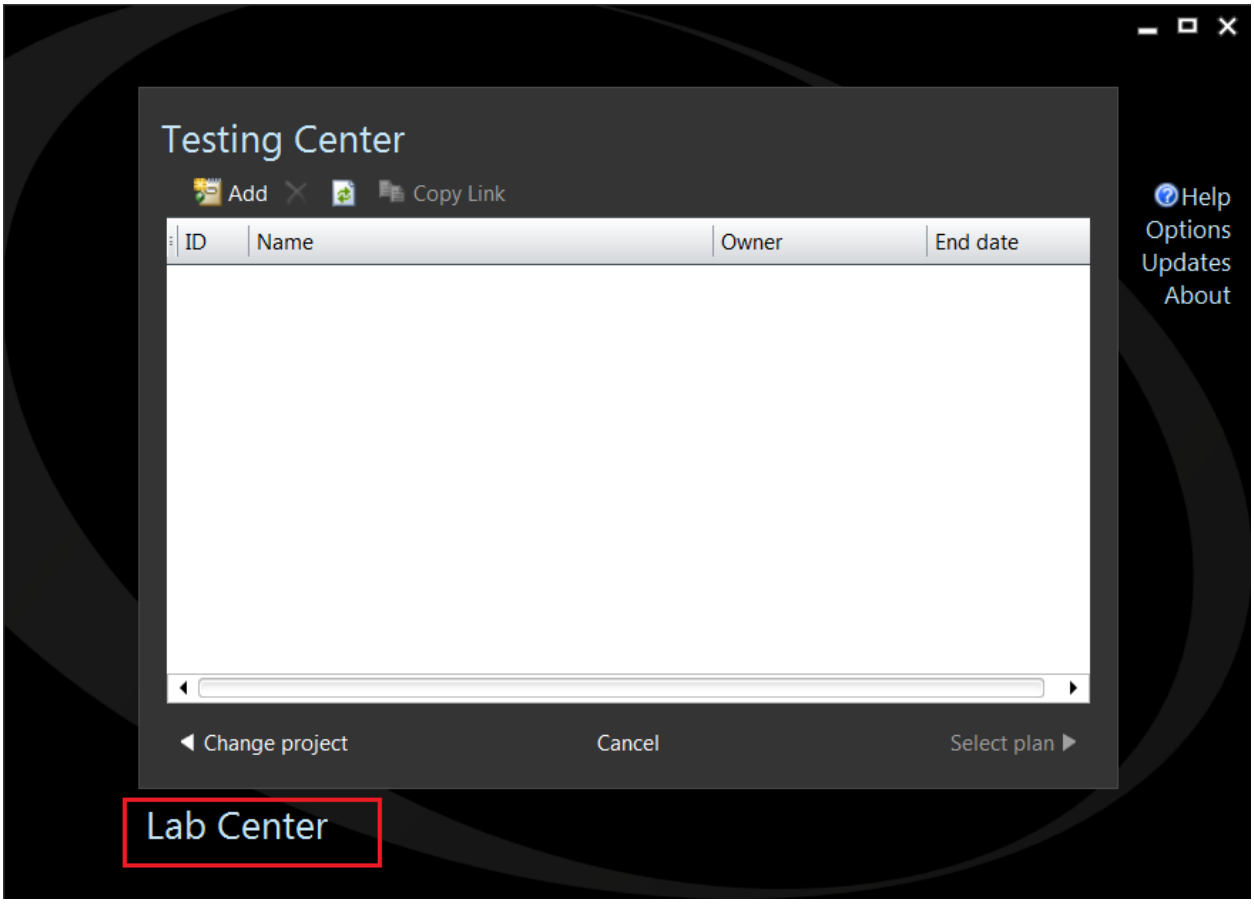
Task 2: Set Up the Environment in Lab Management

In this task you add the web server to a Lab Management environment. The pipeline uses Lab Management to manage the environments and for automation. Lab Management will automatically install the test agents on the target computers.

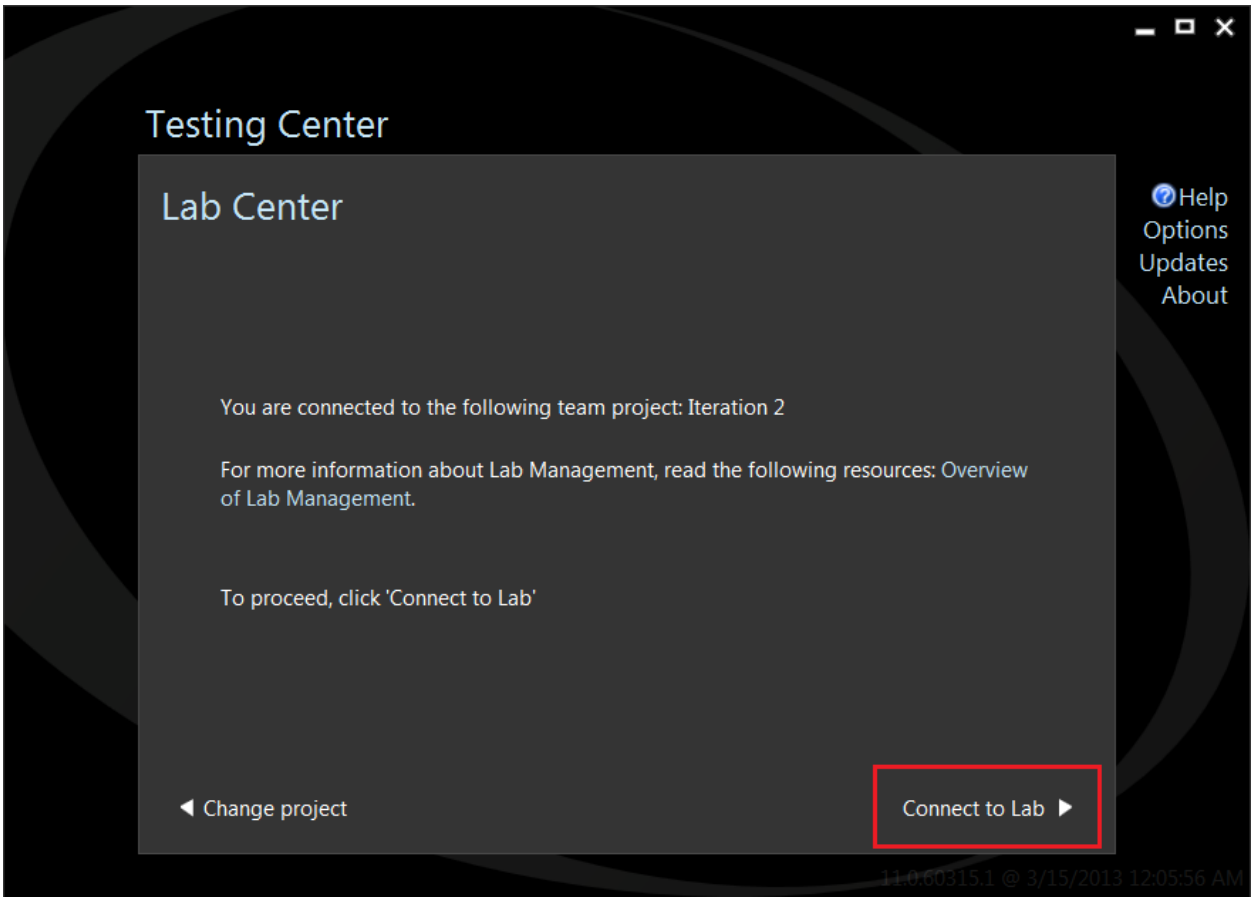
1. Open Microsoft Test Manager.
2. Select the TFS computer that stores the team project that contains the build definitions for the pipeline orchestration. Select **Connect Now**.



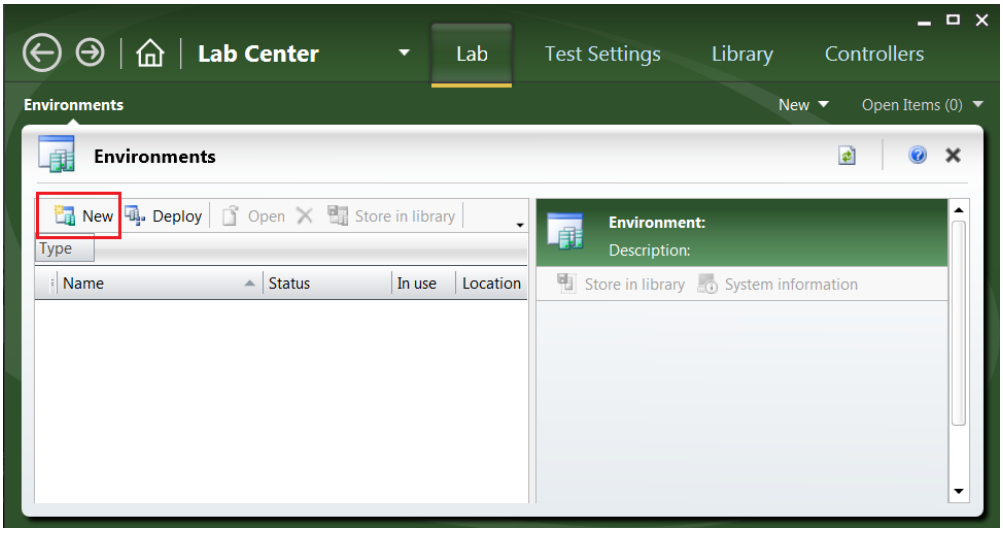
3. The Testing Center page opens. Select **Lab Center** at the bottom of the page.



4. Click **Connect to Lab** on the Lab Center introduction page.

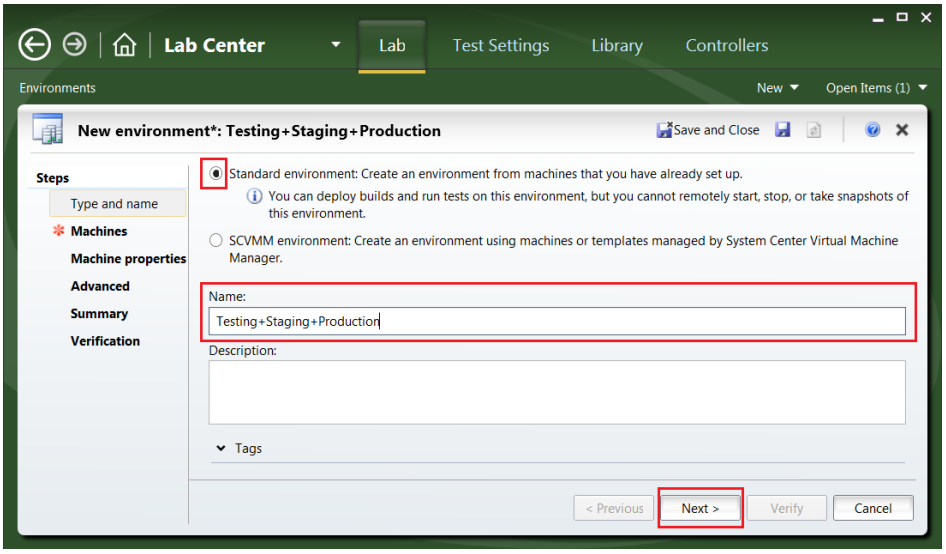


5. The Lab Center panel appears with the **Lab** tab selected. You should see the **Environments** pane. Click **New**.



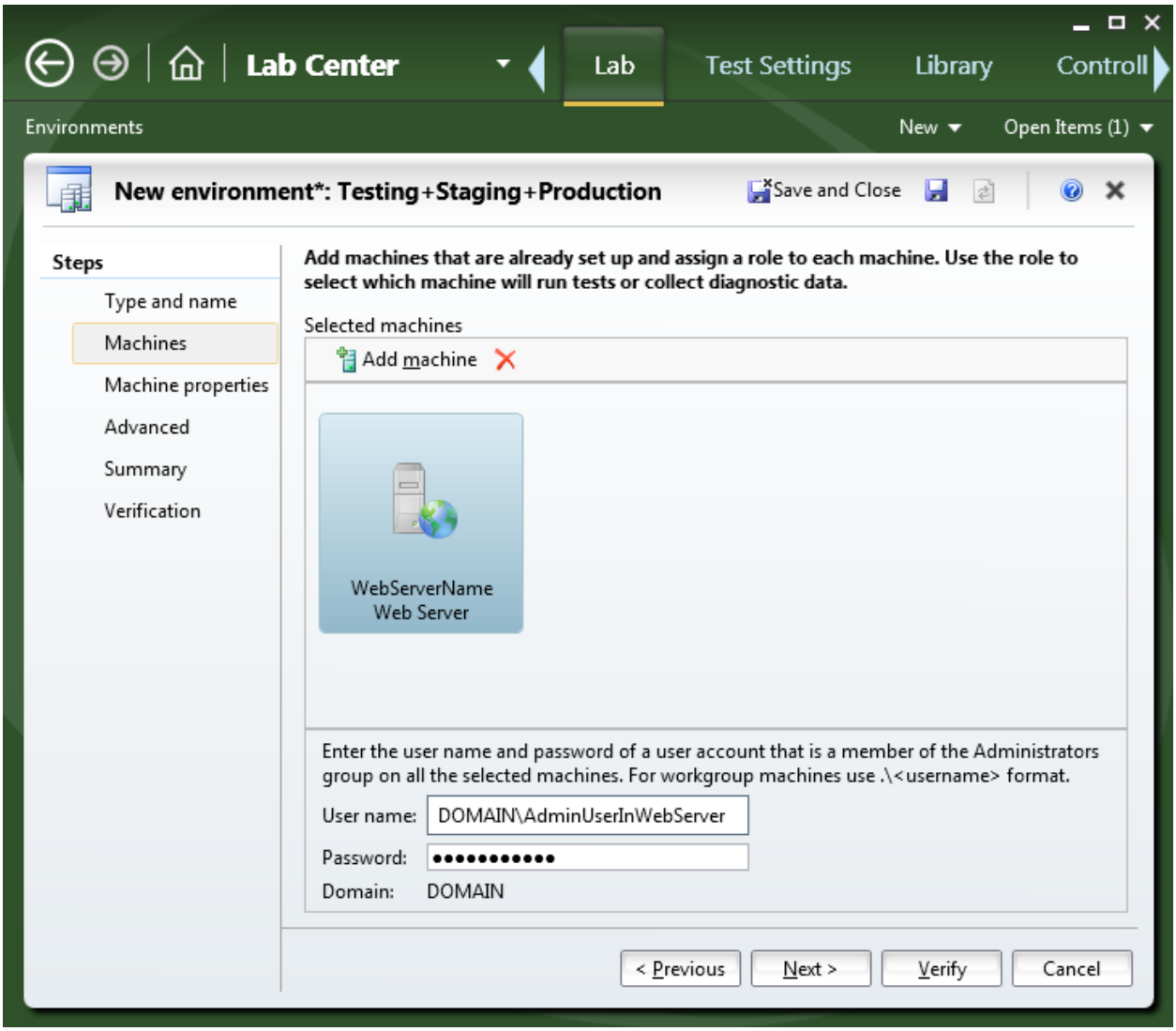
6. The **New Environment** wizard appears. The **Type and name** tab is selected. Select **Standard environment**.

7. In the **Name** text box, provide a meaningful name for the environment. The example shown in the screenshot is **Testing+Staging+Production** because the same machine is used for all three environments. Click **Next**.

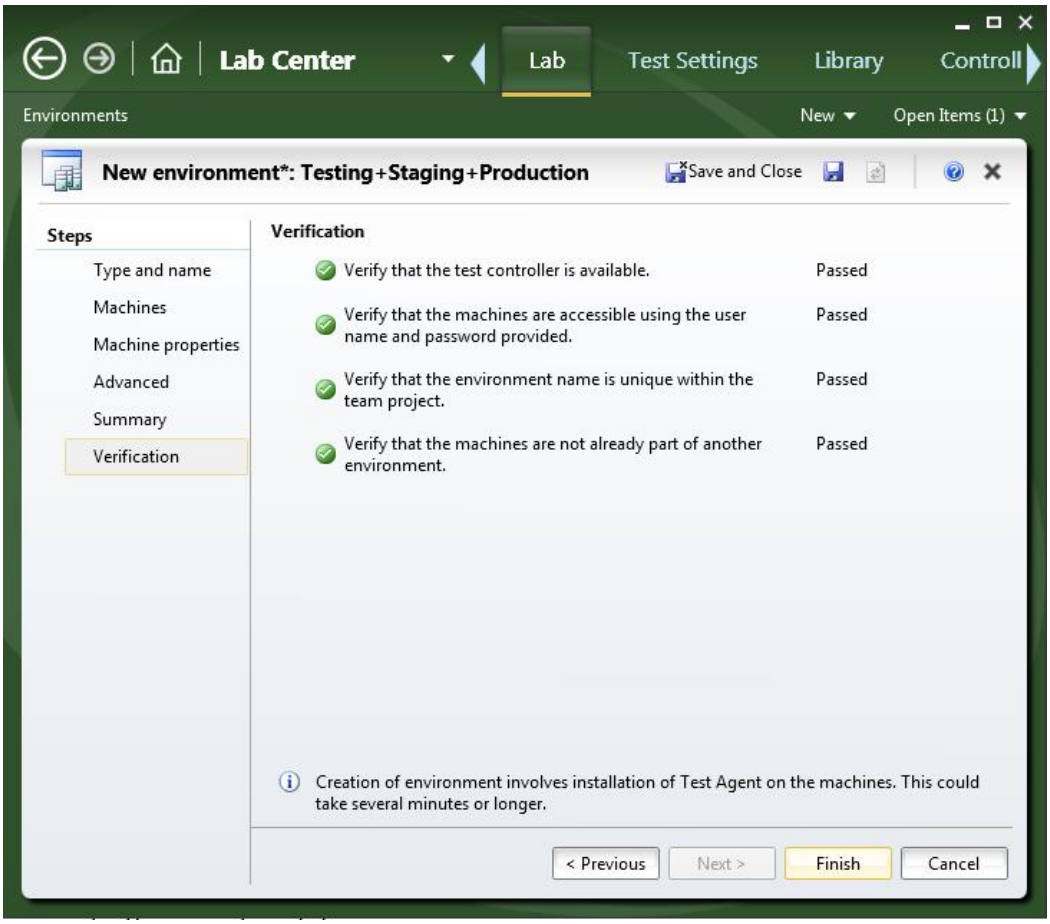


8. The **Machines** tab is selected. Click **Add Machine** and select the web server that you set up in Task 1.
9. In the **Computer name** field, provide either the NetBIOS or DNS name.
10. In the **Type Role** field enter **Web Server**.
11. In the **User name and Password** fields, enter the credentials of someone who is an administrator of the machine. Lab Management uses these credentials to install the agents that will run the automated deployments and tests.

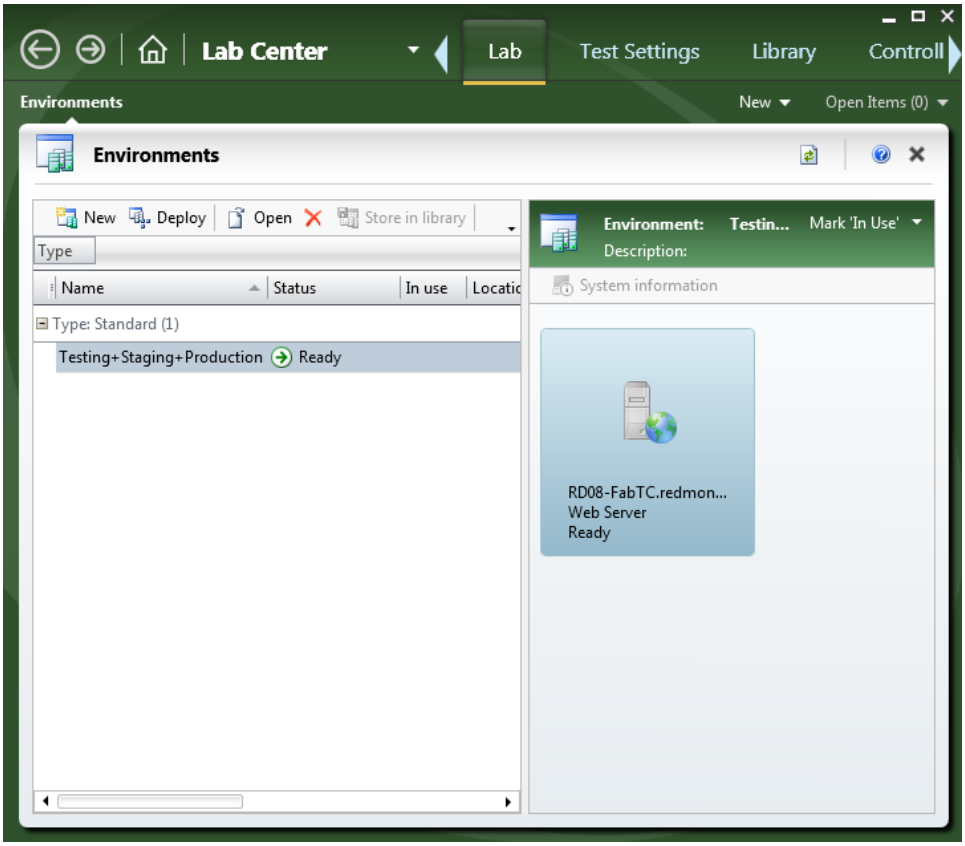
> **NOTE**: This is not the same as the user who runs the agents. The user who runs the agents and performs the deployments and tests does not have to be an administrator, and is the user specified when you configured the test controller.



12. Click **Verify**.
13. If the verification succeeds, click **Finish**.



14. After a few moments, the agents are installed on the web server and the environment is available to the pipeline. The following screenshot shows an example of an environment.

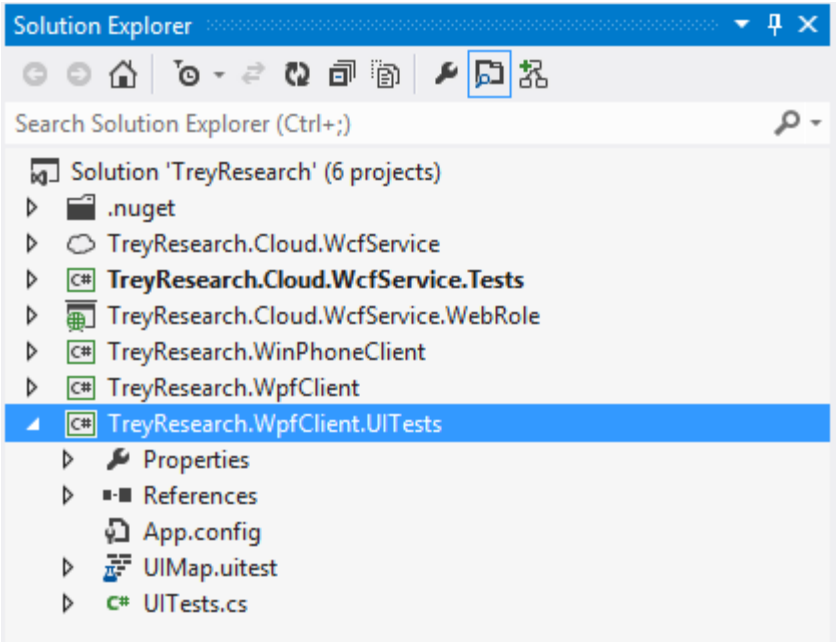


Exercise 6: Adding UI Tests for the Advanced Lab

In this exercise you add the UI tests to the Trey Research solution in order to deploy and run them automatically.

Task 1: Adding UI Tests for Advanced Lab

1. Copy the entire **TreyResearch.WpfClient.UITests** folder from **C:\HOL\Lab03-Automation\Completed-Lab\Advanced** to **C:\HOL\TreyReasearch\Source**.
2. Open Visual Studio and right click on TreyResearch Solution and choose **Add->Existing Project**. Browse to the **C:\HOL\TreyResearch\Source\TreyResearch.WpfClient.UITests\TreyResearch.WpfClient.UITests**.
3. Click **Add**. The final structure of the solution should look like the one below.



4. Check in the pending changes.

Exercise 7: Installing the Test Artifacts (Optional)

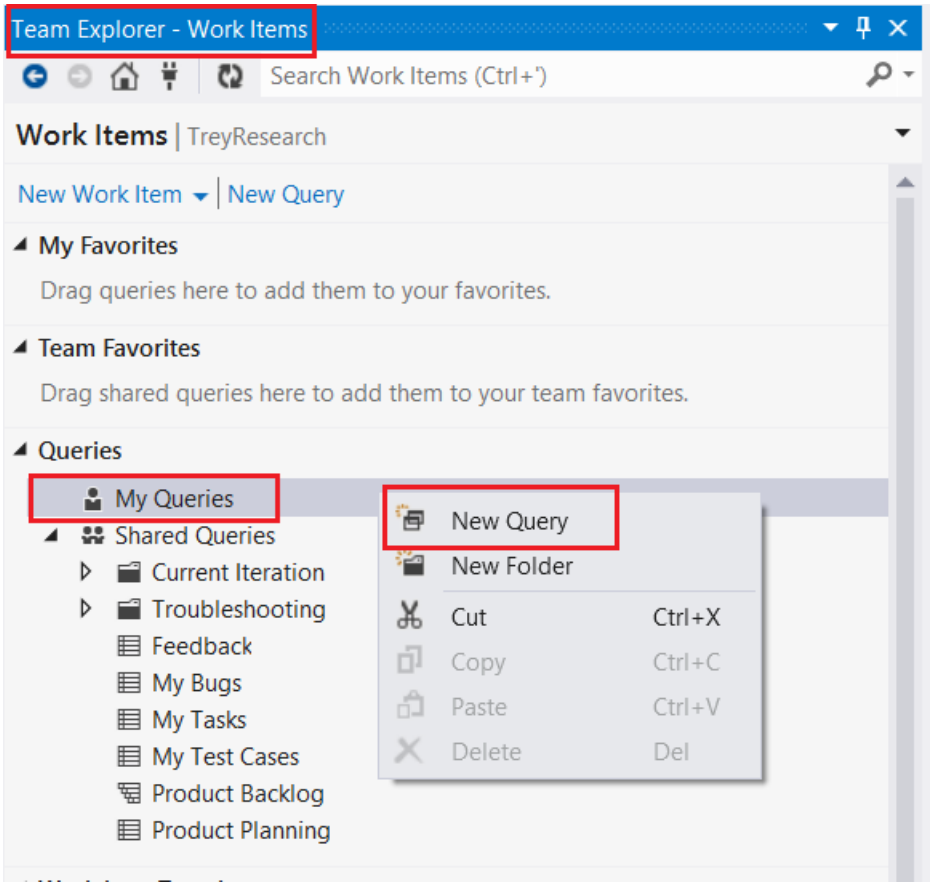
This exercise is optional, but it will help you to understand the automation and how MTM works with the pipeline. For more context, see Lab 1, Exercise 7.

In this exercise you install the work item types (WIT) by copying items stored in a Microsoft Excel spreadsheet to a TFS Excel work form. If you're unfamiliar with TFS WITs, see How To: Create, Share, and Run Work Item Queries (Team System Web Access).

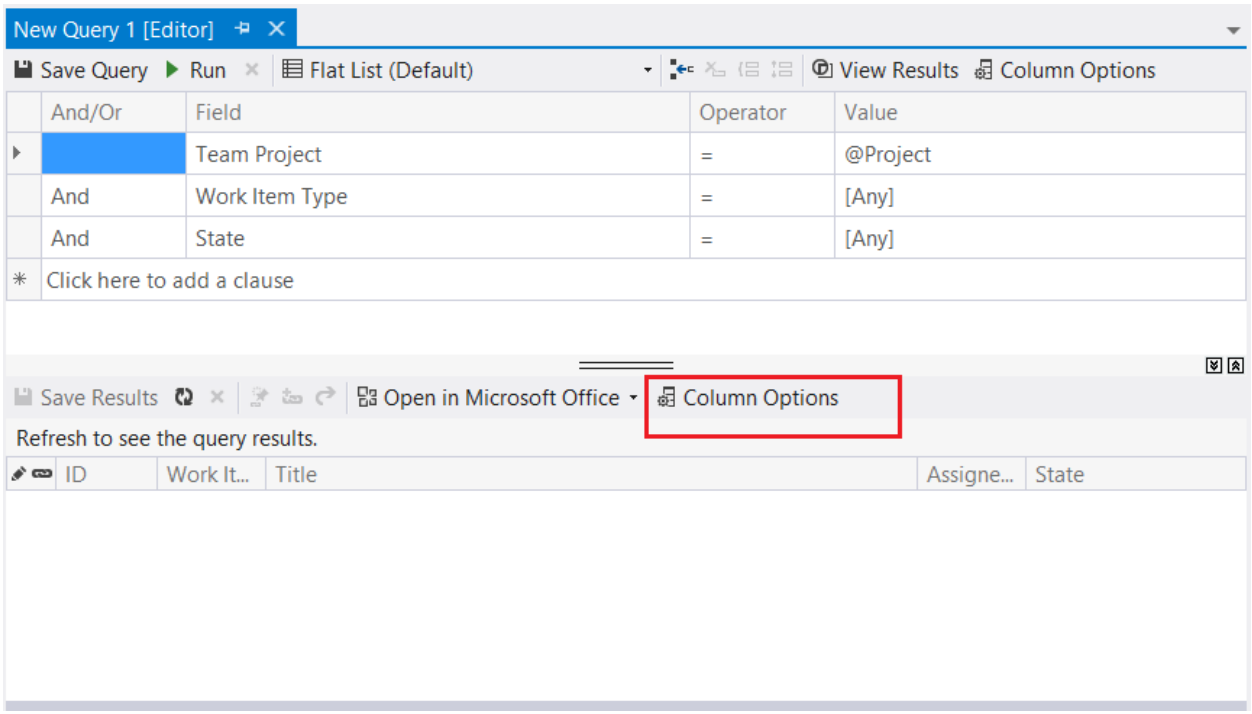
Task 1: Create an Excel TFS Work Items Types Excel Form

In this task you create a TFS Excel form that is used to organize the WITs. This is where you will store the items that are contained in the Excel spreadsheet Lab01_WIT.xlsl.

1. In Visual Studio Team Explorer, open the **Work Items** dialog. Right-click **My Queries** and select **New Query.**

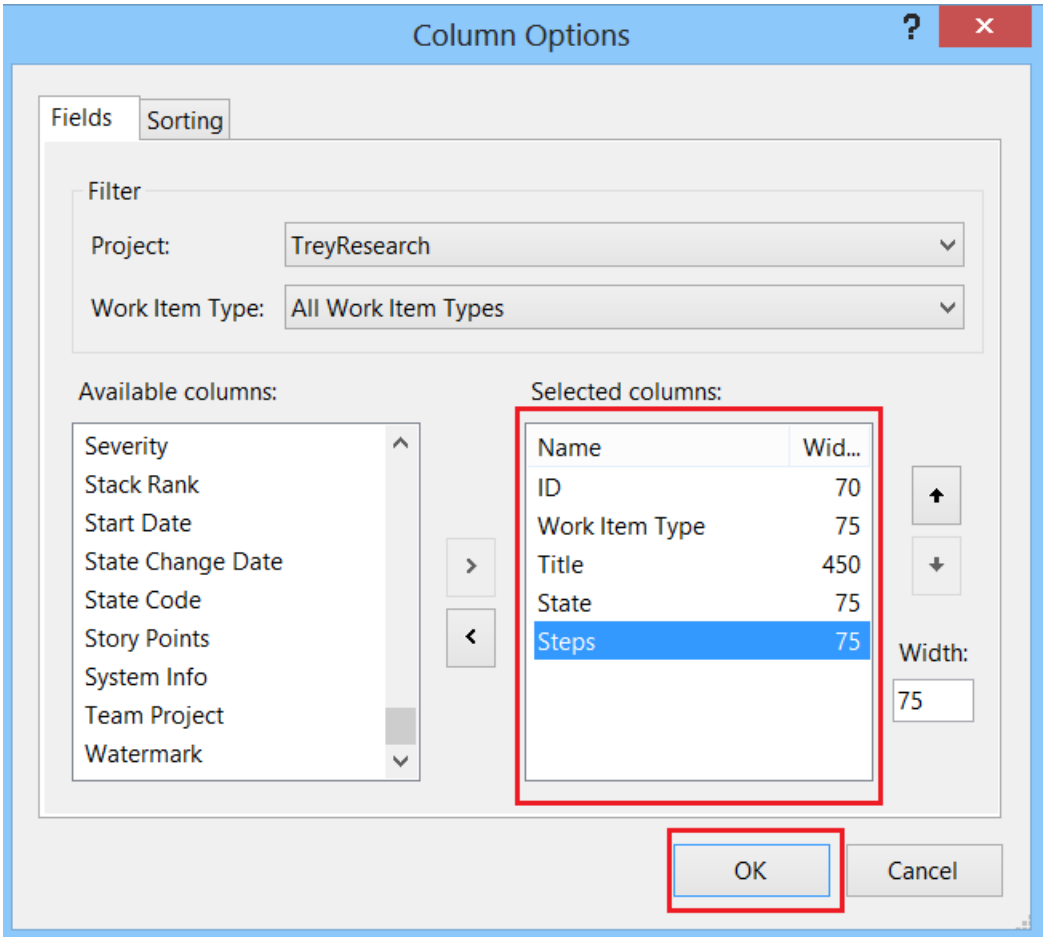


2. The **Query Editor** appears. Click **Column Option**s.

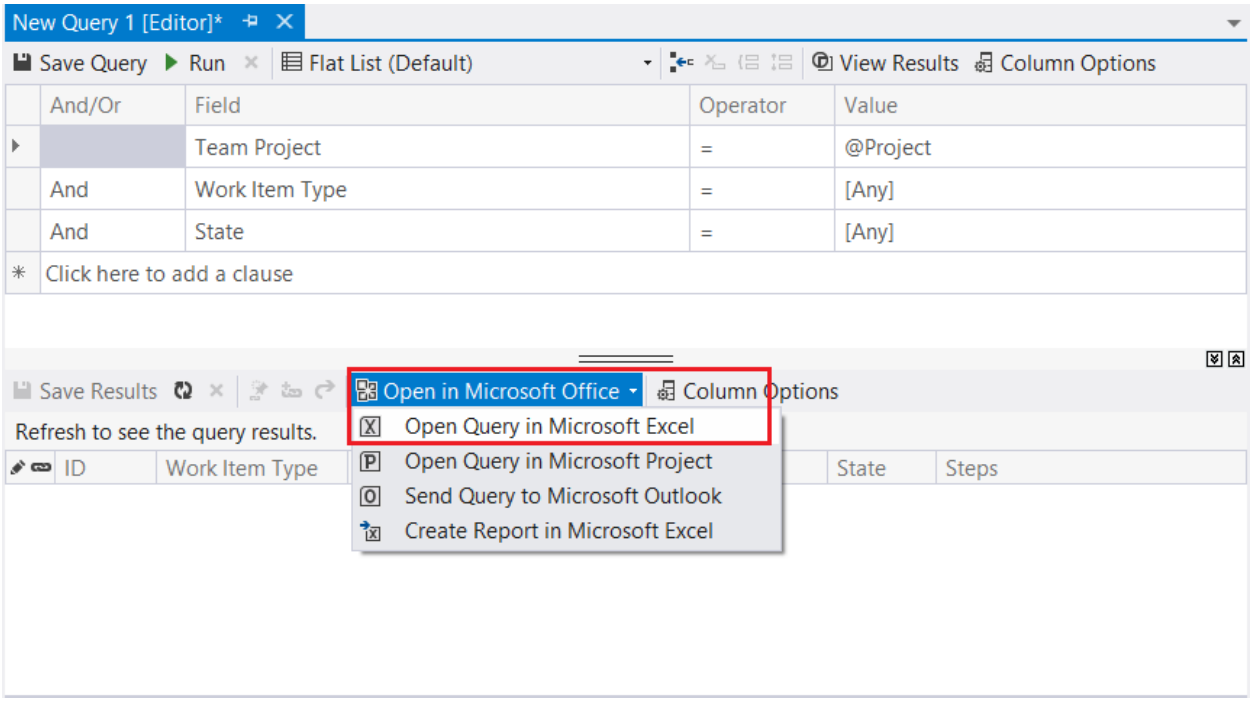


3. Select the following columns in the order listed below. When you are done, click **OK**.

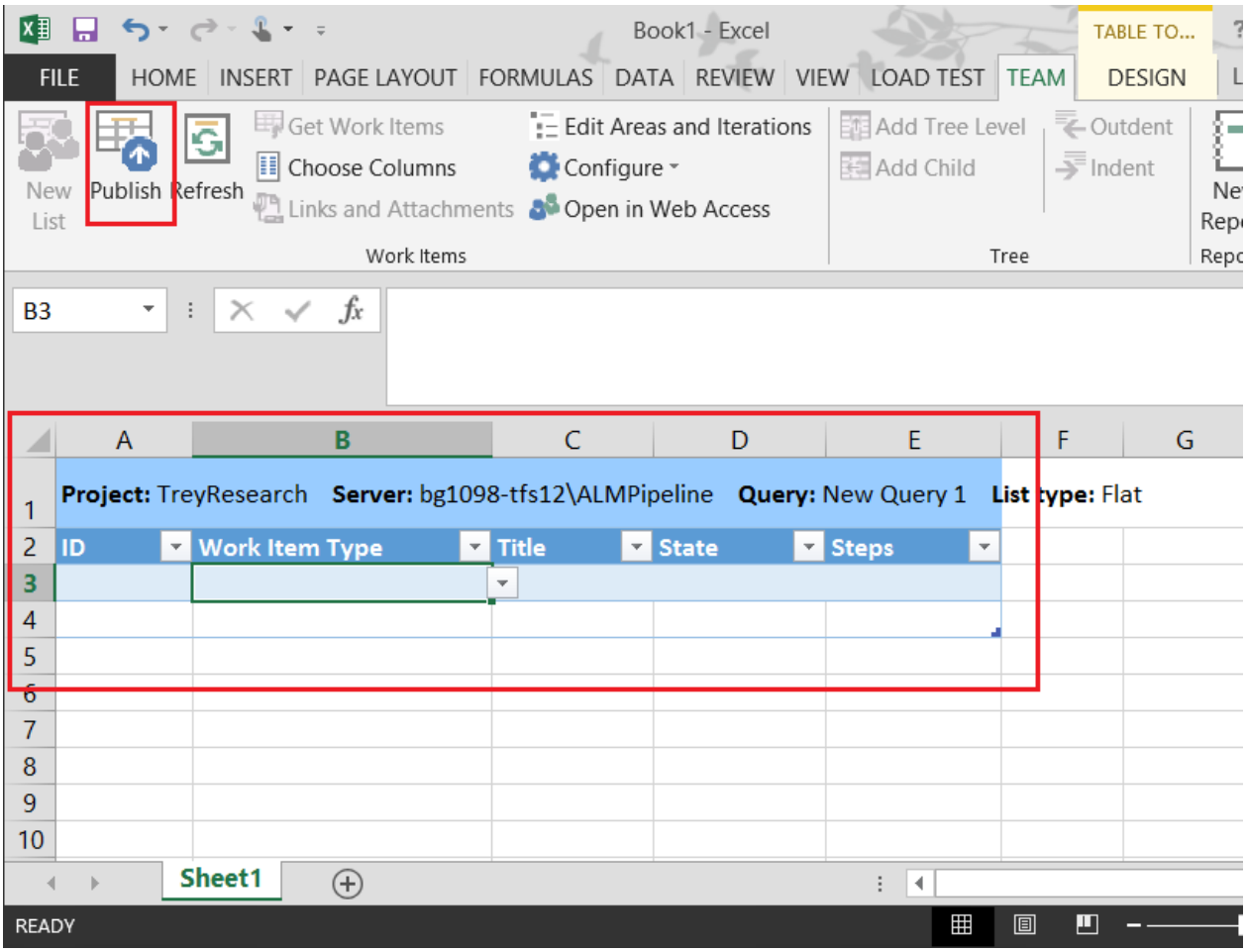
- ID
- Work Item Type
- Title
- State
- Steps



4. Click **Save Query**. The Query Editor should now show the selected columns. Locate the **Open in Microsoft Office** combo box and select **Open Query in Microsoft Excel**.



5. Excel opens a form editor with the columns you selected. Note the **Publish** button. This will publish the data back to TFS once you add the data to the spread sheet.



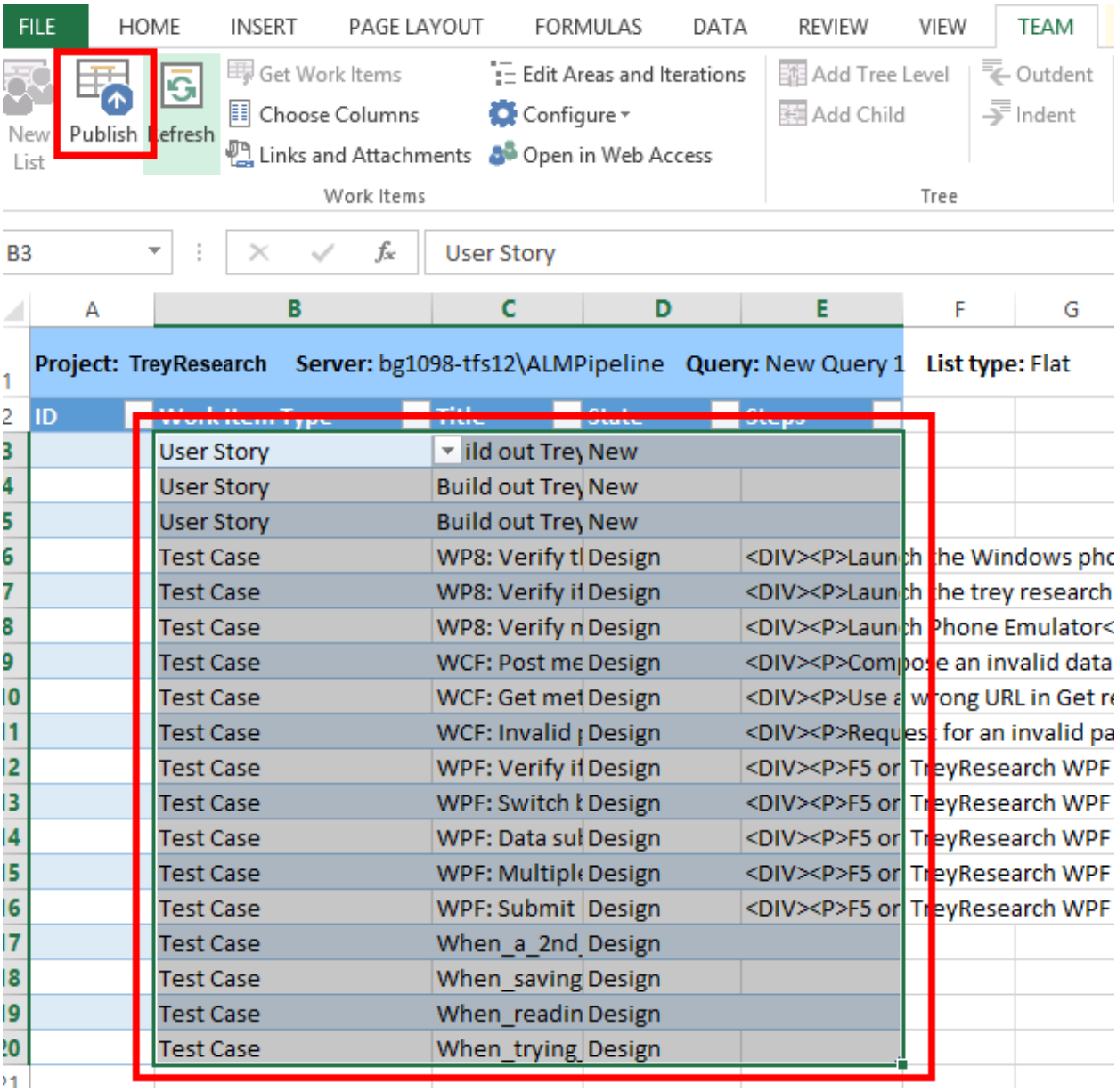
Task 2: Adding the WITs to the TFS Team Project

In this task you add the WITs from the Excel spreadsheet on your local machine to the TFS Excel form.

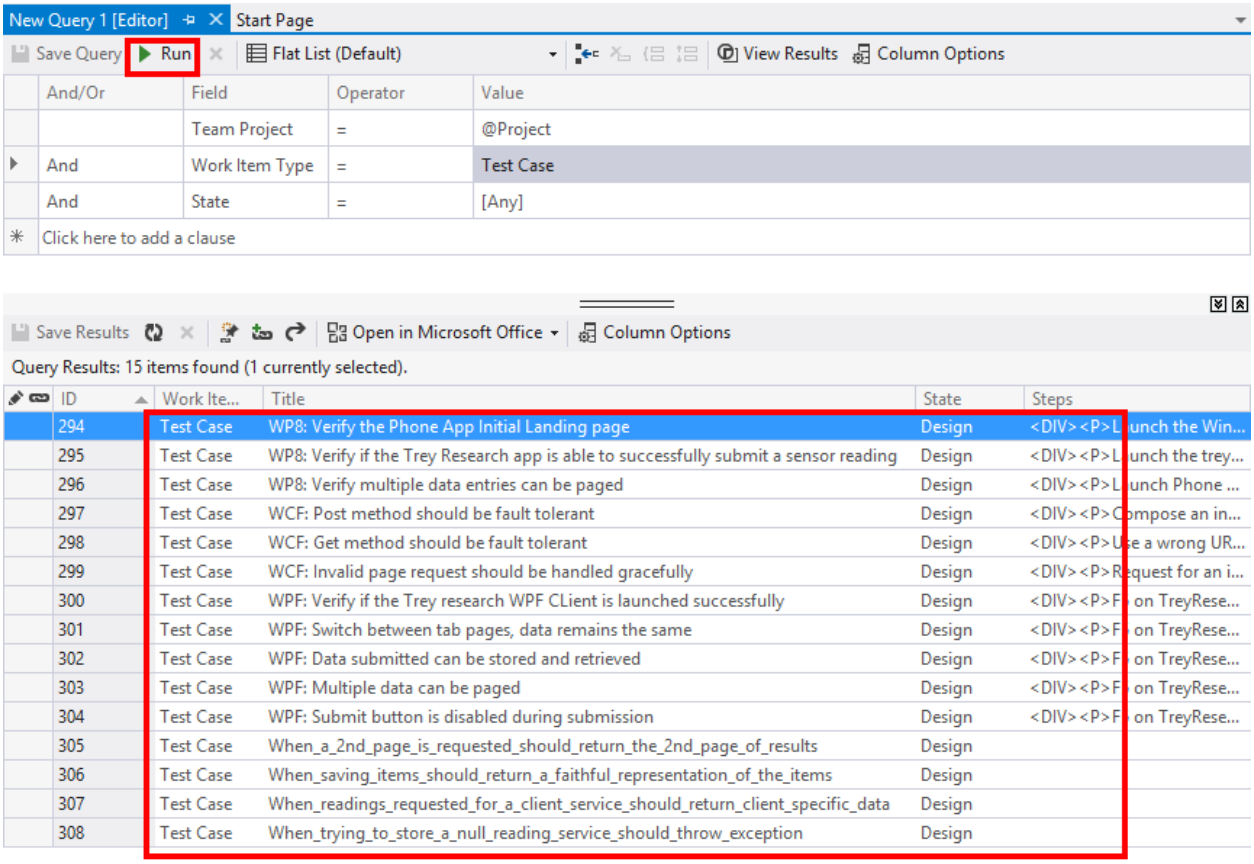
1. Open the Excel spreadsheet named Lab1_WIT.xlsx. It is located in the Lab01-StartingPoint directory. This spreadsheet contains the data for the user stories and the WITs for the test cases. Copy the data. Note that the order of the column values map to the ones in the TFS Excel Query Form worksheet.

User Story	ild out Trey Phone App	New	
User Story	Build out Trey WCF Service	New	
User Story	Build out Trey WPF Client	New	
Test Case	WP8: Verify the Phone App Initial Landing page	Design	<DIV><P>
Test Case	WP8: Verify if the Trey Research app is able to successfully submit a sensor	Design	<DIV><P>
Test Case	WP8: Verify multiple data entries can be paged	Design	<DIV><P>
Test Case	WCF: Post method should be fault tolerant	Design	<DIV><P>
Test Case	WCF: Get method should be fault tolerant	Design	<DIV><P>
Test Case	WCF: Invalid page request should be handled gracefully	Design	<DIV><P>
Test Case	WPF: Verify if the Trey research WPF CLient is launched successfully	Design	<DIV><P>
Test Case	WPF: Switch between tab pages, data remains the same	Design	<DIV><P>
Test Case	WPF: Data submitted can be stored and retrieved	Design	<DIV><P>
Test Case	WPF: Multiple data can be paged	Design	<DIV><P>
Test Case	WPF: Submit button is disabled during submission	Design	<DIV><P>
Test Case	When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results	Design	
Test Case	When_saving_items_should_return_a_faithful_representation_of_the_iten	Design	
Test Case	When_readings_requested_for_a_client_service_should_return_client_spe	Design	
Test Case	When_trying_to_store_a_null_reading_service_should_throw_exception	Design	

2. Paste the copied values Into the TFS Excel form. Begin in the **Work Item Type** column, not the **ID** column.



3. Click **Publish**. The WITs are published to the TFS project.
4. In Visual Studio, refresh your query. The WITs you added appear.



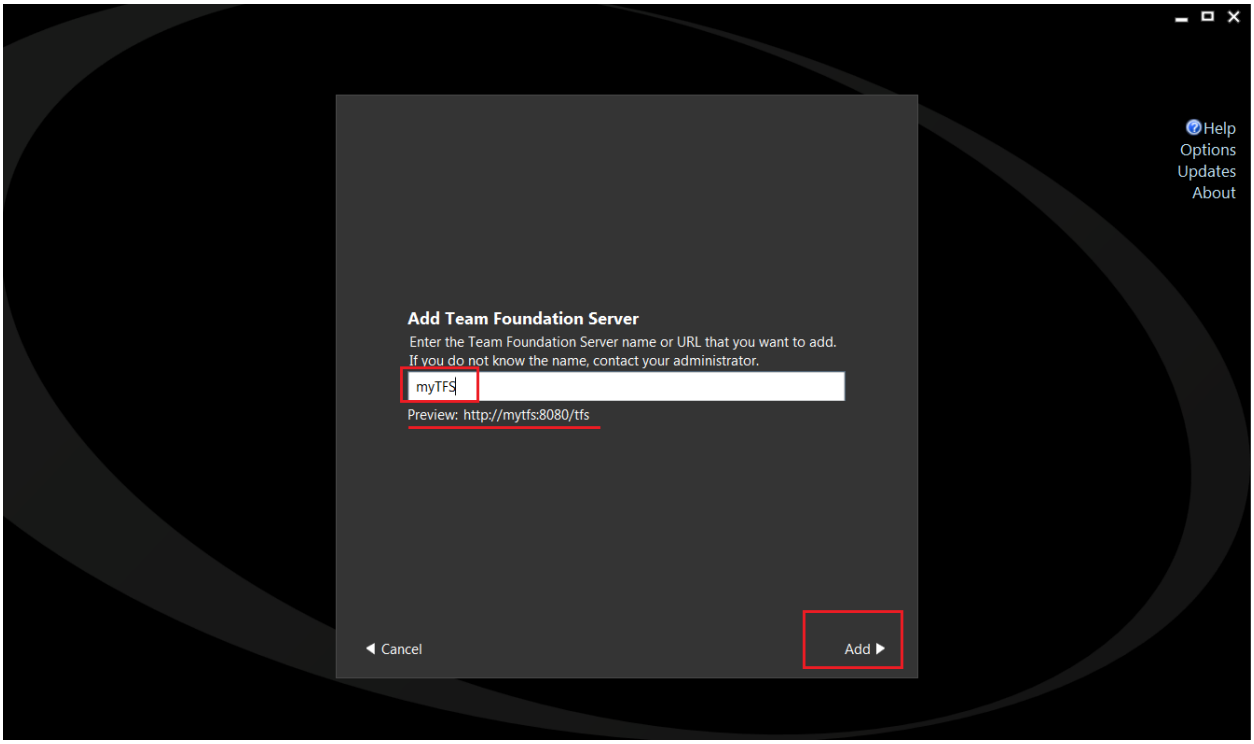
Exercise 8: Creating a Test Plan in Microsoft Test Manager (Optional)

This exercise is optional, but it will help you to understand how the automation and MTM works with the pipeline.

In this exercise, you create an MTM test plan and add test suites to it. MTM allows you to manage builds, test environments and test cases. If you aren't familiar with MTM, see Getting Started with Lab Management. There is also some information about MTM in the Introduction document that is included with this guidance.

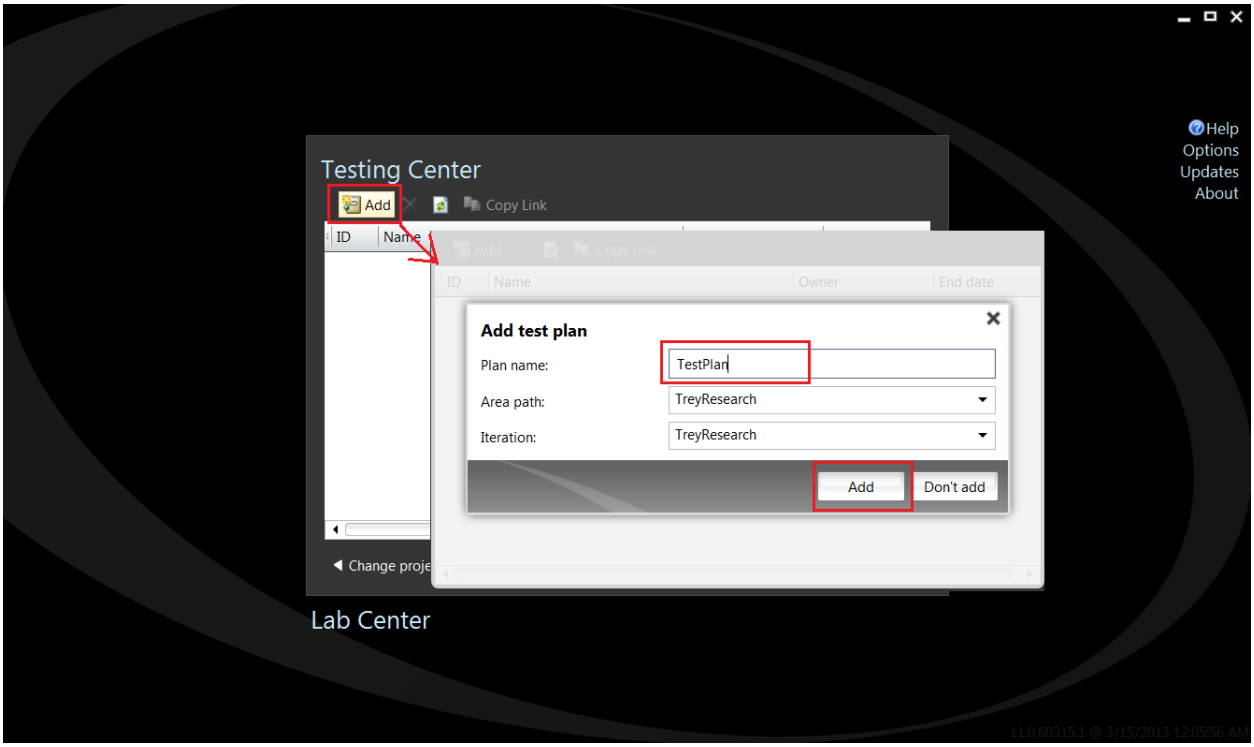
Task 1: Create the Test Plan

1. Open MTM and add the name of the Team Foundation Server that stores the Trey Research team project. Click **Add**.

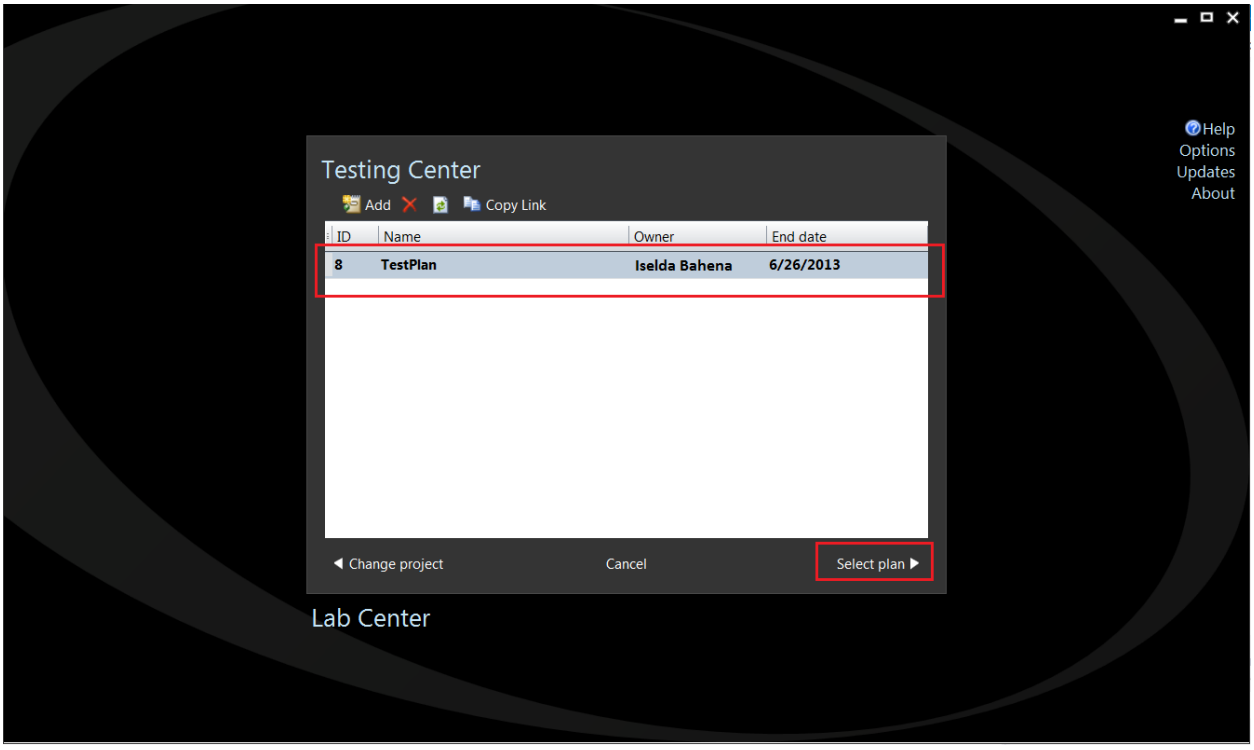


2. Select the **TreyResearch** team project. Click **Connect Now**.

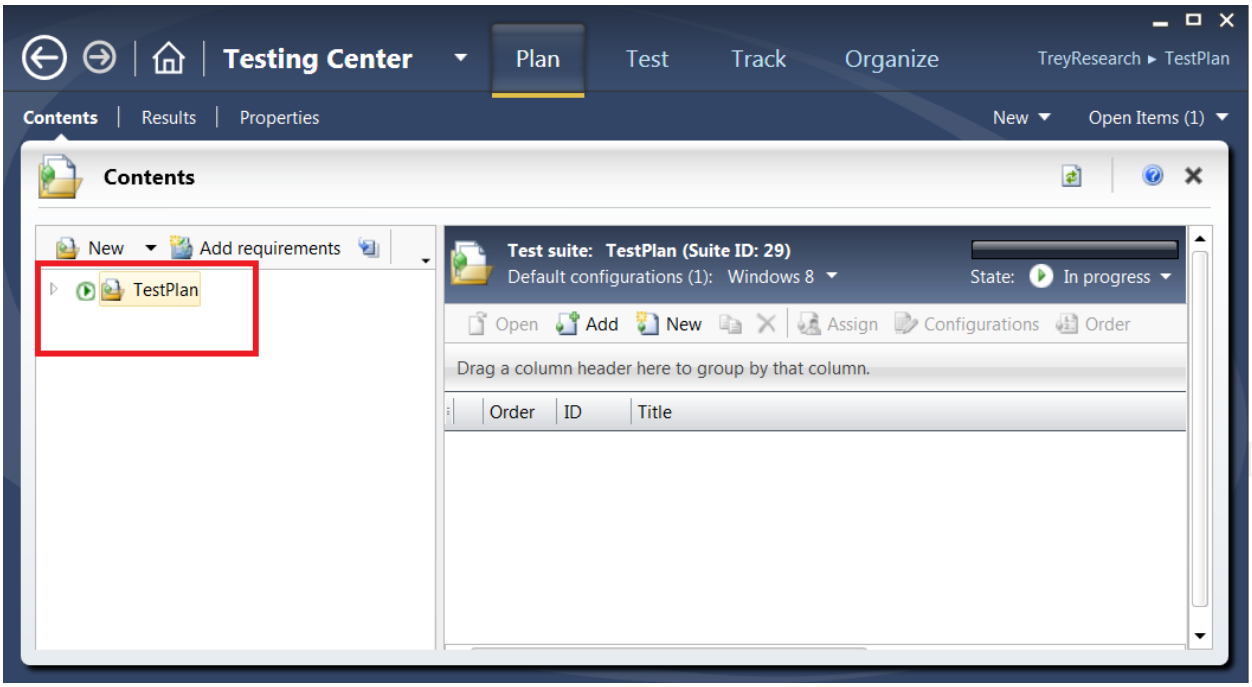
3. The **Testing Center** dialog box opens. Click **Add** to create a test plan. The **Add Test Plan** dialog box opens. In the **Plan name** box, enter **TestPlan**. Click **Add**.



4. Select the test plan. Click **Select plan**.



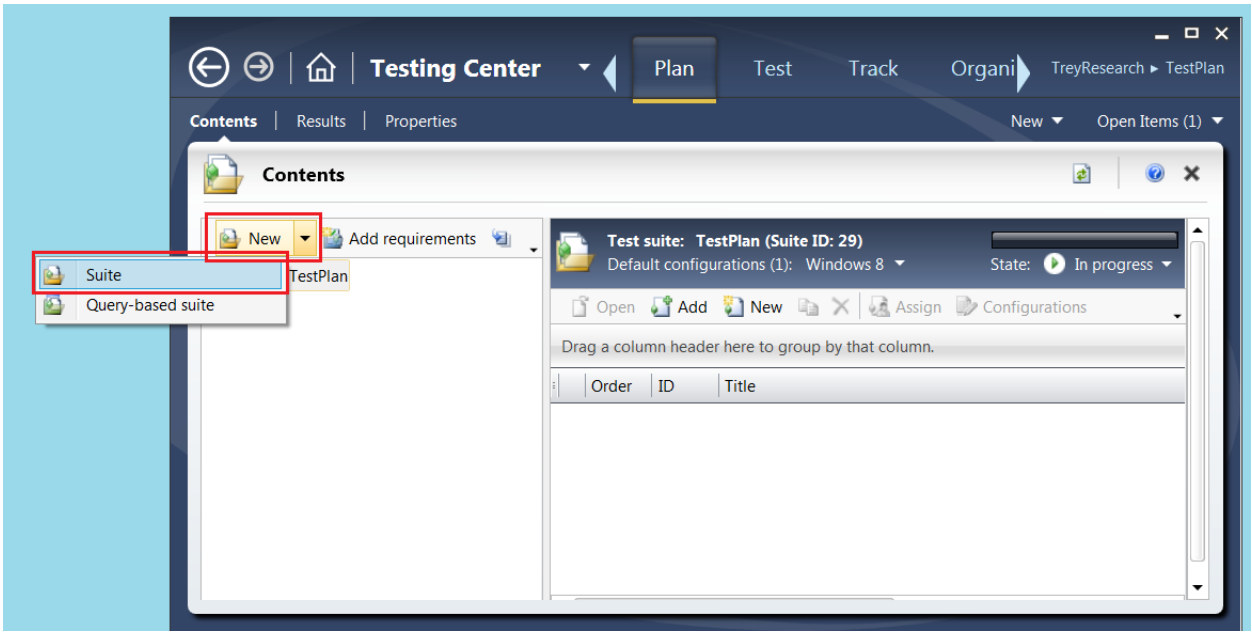
5. The Testing Center displays the new test plan. If you use the tree view, the top node is the **TestPlan** node.



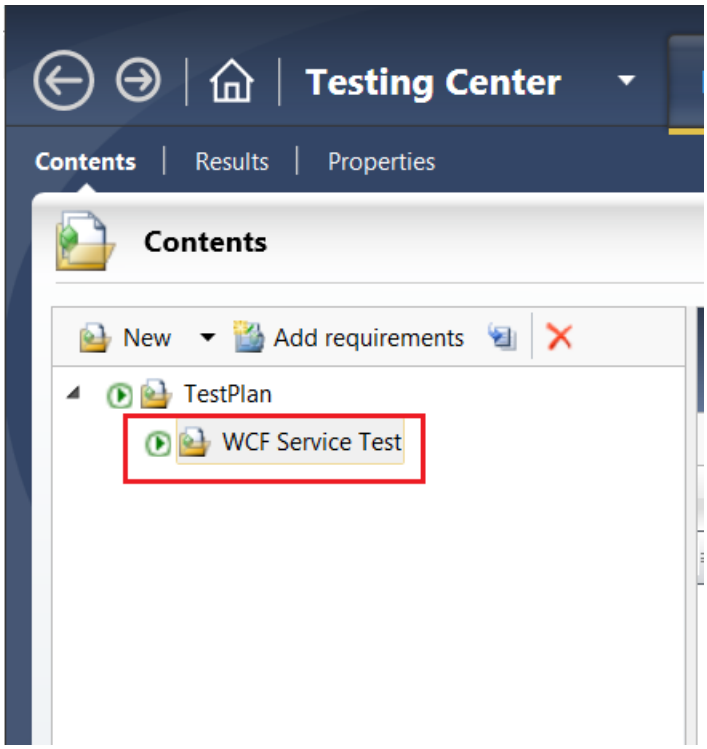
Task 2: Add Test Suites to the Test Plan

In this task, you add test suites to the test plan. Suites organize your test cases. They can be composed of both manual and automated test cases. For more information, see [Quick Start Guide for Manual Testing using Microsoft Test Manager.]

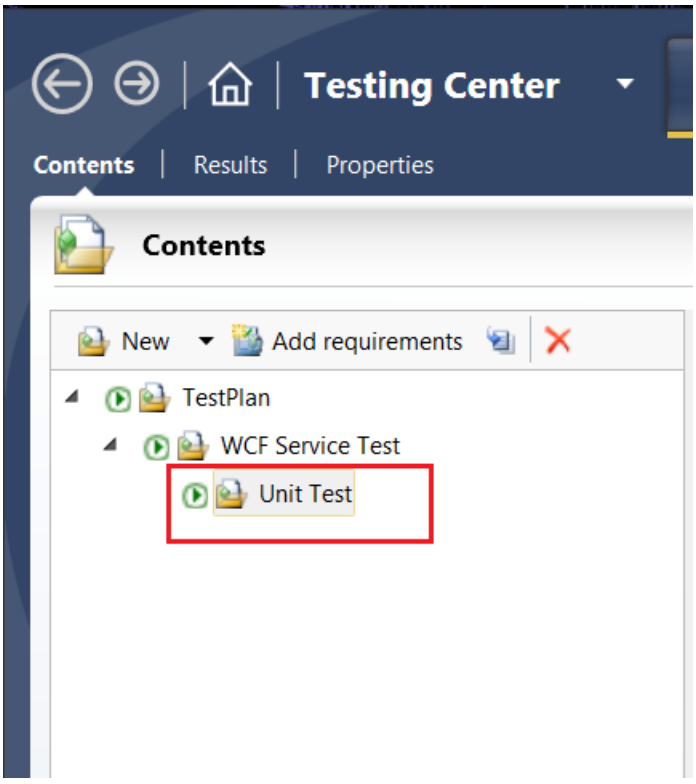
1. Select the **Test Plan** node. Click **New**. Select **Suite** from the drop-down list. A new suite node appears under **TestPlan**.



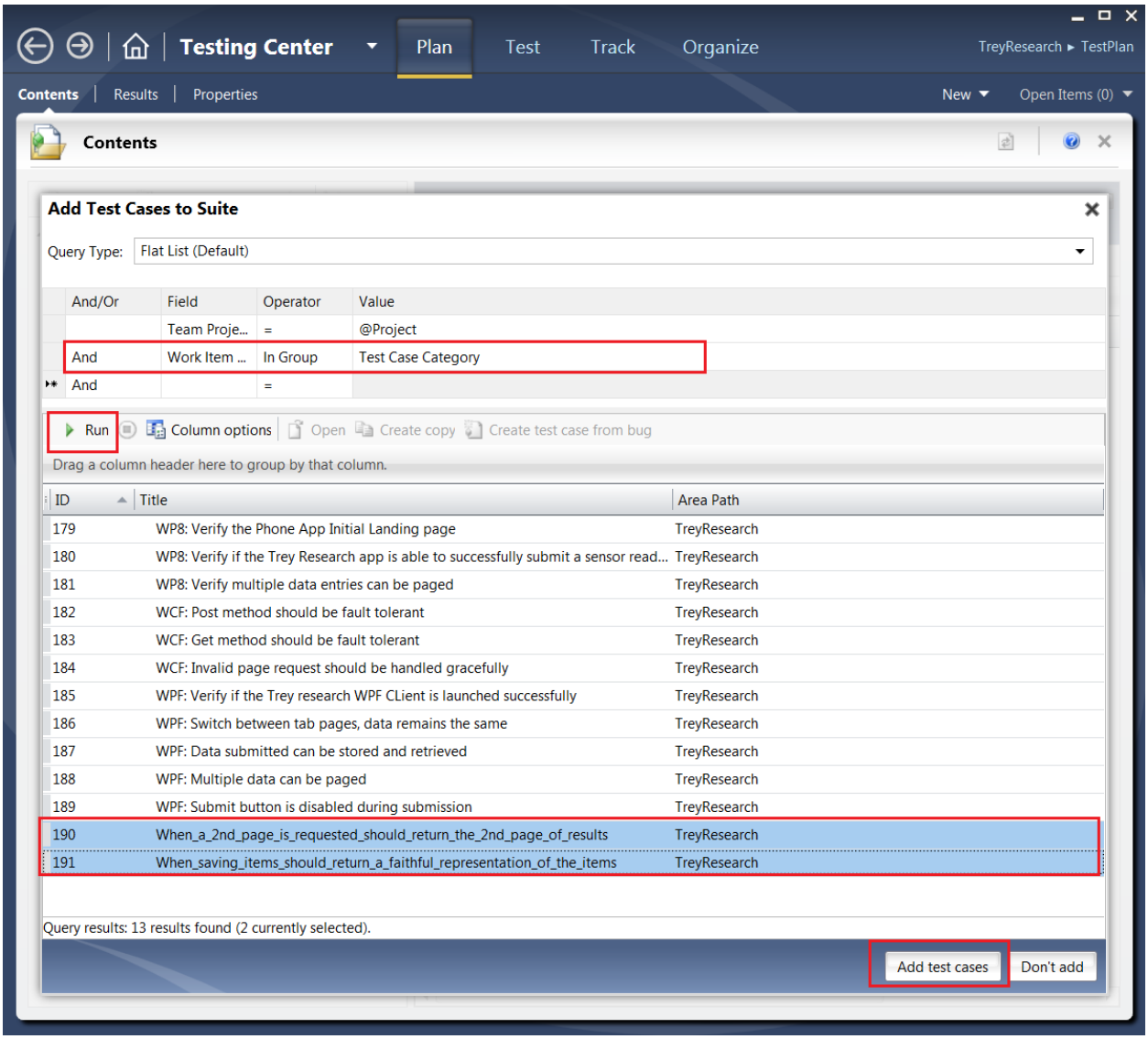
2. Name the suite **WCF Service Test**.



3. Select **WCF Service Test**. Right-click on it. In the context menu, click **New suite**. Name this suite **Unit Test**.

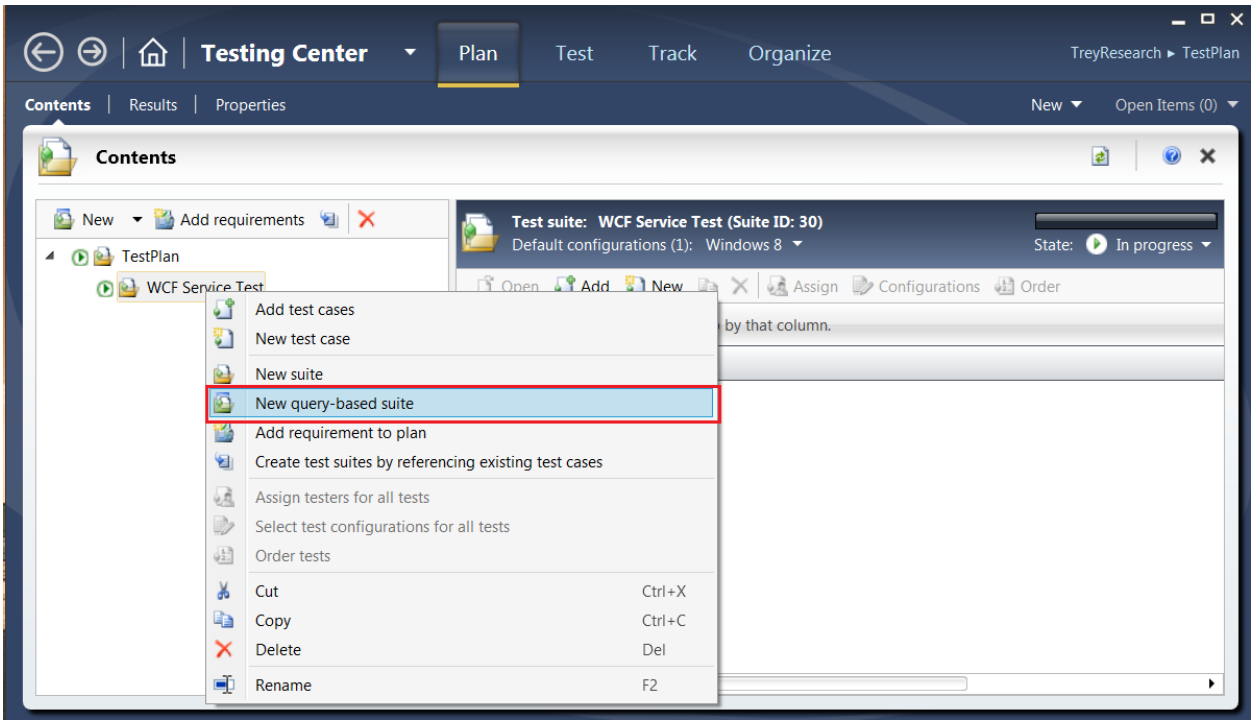


4. Right-click on **Unit Test**. Click **Add test cases**. The query window appears. Click **Run**. Because the filter is set to **Test Case Category**, the test cases you added earlier are visible. Select the two test cases without a WPF:, a WP8: or a WCF: in front of the name. Click **Add test cases**. You return to the Testing Center page.

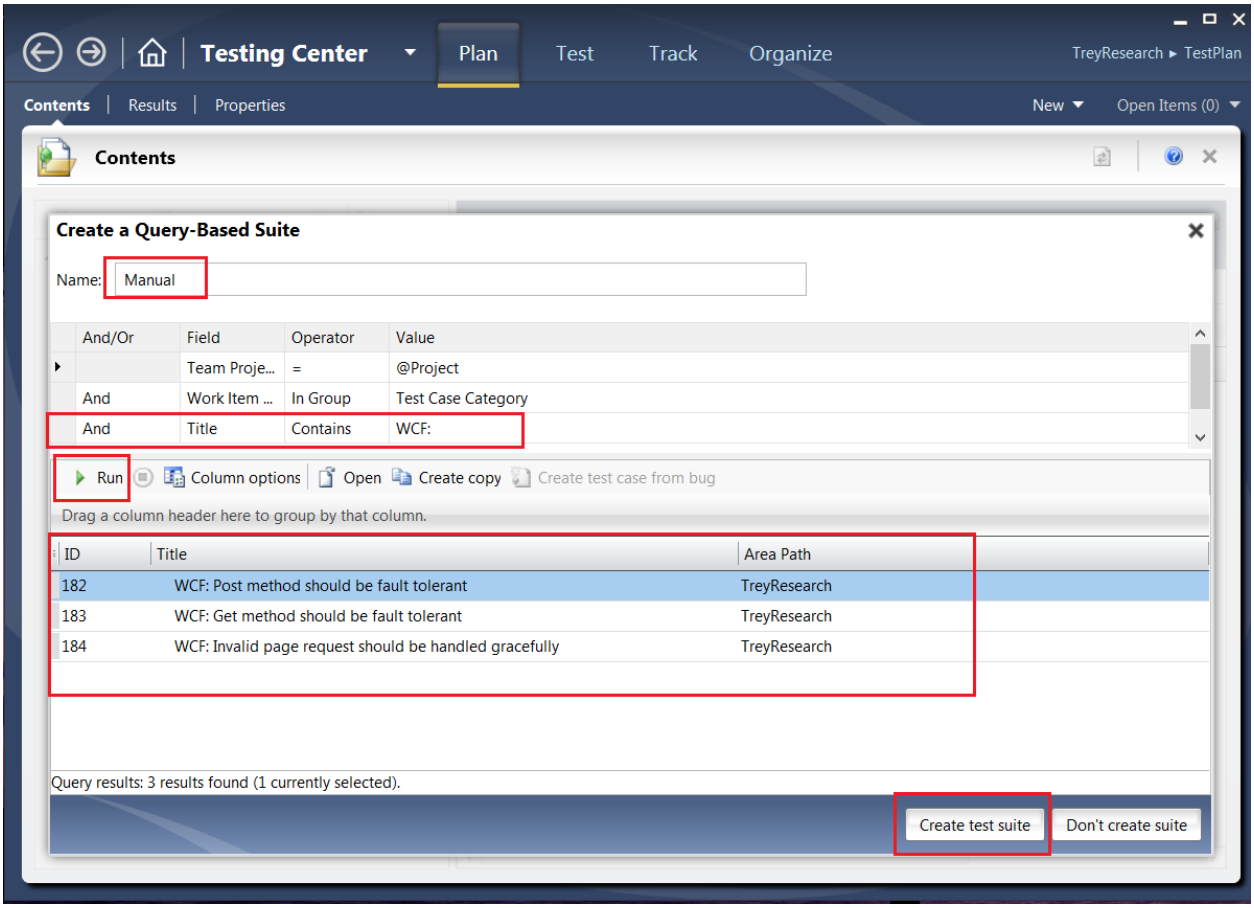


5. Select **WCF Service Test**. Right-click on it. In the context menu, select **New query-based suite**. The query edit form appears.

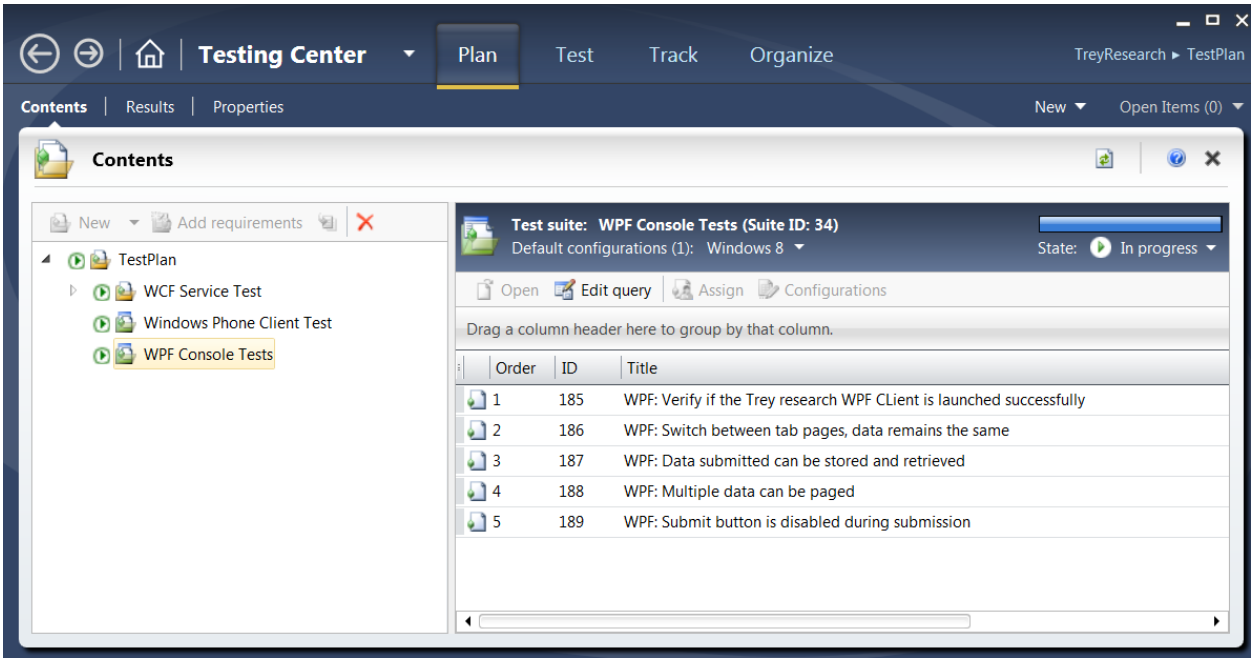
Note: Query-based suites allow you to actively query TFS for new or changed test cases. Every time you open your test plan, it queries TFS for any changes.



6. In the query editor, in the **Name** text box, enter **Manual**. For the filter, use **And Title Contains WCF:**. Click **Run**. All the test cases with **WCF:** in the title appear. Click **Create test suite**.



7. Select **TestPlan**. Add a query-based suite. Name this suite **Windows Phone Client Test**. Repeat step 6 but the filter is now **And Title Contains WP8:**. Click **Create test suite**.
8. Select the **TestPlan** again. Add a query-based suite named **WPF Console Tests**. Repeat step 6 but the filter is now **And Title Contains WPF:** Click **Create test suite**.
9. The test plan should now look like the following screenshot.



Note: Advance Lab using Windows Phone 8

- WP8: Verify the Phone App Initial Landing page
- WP8: Verify if the Trey Research app is able to successfully submit a sensor reading
- WP8: Verify multiple data entries can be paged

Summary

In this lab you set up everything you will need for the lab that follows. Because it is an advanced lab, it includes the steps required for the Windows Phone 8 app and for the Windows Azure services.

You first installed the Trey Research solution, created the TreyResearch team project, and then added the solution to the project. You then built the solution and ran the unit tests.

In the next exercise, you published and validated the WCF service web role to the Windows Azure VM. You then deployed the WPF application and the Windows Phone 8 client.

In the third exercise you installed the test artifacts. You added items in an Excel spreadsheet to a TFS Excel form.

In the fourth exercise you created a simple build definition for a pipeline that performs continuous integration and runs unit tests. In the fifth exercise, you created a tested plan in MTM, added test cases to the test suites, and added the test suites to the test plan. Finally, you installed the TFS Build Extensions, which you will need in the next lab.

Copyright