Hands-on Lab 3.3: Running the Automated Tests

Table of Contents

Objectives

This HOL demonstrates how to set up and run automated tests in the pipeline stages. It is part three of the Automation HOL. Together, the three parts demonstrate how to use Microsoft Visual Studio, Microsoft Test Manager (MTM), Microsoft Team Foundation Server (TFS) and Lab Management to automate, deploy, and test the following technologies:

- Windows Communications Foundation (WCF)
- Windows Presentation Foundation (WPF)

The subject of automation is covered in Chapter 4 of Building a Release Pipeline with Team Foundation Server 2012. Because the intention of this HOL is to show you how to incorporate automated tests into a continuous delivery pipeline, it relies on a test plan and automated tests that are supplied as part of the sample Trey Research application. The lab uses MTM to manage the test plan, test suites, and test cases. The lab does not show you how to how to use MTM to create a test strategy or a test plan. It also assumes that you know how to write automated tests. If you are unfamiliar with these tasks, refer to the articles mentioned in this lab.

Prerequisites

Make sure you have completed all the previous labs.

NOTE: You can use either standard environments or SCVMM environments for this lab. Standard environments are much simpler to set up because you do not need to configure the network virtualization but some features, such as snapshots, are not available.

The Trey Research application is in your TreyResearch folder. Visual Studio solutions that are the result of completing all of the tasks in an exercise are in the **Lab03-Automation\Completed-Lab** folder.

Your run the examples for this lab on your local computer.

Time

If you are familiar with Visual Studio, Web Deploy (MSDeploy), IIS, MSBuild, Wix, TFS, and Lab Management, you should be able to complete all 3 parts of the HOL in three to four hours.

You should be able to complete all of the exercises in this lab in approximately 60 minutes.

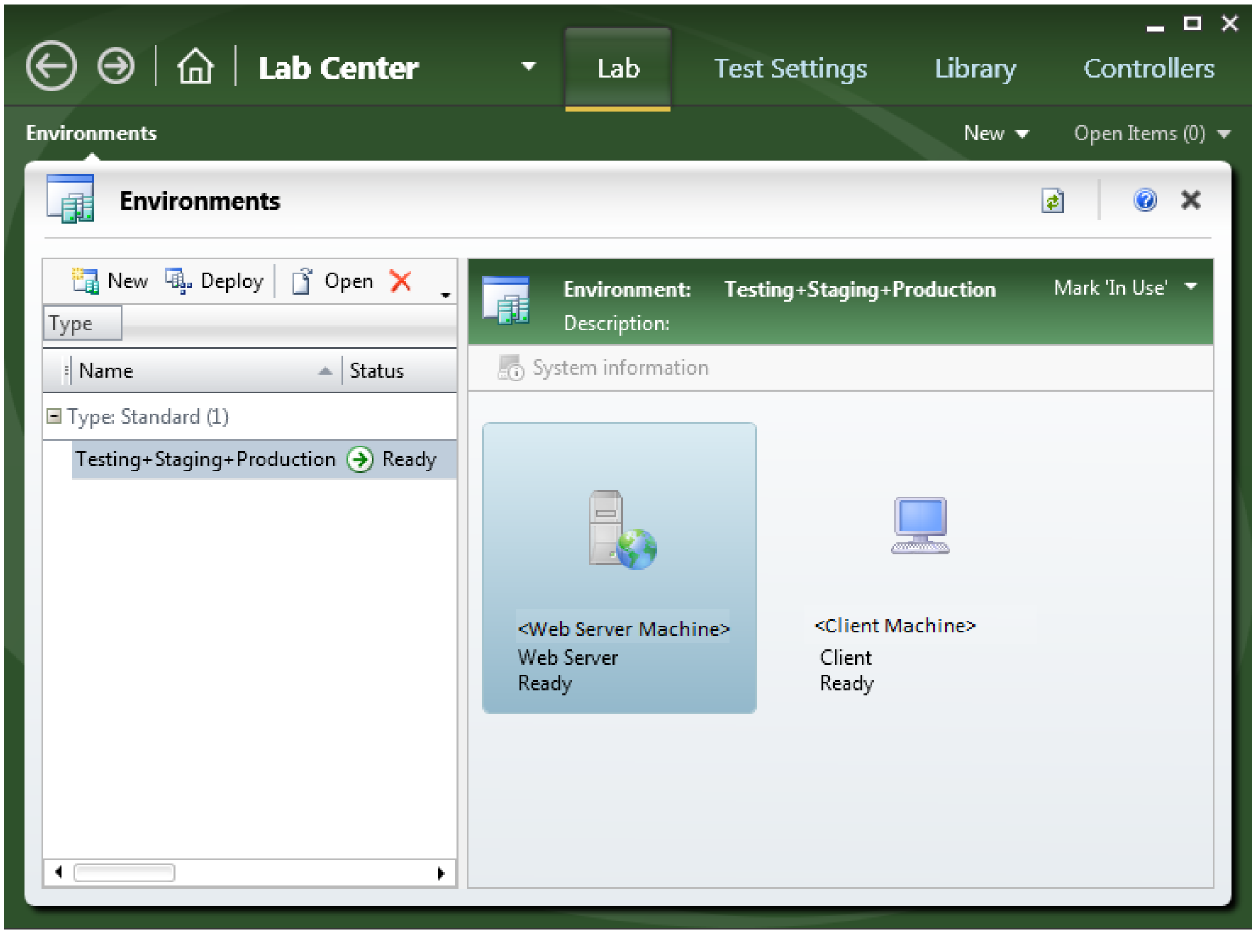
Exercise 1: Setting Up the Target Environments

In this exercise you set up the target environments so they can run the automated tests.

Task 1: Set Up the Environment in Lab Management

In this task, you make a single change to the Lab Management environment that you used in the other automation labs. Because some of the build verification tests include automated user interface tests (also known as CodedUI), the environment must supply the test agent with a user interface.

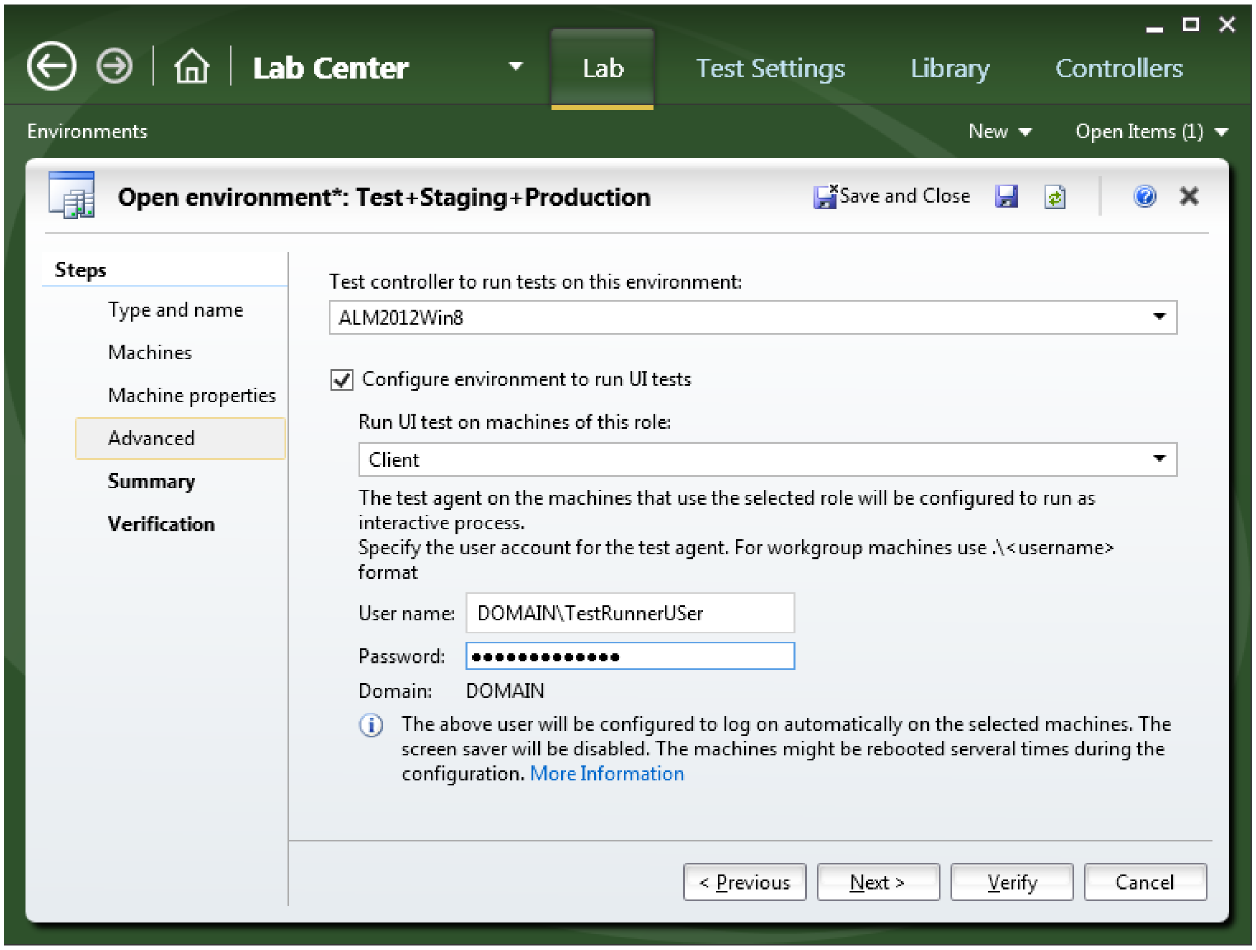
1. Open the **Lab Management** console for the project. Select the environment you used in the previous automation labs. Click **Open**.



2. Select the **Advanced** tab. Select **Configure environment to run UI tests**. Select **Client** from the **Run UI test on machines of this role** drop-down list. This is where where the WPF application is deployed. The WPF application has the user interface.

3. Enter the **User name** and **Password** for the user account that will run the UI tests.

Note: The user account depends on your environment. If you are using the Brian Keller VM then use the adminstrator account.



Open environment*: Test+Staging+Production

Test controller to run tests on this environment: ALM2012Win8

✓ Configure environment to run UI tests

Run UI test on machines of this role: Client

The test agent on the machines that use the selected role will be configured to run as interactive process.
Specify the user account for the test agent. For workgroup machines use .\<username> format

User name: DOMAIN\TestRunnerUSer

Password: ●●●●●●●●●●●●●

Domain: DOMAIN

The above user will be configured to log on automatically on the selected machines. The screen saver will be disabled. The machines might be rebooted serveral times during the configuration. More Information

4. Click **Verify**. After all the verifications pass, click **Finish**.

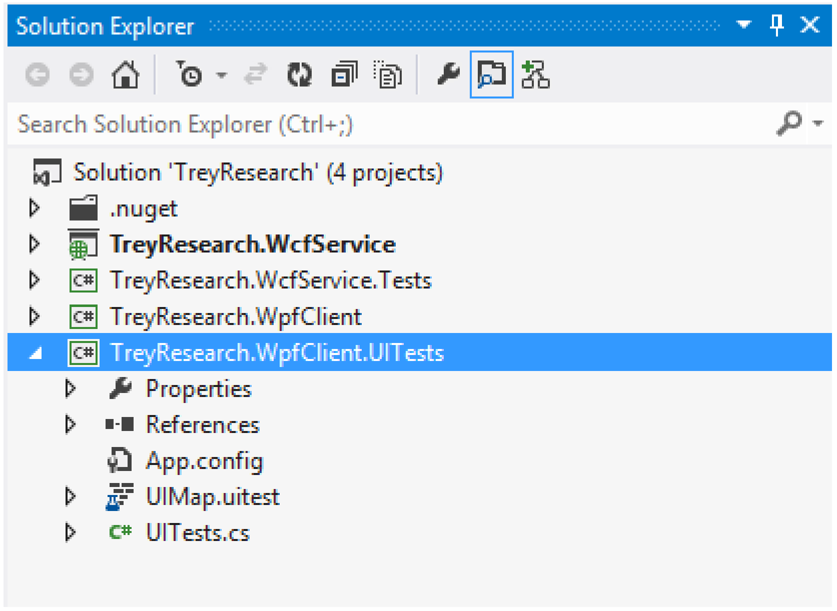
NOTE: Even if you check the **Configure environment to run UI tests** option here, you won't be able to run the UI tests unless you have an open session on the target computer. This can either be a remote desktop session or a local session, but it must be active, not minimized, and the screensaver should be disabled. For more information refer to How to: Set Up Your Test Agent to Run Tests that Interact with the Desktop.

Exercise 2: Add the UI Tests to the Trey Research Solution

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

Task 1 : Add the UI Tests

1. Copy the entire **TreyResearch.WpfClient.UITests** folder from **C:\HOL\Lab03-Automation\Completed-Lab\Core** to **C:\HOL\TreyReasearch\Source**.
2. Open Visual Studio and right-click on the **TreyResearch** solution. Point to **Add**. Select **Existing Project**. Browse to **C:\HOL\TreyResearch\Source\TreyResearch.WpfClient.UITests\TreyResearch.WpfClient.UITests**.
3. Click **Add**. The following screenshot shows what the final file hierarchy should look like.



```
Solution 'TreyResearch' (4 projects)
  .nuget
  TreyResearch.WcfService
  TreyResearch.WcfService.Tests
  TreyResearch.WpfClient
  TreyResearch.WpfClient.UITests
    Properties
    References
    App.config
    UIMap.uitest
    UITests.cs
```

4. Check in the changes.

Exercise 3: Setting up Automated Testing for the Commit Stage

In this exercise you categorize the unit tests that are included in the Trey Research sample application. You then modify the commit stage build definition to run the automated tests. In terms of automation, the commit stage differs from other stages in the pipeline because it only runs unit tests, while other stages can run a variety of tests.

If you need information about how to write unit tests, see Chapter 2 Unit Testing: Testing the Inside of *Testing for Continuous Delivery with Visual Studio 2012*. For more information about test categories, see How To: Group and Run Automated Tests Using Test Categories.

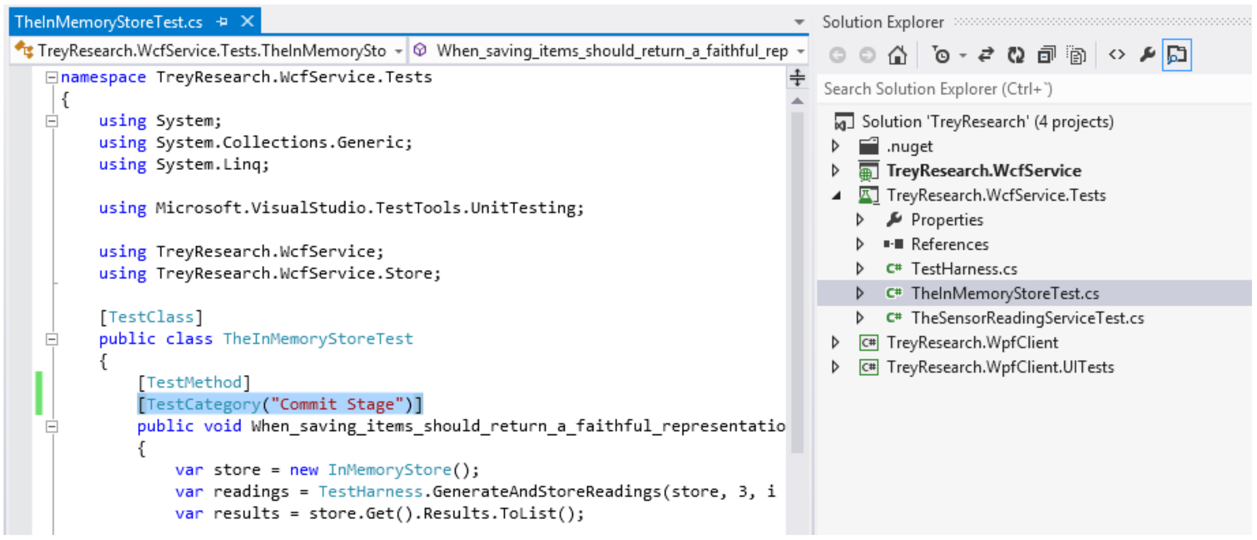
Task 1: Categorize the Unit Tests

In this task you categorize the unit tests so that the commit stage knows which tests to run. To do this you use the **Test Traits** feature in Visual Studio. Specifically, you use the **Test Category** trait.

1. In Visual Studio, open the **TreyResearch** solution. Inside it, there is a project named **TreyResearch.WcfService.Test**. It contains some unit tests. Open the test class named **TheInMemoryStoreTest.cs**.
2. The class contains two test methods. You will assign a category only to the first test. Later you will see that the commit stage only runs the test that belongs to the category. Locate the test method named **When_saving_items_should_return_a_faithful_representation_of_the_items**.
3. Just after the **[TestMethod]** attribute and before the method signature, add the following attribute that specifies the test method category.

C#

```
[TestCategory("Commit Stage")]
```

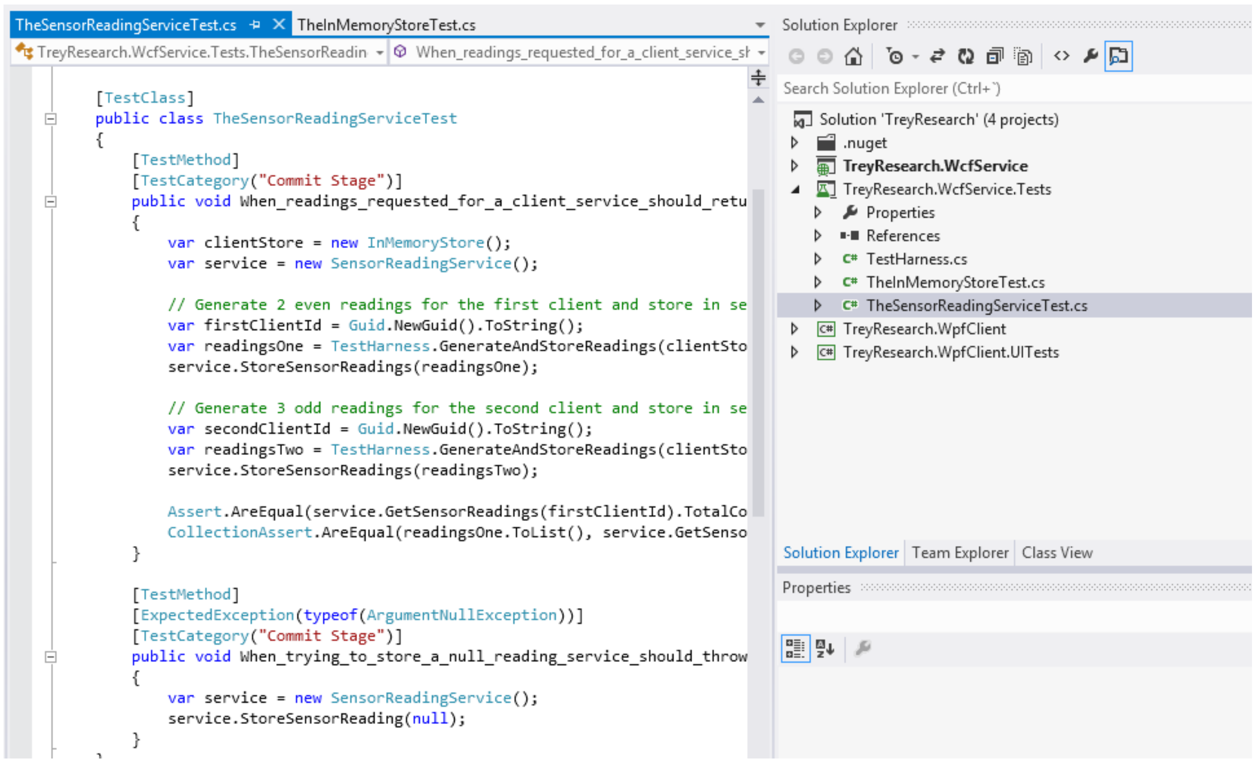


TheInMemoryStoreTest.cs
TreyResearch.WcfService.Tests.TheInMemorySto
When_saving_items_should_return_a_faithful_rep
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'TreyResearch' (4 projects)
.nuget
TreyResearch.WcfService
TreyResearch.WcfService.Tests
Properties
References
TestHarness.cs
TheInMemoryStoreTest.cs
TheSensorReadingServiceTest.cs
TreyResearch.WpfClient
TreyResearch.WpfClient.UITests

```
namespace TreyResearch.WcfService.Tests
{
    using System;
    using System.Collections.Generic;
    using System.Linq;

    using Microsoft.VisualStudio.TestTools.UnitTesting;

    using TreyResearch.WcfService;
    using TreyResearch.WcfService.Store;

    [TestClass]
    public class TheInMemoryStoreTest
    {
        [TestMethod]
        [TestCategory("Commit Stage")]
        public void When_saving_items_should_return_a_faithful_representatio
        {
            var store = new InMemoryStore();
            var readings = TestHarness.GenerateAndStoreReadings(store, 3, i
            var results = store.Get().Results.ToList();
```

4. Save the file and check it in.

Note: You can assign this category to any test that you want to run during the commit stage. For example, the following screenshot shows that the two tests contained in the **TheSensorReadingServiceTest** test class have been added to the category.

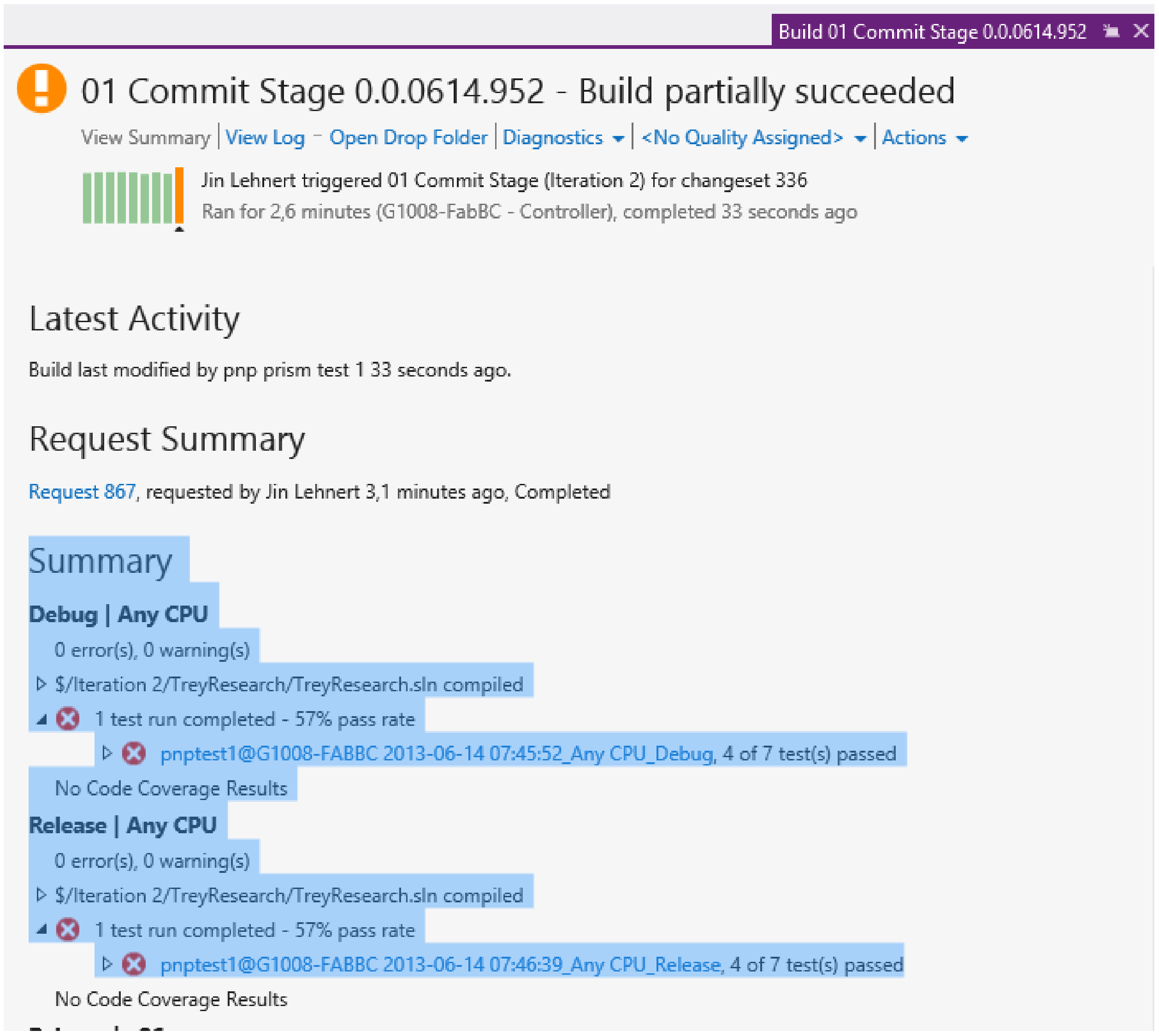

TheSensorReadingServiceTest.cs
TheInMemoryStoreTest.cs
TreyResearch.WcfService.Tests.TheSensorReadin
When_readings_requested_for_a_client_service_sh
Solution Explorer
Search Solution Explorer (Ctrl+`)
Solution 'TreyResearch' (4 projects)
.nuget
TreyResearch.WcfService
TreyResearch.WcfService.Tests
Properties
References
TestHarness.cs
TheInMemoryStoreTest.cs
TheSensorReadingServiceTest.cs
TreyResearch.WpfClient
TreyResearch.WpfClient.UITests
Solution Explorer
Team Explorer
Class View
Properties

```
[TestClass]
public class TheSensorReadingServiceTest
{
    [TestMethod]
    [TestCategory("Commit Stage")]
    public void When_readings_requested_for_a_client_service_should_retu
    {
        var clientStore = new InMemoryStore();
        var service = new SensorReadingService();

        // Generate 2 even readings for the first client and store in se
        var firstClientId = Guid.NewGuid().ToString();
        var readingsOne = TestHarness.GenerateAndStoreReadings(clientSto
        service.StoreSensorReadings(readingsOne);

        // Generate 3 odd readings for the second client and store in se
        var secondClientId = Guid.NewGuid().ToString();
        var readingsTwo = TestHarness.GenerateAndStoreReadings(clientSto
        service.StoreSensorReadings(readingsTwo);

        Assert.AreEqual(service.GetSensorReadings(firstClientId).TotalCo
        CollectionAssert.AreEqual(readingsOne.ToList(), service.GetSenso
    }

    [TestMethod]
    [ExpectedException(typeof(ArgumentNullException))]
    [TestCategory("Commit Stage")]
    public void When_trying_to_store_a_null_reading_service_should_throw
    {
        var service = new SensorReadingService();
        service.StoreSensorReading(null);
    }
```

Task 2: Set Up the Commit Stage to Run the Automated Tests

In this task you edit the commit stage build definition so that it only runs unit tests that belong to the **Commit Stage** category.

1. In step 4 of the previous task you checked in some changes. The commit stage should have been triggered. Open the commit stage build summary. You should see that some tests ran.



Build 01 Commit Stage 0.0.0614.952

01 Commit Stage 0.0.0614.952 - Build partially succeeded

View Summary | View Log - Open Drop Folder | Diagnostics | <No Quality Assigned> | Actions

Jin Lehnert triggered 01 Commit Stage (Iteration 2) for changeset 336
Ran for 2,6 minutes (G1008-FabBC - Controller), completed 33 seconds ago

Latest Activity

Build last modified by pnp prism test 1 33 seconds ago.

Request Summary

Request 867, requested by Jin Lehnert 3,1 minutes ago, Completed

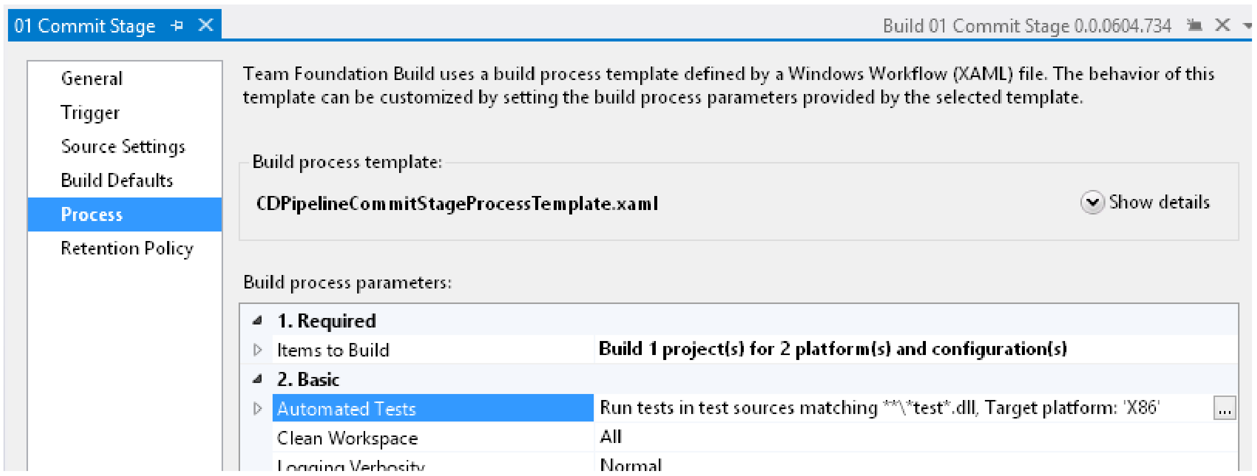
Summary

Debug | Any CPU
0 error(s), 0 warning(s)
$/Iteration 2/TreyResearch/TreyResearch.sln compiled
1 test run completed - 57% pass rate
pnptest1@G1008-FABBC 2013-06-14 07:45:52_Any CPU_Debug, 4 of 7 test(s) passed
No Code Coverage Results

Release | Any CPU
0 error(s), 0 warning(s)
$/Iteration 2/TreyResearch/TreyResearch.sln compiled
1 test run completed - 57% pass rate
pnptest1@G1008-FABBC 2013-06-14 07:46:39_Any CPU_Release, 4 of 7 test(s) passed
No Code Coverage Results

NOTE: The failing tests in the screenshot above are CodedUI tests that cannot run in the commit stage. Ignore these failures.

2. By default, the build definition runs all tests that are in assemblies that contain the word **test** in their names. You can verify this. Open the **01 Commit Stage** build definition. Select the **Process** tab. Go the the **Automated Tests** parameter under the **2. Basic** group. You will see the instruction for which automated tests to run during the build.



01 Commit Stage

Build 01 Commit Stage 0.0.0604.734

- General
- Trigger
- Source Settings
- Build Defaults
- Process
- Retention Policy

Team Foundation Build uses a build process template defined by a Windows Workflow (XAML) file. The behavior of this template can be customized by setting the build process parameters provided by the selected template.

Build process template:

CDPipelineCommitStageProcessTemplate.xaml

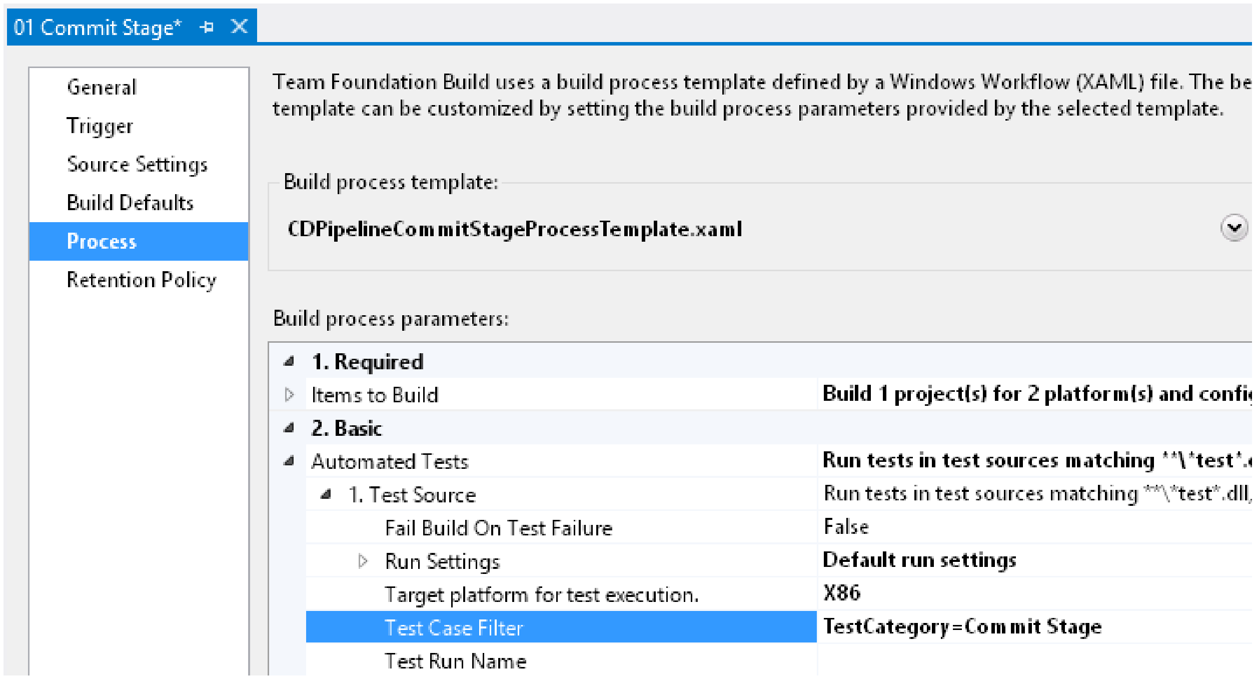
Show details

Build process parameters:

1. Required	
Items to Build	**Build 1 project(s) for 2 platform(s) and configuration(s)**
2. Basic	
Automated Tests	Run tests in test sources matching **\*test*.dll, Target platform: 'X86'
Clean Workspace	All
Logging Verbosity	Normal

.

3. In the **Process** section of the **01 Commit Stage** build definition, expand the **Automated Tests** parameter. Expand the **1. Test Source** parameter.
4. Locate the parameter named **Test Case Filter**. Add the filter **TestCategory=Commit Stage**. This filter means that only tests that belong to the **Commit Stage** category run.



01 Commit Stage*

- General
- Trigger
- Source Settings
- Build Defaults
- Process
- Retention Policy

Team Foundation Build uses a build process template defined by a Windows Workflow (XAML) file. The be template can be customized by setting the build process parameters provided by the selected template.

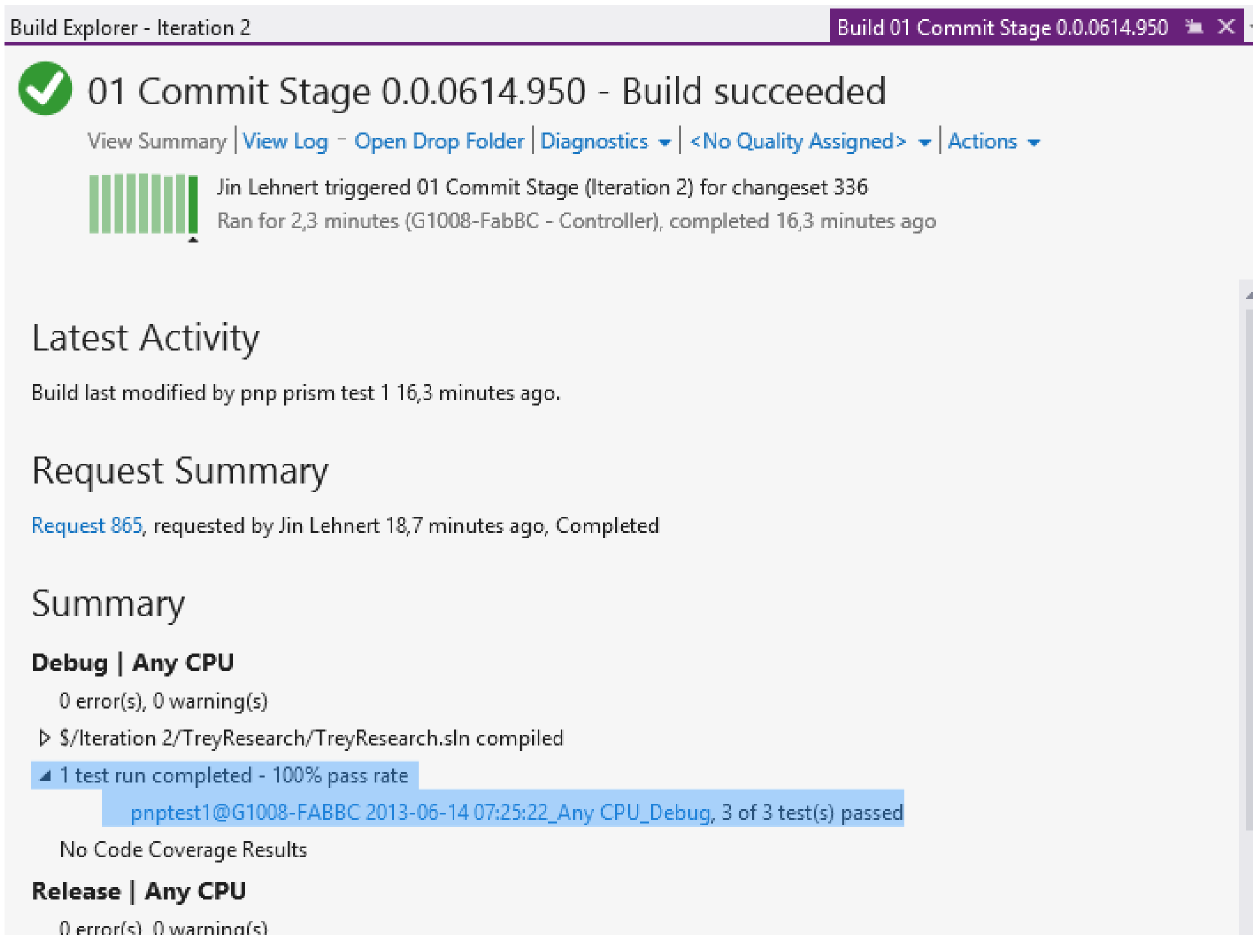
Build process template:

CDPipelineCommitStageProcessTemplate.xaml

Build process parameters:

1. Required	
Items to Build	**Build 1 project(s) for 2 platform(s) and confi**
2. Basic	
Automated Tests	**Run tests in test sources matching **\*test*.**
1. Test Source	Run tests in test sources matching **\*test*.dll,
Fail Build On Test Failure	False
Run Settings	**Default run settings**
Target platform for test execution.	**X86**
Test Case Filter	**TestCategory=Commit Stage**
Test Run Name	

5. Save the build definition and check it in. This should trigger a build.
6. Open the build summary. You should see that only one unit test ran.


Build Explorer - Iteration 2

Build 01 Commit Stage 0.0.0614.950

01 Commit Stage 0.0.0614.950 - Build succeeded

View Summary | View Log - Open Drop Folder | Diagnostics | <No Quality Assigned> | Actions

Jin Lehnert triggered 01 Commit Stage (Iteration 2) for changeset 336
Ran for 2,3 minutes (G1008-FabBC - Controller), completed 16,3 minutes ago

Latest Activity

Build last modified by pnp prism test 1 16,3 minutes ago.

Request Summary

Request 865, requested by Jin Lehnert 18,7 minutes ago, Completed

Summary

Debug | Any CPU

0 error(s), 0 warning(s)

$/Iteration 2/TreyResearch/TreyResearch.sln compiled

1 test run completed - 100% pass rate

pnptest1@G1008-FABBC 2013-06-14 07:25:22_Any CPU_Debug, 3 of 3 test(s) passed

No Code Coverage Results

Release | Any CPU

0 error(s), 0 warning(s)

Exercise 4: Setting up Automated Testing for the Remaining Stages

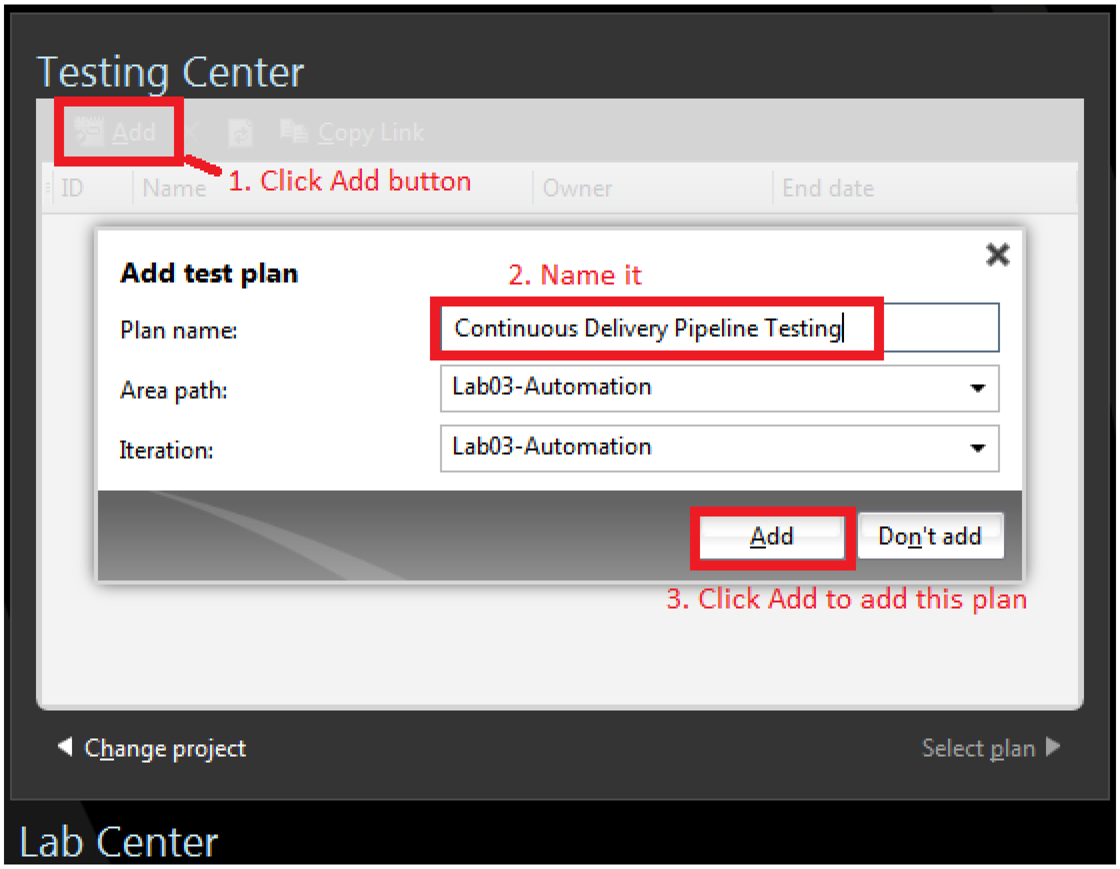
In this exercise,you use Microsoft Test Manager (MTM) to define the test settings and test configuration, to create test suites, and to associate automated tests with test cases.

For more information about defining test settings, see Create Test Settings for Automated Tests as Part of a Test Plan. For more information about test configurations, see Test Configuration – specifying test platforms.

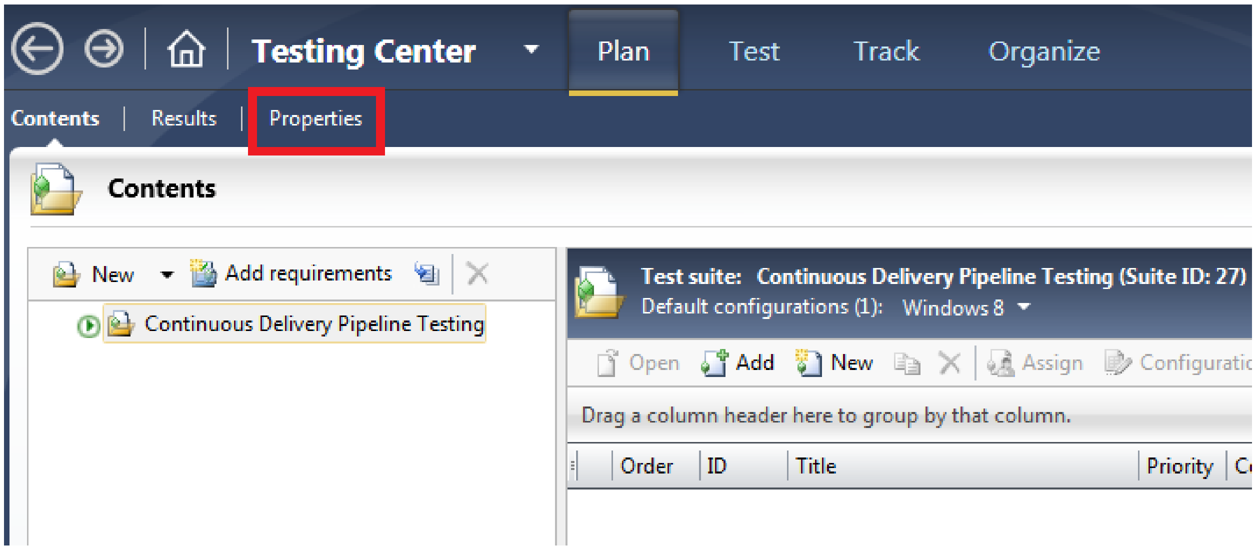
Task 1: Define the Test Settings

In this task, you use MTM to define the test settings and the test configuration of the test plan. (Lab Management builds and runs tests in the context of an MTM test plan.)

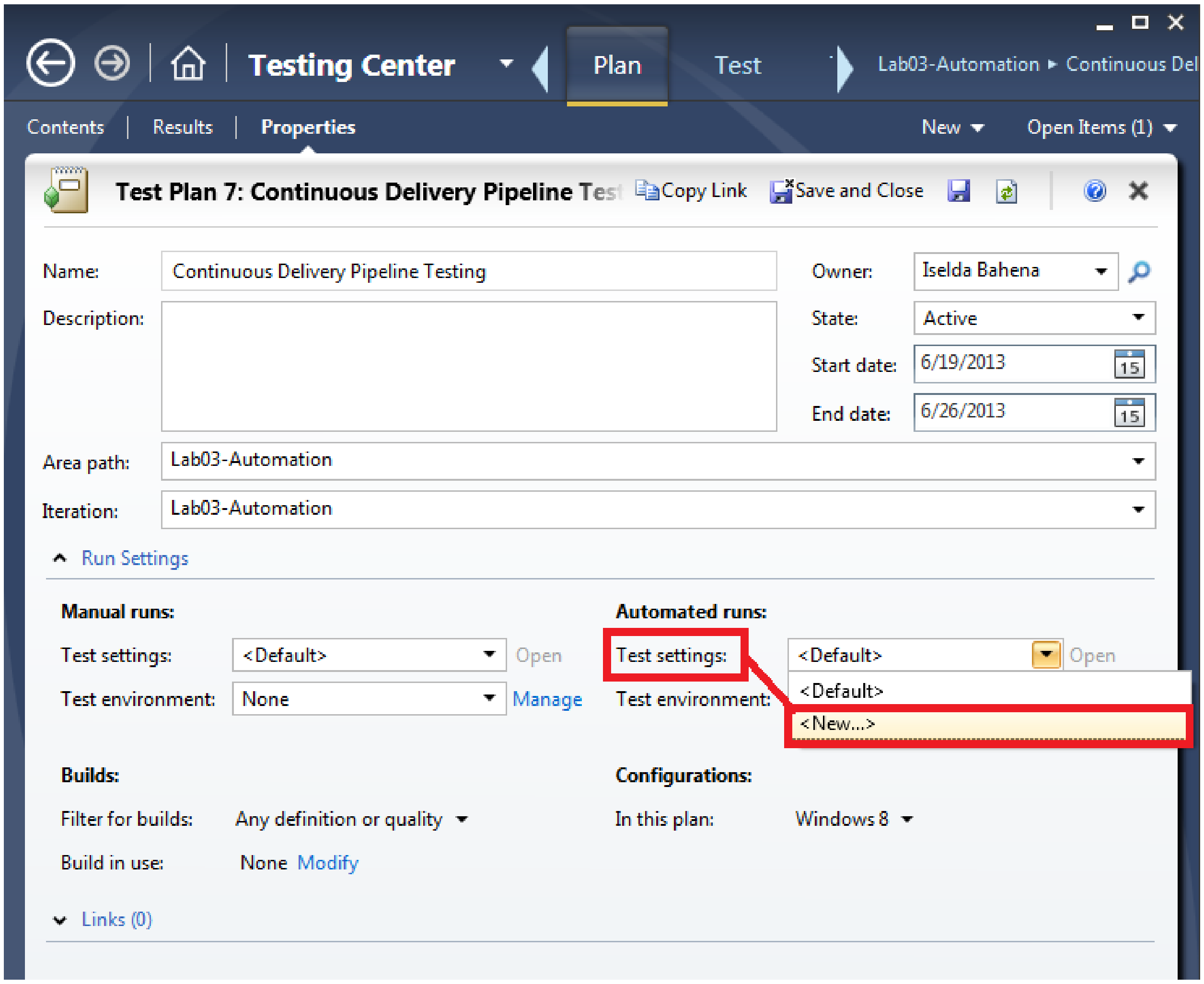
1. Open MTM and connect to the Trey Research team project.
2. Create a new test plan by clicking the **Add** button. In the **Plan name** text box, enter **Continuous Delivery Pipeline Testing**. Click **Add** (toward the bottom of the dialog box).



3. Select the **Plan** tab. Select **Properties**. The **Properties** page opens.


Testing Center
Plan
Test
Track
Organize
Contents
Results
Properties
Contents
New
Add requirements
Continuous Delivery Pipeline Testing
Test suite: Continuous Delivery Pipeline Testing (Suite ID: 27)
Default configurations (1): Windows 8
Open
Add
New
Assign
Configuratio
Drag a column header here to group by that column.
Order
ID
Title
Priority

4. Expand the **Run Settings** section. In the **Automated Runs** section, select **New** from the **Test settings** drop-down list. The **Properties** page for the test settings opens.


Testing Center
Plan
Test
Lab03-Automation ▸ Continuous Del
Contents | Results | Properties
New
Open Items (1)
Test Plan 7: Continuous Delivery Pipeline Test
Copy Link
Save and Close

Name: Continuous Delivery Pipeline Testing
Owner: Iselda Bahena
Description:
State: Active
Start date: 6/19/2013
End date: 6/26/2013
Area path: Lab03-Automation
Iteration: Lab03-Automation

Run Settings

Manual runs:
Test settings: <Default> Open
Test environment: None Manage

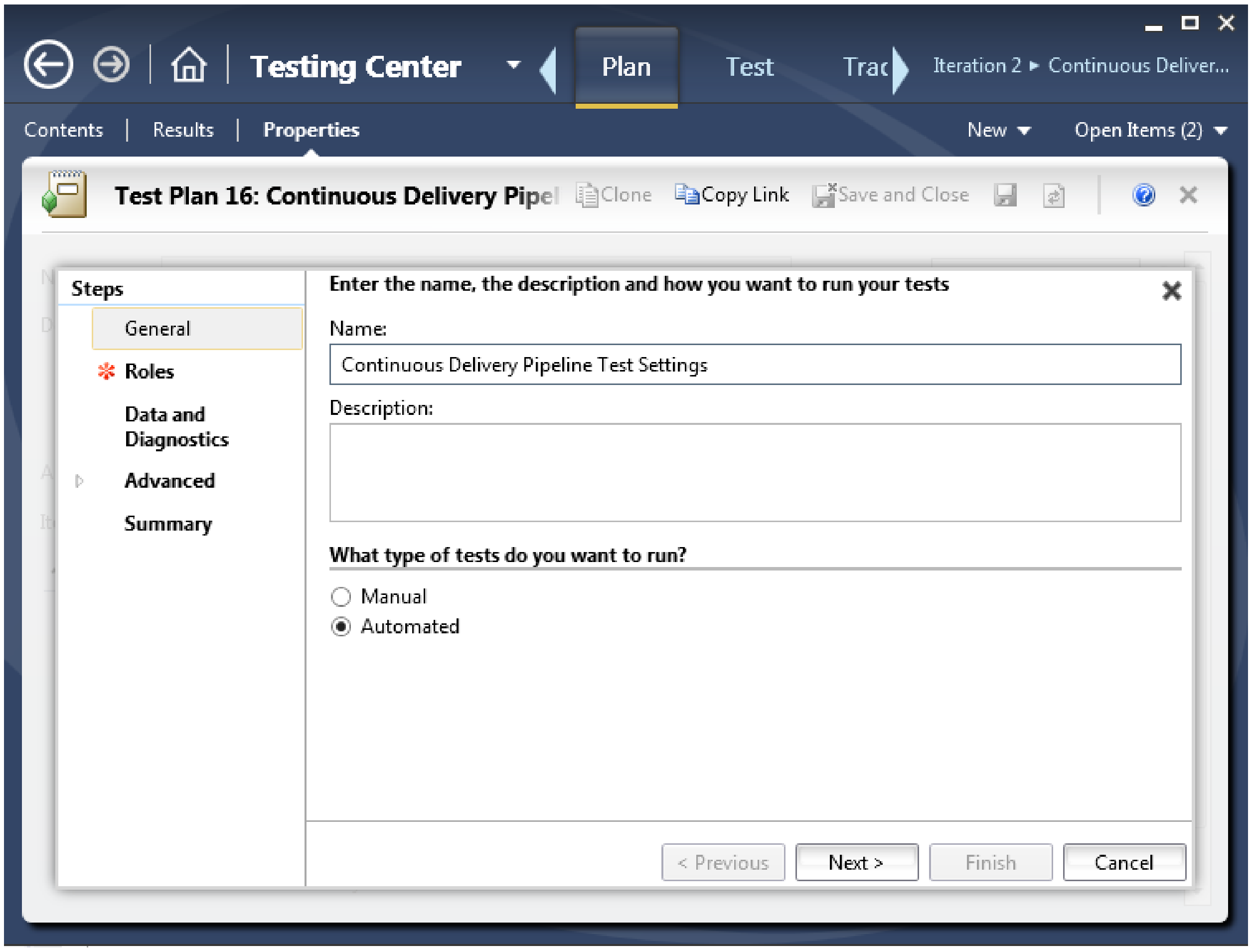
Automated runs:
Test settings: <Default> Open
<Default>
<New...>
Test environment:

Builds:
Filter for builds: Any definition or quality
Build in use: None Modify

Configurations:
In this plan: Windows 8

Links (0)

5. In the **General** tab, in the **Name** text box, enter **Continuous Delivery Pipeline Test Settings**. Leave the **Automated** option selected. Click **Next**.



Test Plan 16: Continuous Delivery Pipel

Enter the name, the description and how you want to run your tests

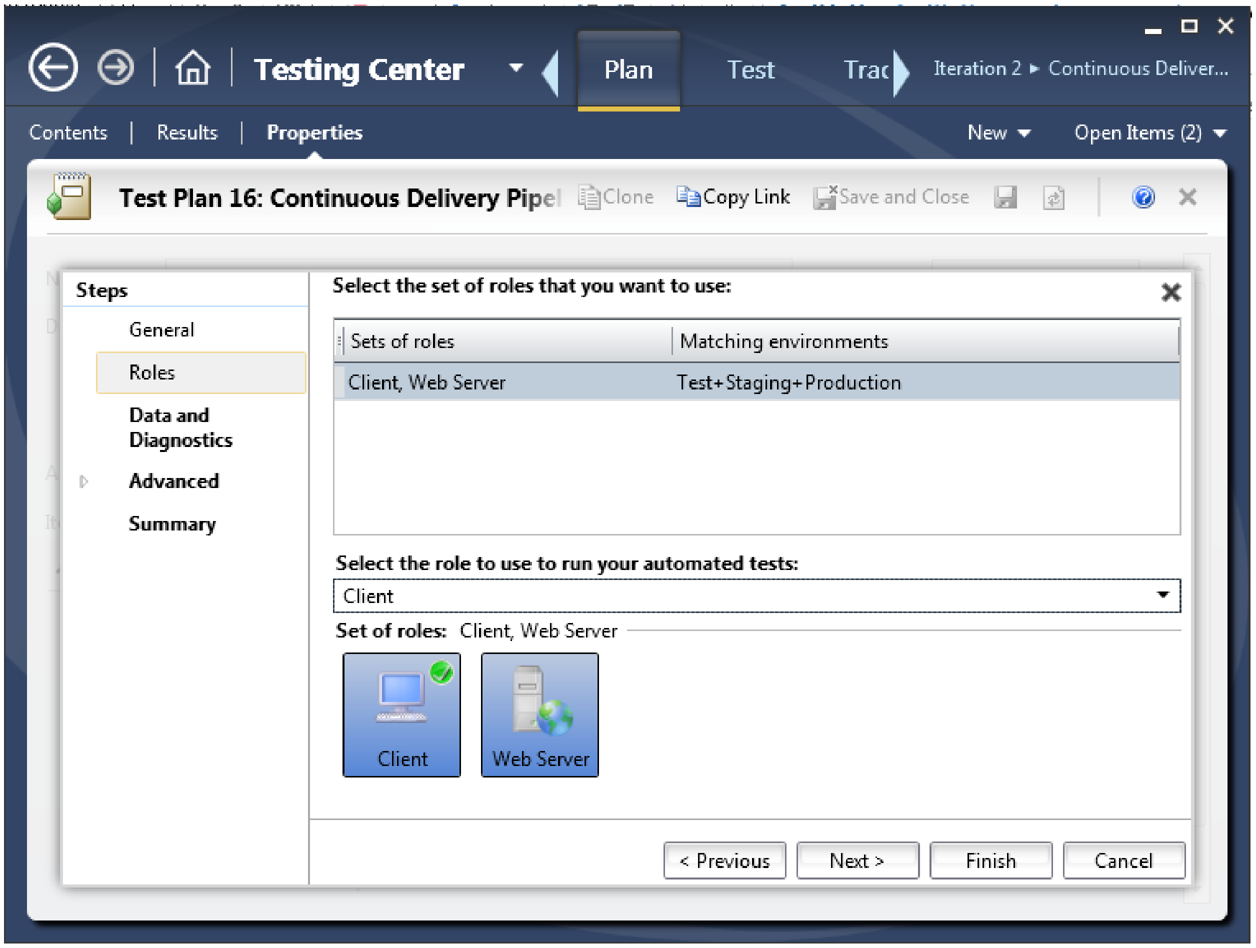
Name: Continuous Delivery Pipeline Test Settings

Description:

What type of tests do you want to run?

Manual [✓ Automated]

6. The **Roles** tab is selected. Select **Client, Web Server** under **Sets of roles**. Select **Client** from the **Select the role to use to run your automated tests** drop-down list. Click **Next**.



Select the set of roles that you want to use:

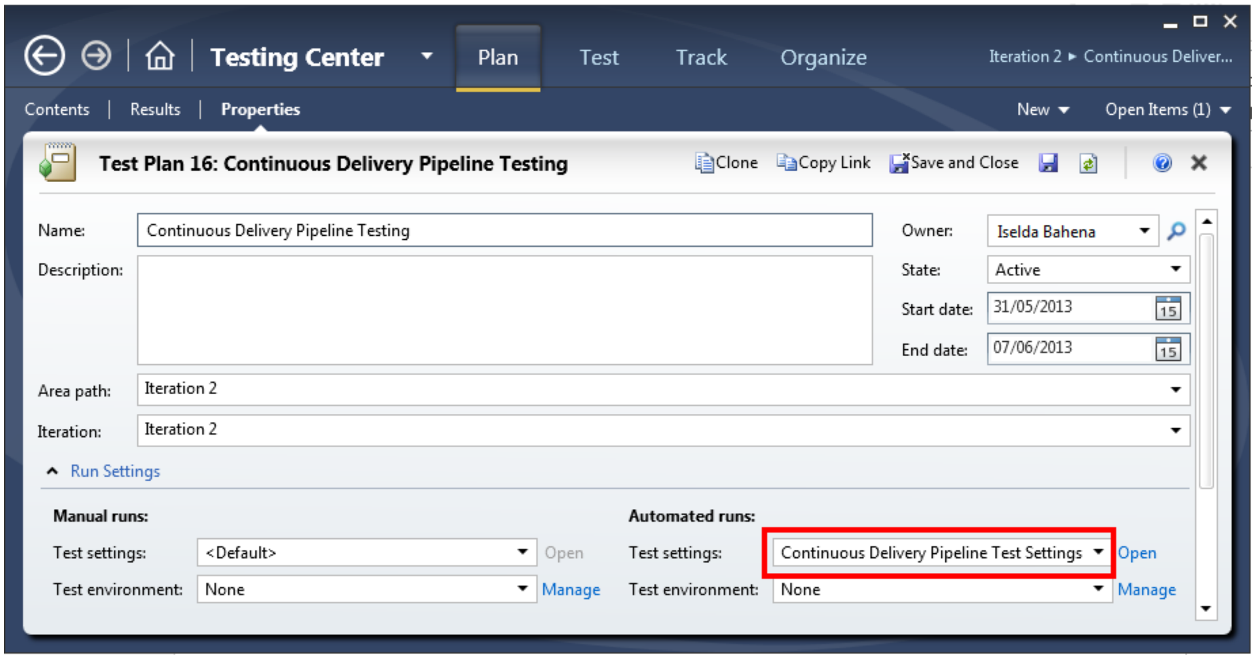
Sets of roles	Matching environments
Client, Web Server	Test+Staging+Production

Select the role to use to run your automated tests: Client

Set of roles: Client, Web Server

7. The **Data and Diagnostics** tab is selected. Leave the defaults. This means only system information will be collected. Click **Next**.

8. The **Advanced** tab is selected. Click **Finish**. The **Test Plan** page for the **Continuous Delivery Pipeline Testing** plan appears. Save the plan.



Test Plan 16: Continuous Delivery Pipeline Testing

Name: Continuous Delivery Pipeline Testing
Description:
Owner: Iselda Bahena
State: Active
Start date: 31/05/2013
End date: 07/06/2013
Area path: Iteration 2
Iteration: Iteration 2

Run Settings

Manual runs:
Test settings: <Default> Open
Test environment: None Manage

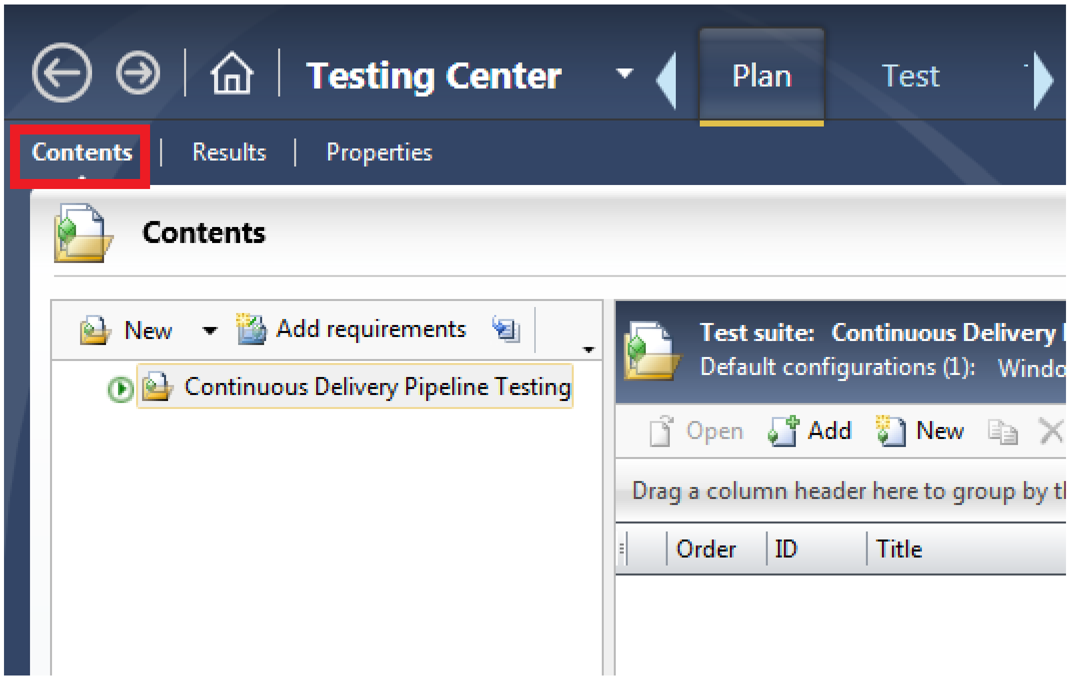
Automated runs:
Test settings: Continuous Delivery Pipeline Test Settings Open
Test environment: None Manage

Task 2: Create Test Suites and Add Test Cases

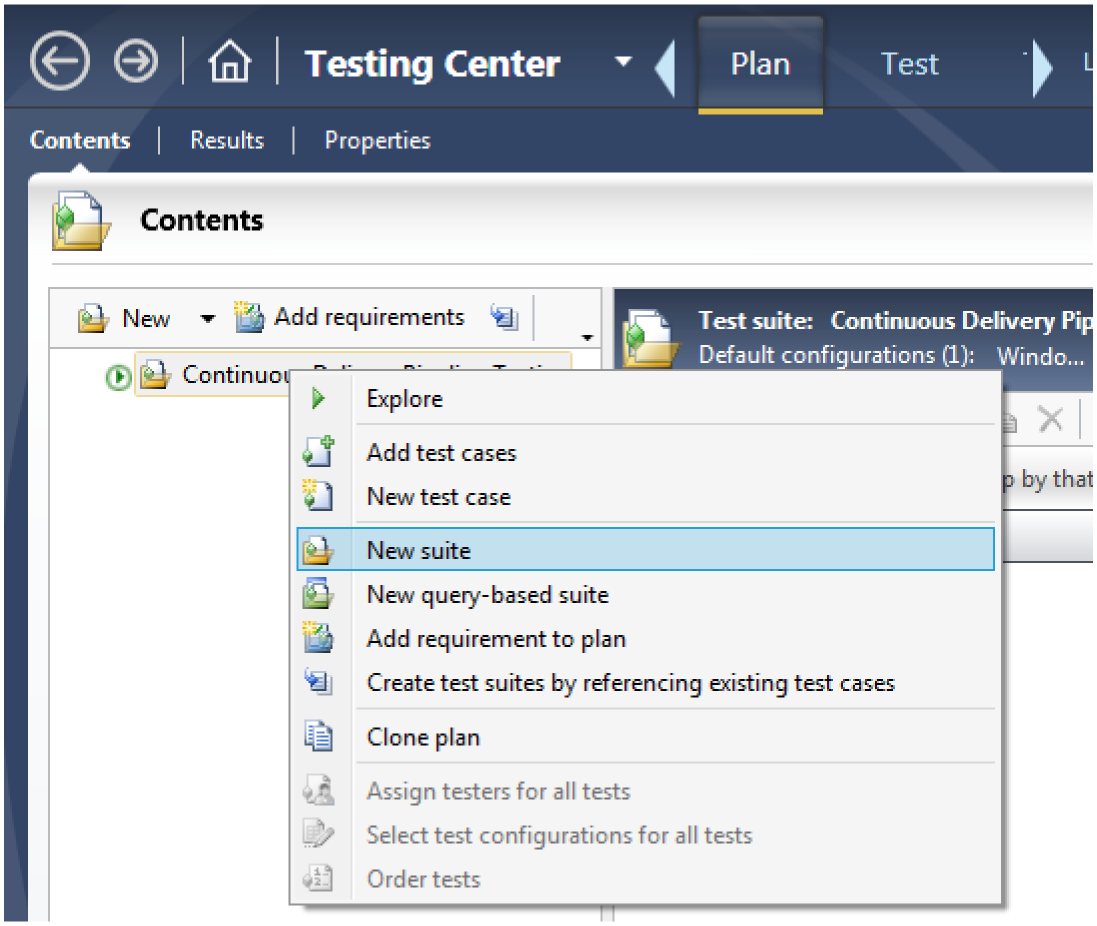
In this task you organize the test cases into test suites that are included in the test plan.

There will be one test suite for the acceptance tests, which run in the acceptance test stage of the pipeline. There will be another test suite for the build verification tests, which run in every stage but the commit stage. The build verification tests validate that each automated deployment succeeded. For more information about how to define and work with test suites, see Organizing Test Cases Using Test Suites.

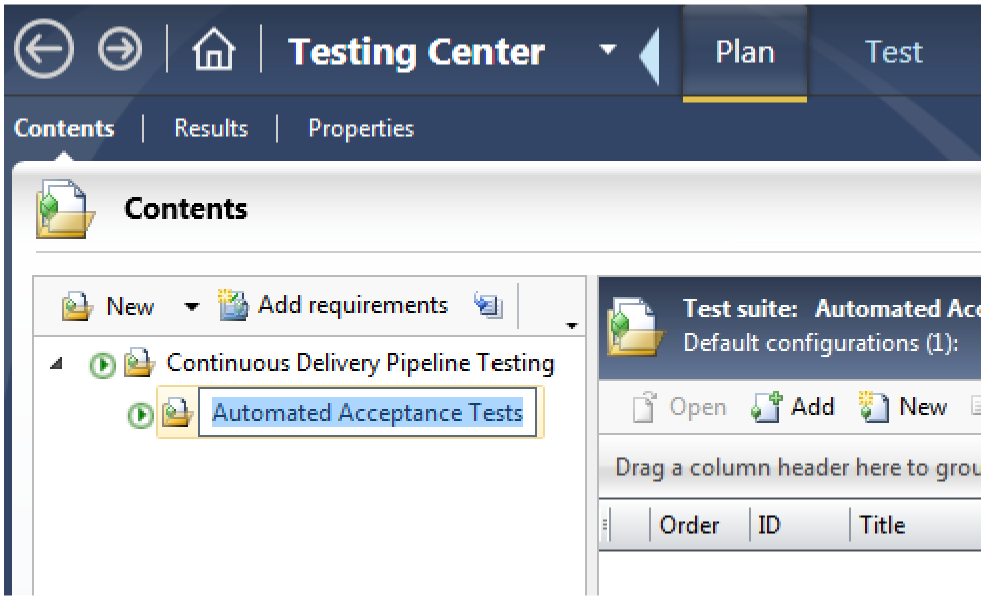
1. Open the Continuous Delivery Pipeline Testing plan. Make sure you're in the **Contents** section of the **Plan** tab. The following screenshot shows an example of what you should see.


Testing Center
Plan
Test
Contents
Results
Properties
Contents
New
Add requirements
Continuous Delivery Pipeline Testing
Test suite: Continuous Delivery
Default configurations (1): Windo
Open
Add
New
Drag a column header here to group by t
Order
ID
Title

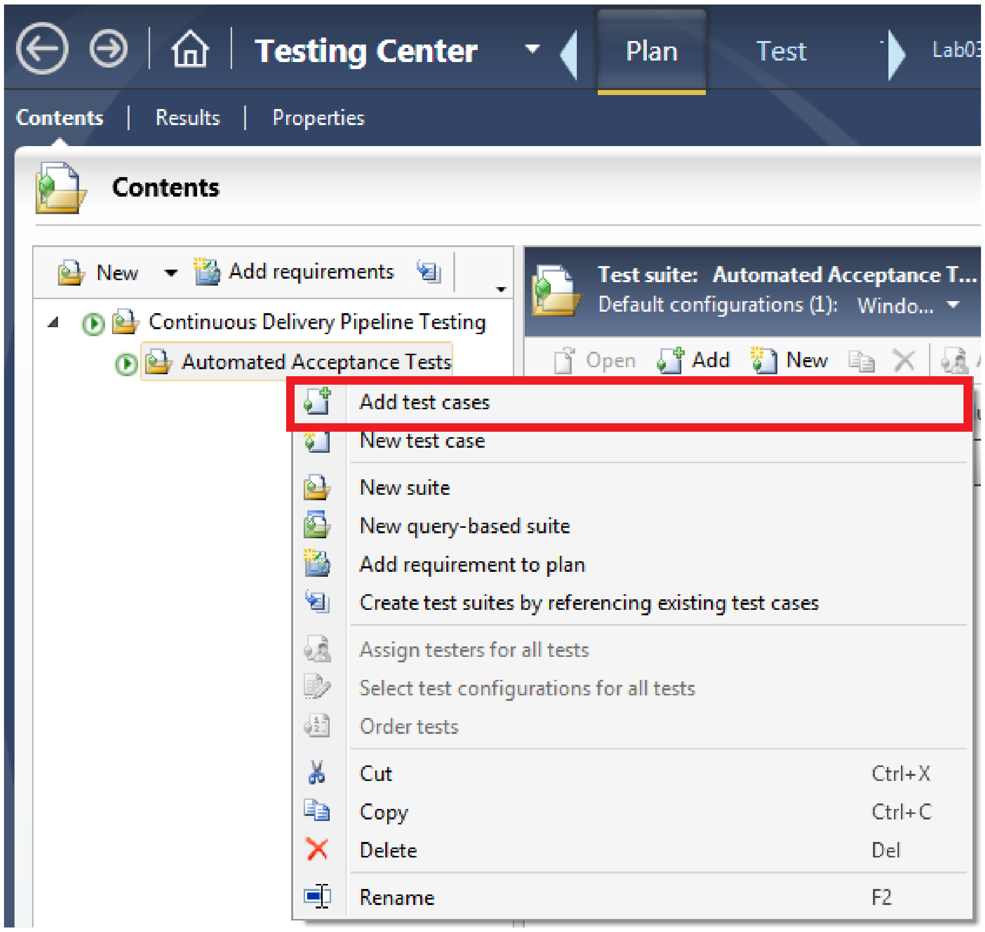
2. Right-click on **Continuous Delivery Pipeline Testing** and select **New suite**.


Testing Center
Plan
Test
Contents
Results
Properties
Contents
New
Add requirements
Test suite: Continuous Delivery Pip
Default configurations (1): Windo...
Explore
Add test cases
New test case
New suite
New query-based suite
Add requirement to plan
Create test suites by referencing existing test cases
Clone plan
Assign testers for all tests
Select test configurations for all tests
Order tests

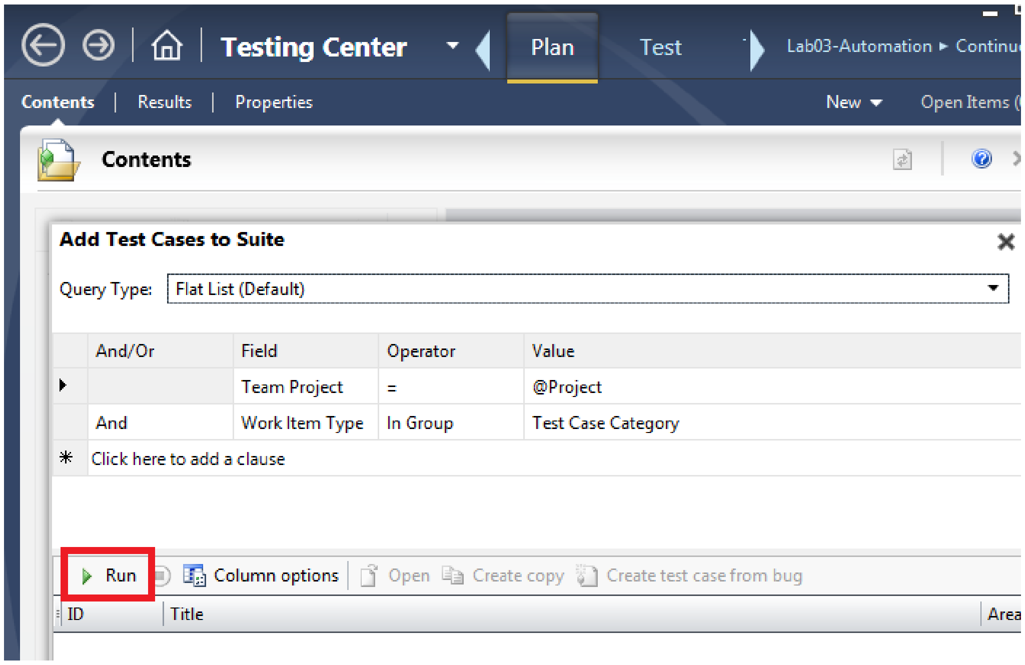
3. A new test suite is created. Name it **Automated Acceptance Tests**.


Testing Center
Plan
Test
Contents
Results
Properties
Contents
New
Add requirements
Continuous Delivery Pipeline Testing
Automated Acceptance Tests
Test suite: Automated Acc
Default configurations (1):
Open
Add
New
Drag a column header here to grou
Order
ID
Title

4. Right-click on the test suite and select **Add test cases**.


Testing Center
Plan
Test
Lab03
Contents
Results
Properties
Contents
New
Add requirements
Continuous Delivery Pipeline Testing
Automated Acceptance Tests
Test suite: Automated Acceptance T...
Default configurations (1): Windo...
Open
Add
New
Add test cases
New test case
New suite
New query-based suite
Add requirement to plan
Create test suites by referencing existing test cases
Assign testers for all tests
Select test configurations for all tests
Order tests
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Rename F2

5. A query windows opens. Click **Run** to perform a query. The following screenshot shows an example.



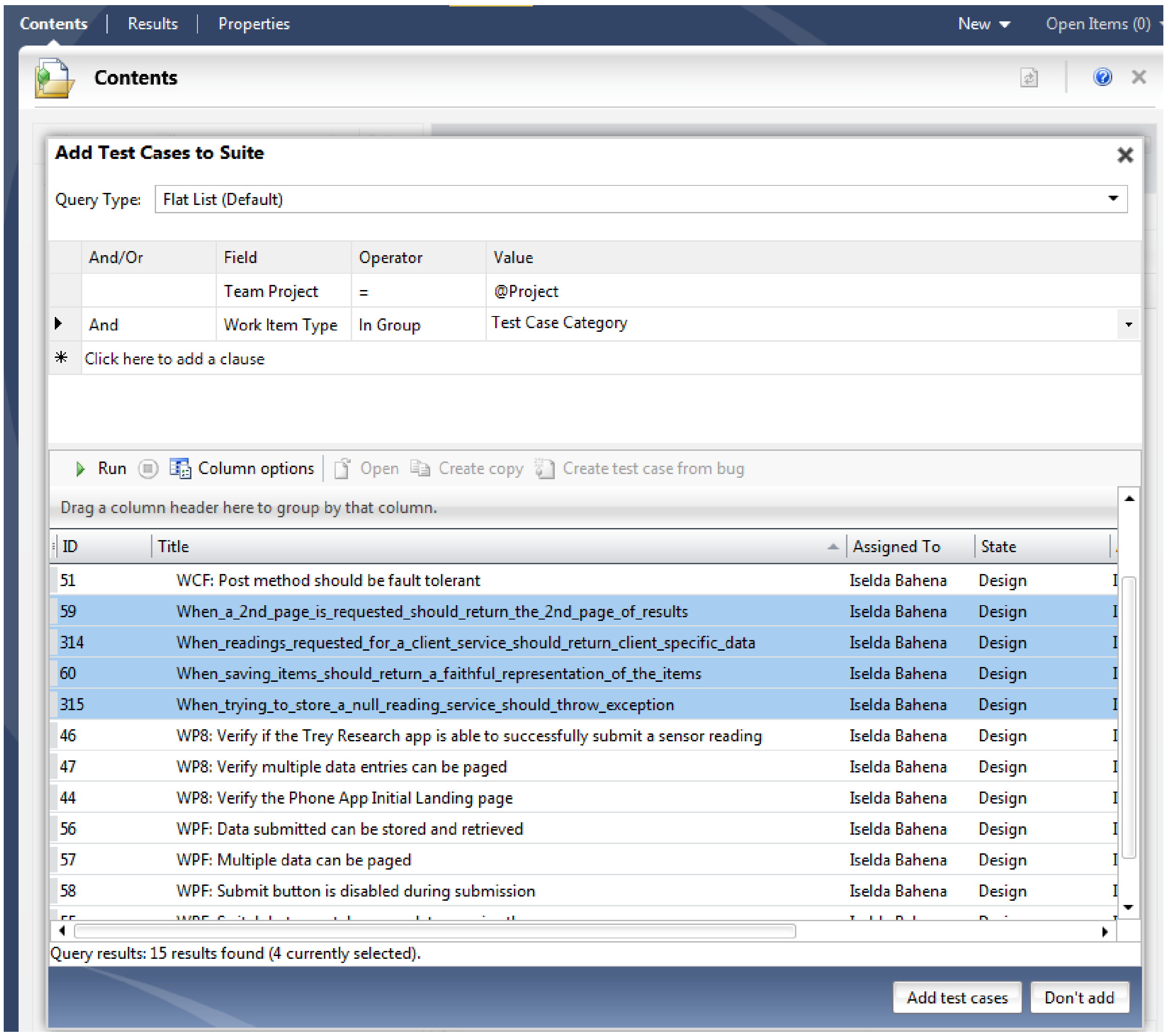
Add Test Cases to Suite

Query Type: Flat List (Default)

And/Or	Field	Operator	Value
	Team Project	=	@Project
And	Work Item Type	In Group	Test Case Category
Click here to add a clause			

6. From the results, choose these four test cases and then click **Add test cases**.

   - When_saving_items_should_return_a_faithful_representation_of_the_items
   - When_trying_to_store_a_null_reading_service_should_throw_exception
   - When_readings_requested_for_a_client_service_should_return_client_specific_data
   - When_a_2nd_page_is_requested_should_return_the_2nd_page_of_result


Contents | Results | Properties
New
Open Items (0)
Contents
Add Test Cases to Suite

Query Type: Flat List (Default)

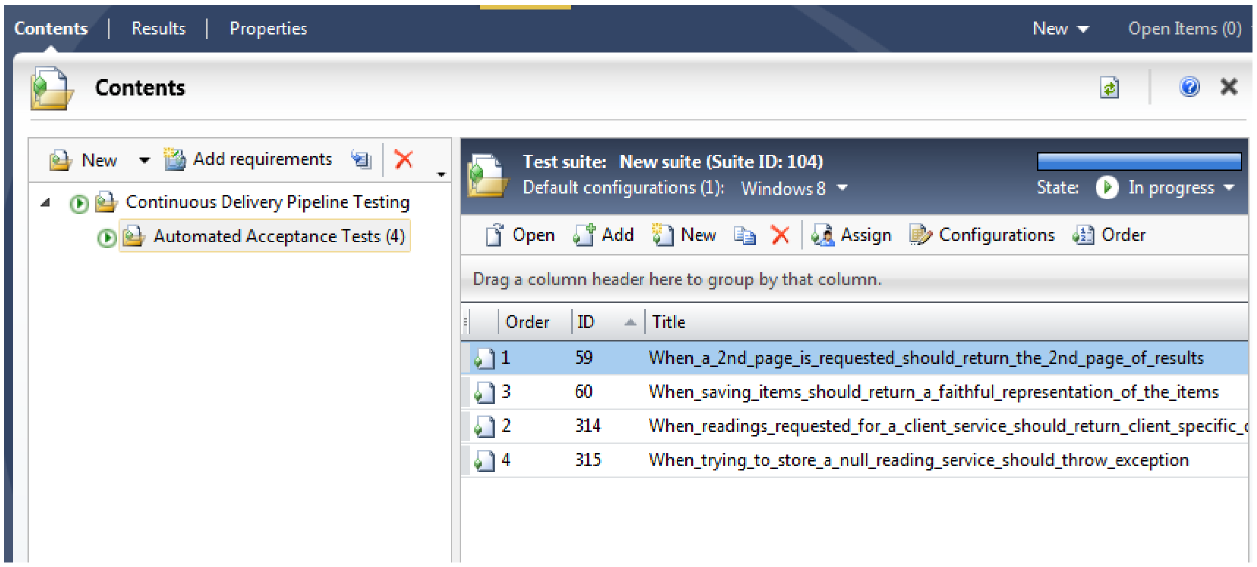
	And/Or	Field	Operator	Value
		Team Project	=	@Project
▶	And	Work Item Type	In Group	Test Case Category
✳	Click here to add a clause			

Run
Column options
Open
Create copy
Create test case from bug
Drag a column header here to group by that column.

ID	Title	Assigned To	State
51	WCF: Post method should be fault tolerant	Iselda Bahena	Design
59	When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results	Iselda Bahena	Design
314	When_readings_requested_for_a_client_service_should_return_client_specific_data	Iselda Bahena	Design
60	When_saving_items_should_return_a_faithful_representation_of_the_items	Iselda Bahena	Design
315	When_trying_to_store_a_null_reading_service_should_throw_exception	Iselda Bahena	Design
46	WP8: Verify if the Trey Research app is able to successfully submit a sensor reading	Iselda Bahena	Design
47	WP8: Verify multiple data entries can be paged	Iselda Bahena	Design
44	WP8: Verify the Phone App Initial Landing page	Iselda Bahena	Design
56	WPF: Data submitted can be stored and retrieved	Iselda Bahena	Design
57	WPF: Multiple data can be paged	Iselda Bahena	Design
58	WPF: Submit button is disabled during submission	Iselda Bahena	Design

Query results: 15 results found (4 currently selected).

7. You should see that the test cases are now included in the Automated Acceptance Tests test suite.

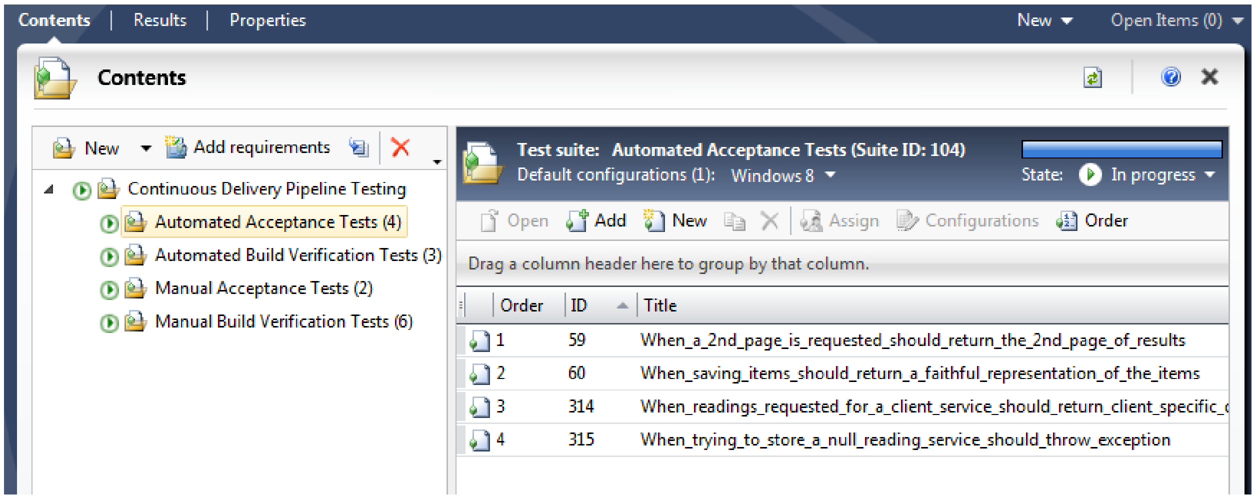


Test suite: New suite (Suite ID: 104)
Default configurations (1): Windows 8
State: In progress

Drag a column header here to group by that column.

Order	ID	Title
1	59	When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results
3	60	When_saving_items_should_return_a_faithful_representation_of_the_items
2	314	When_readings_requested_for_a_client_service_should_return_client_specific_
4	315	When_trying_to_store_a_null_reading_service_should_throw_exception

8. Perform steps 2 through 7 for a new test suite named Automated Build Verification Tests. Add the following test cases to the test suite:
   - WPF: Verify if the Trey research client is launched successfully
   - WPF: Switch between tab pages, data in Recording reading tab should change
   - WPF: Data submitted can be stored and retrieved
9. Add another test suite named Manual Acceptance Tests and add the following test cases to the test suite:
   - WPF: Multiple data can be paged
   - WPF: Submit button is disabled during submission
10. Add another test suite named Manual Build Verification Tests and add the following test cases to the test suite:
    - WCF: Post method should be fault tolerant
    - WCF: Get method should be fault tolerant
    - WCF: Invalid page request should be handled gracefully
11. The following screenshot shows what the final test plan hierarchy should look like.

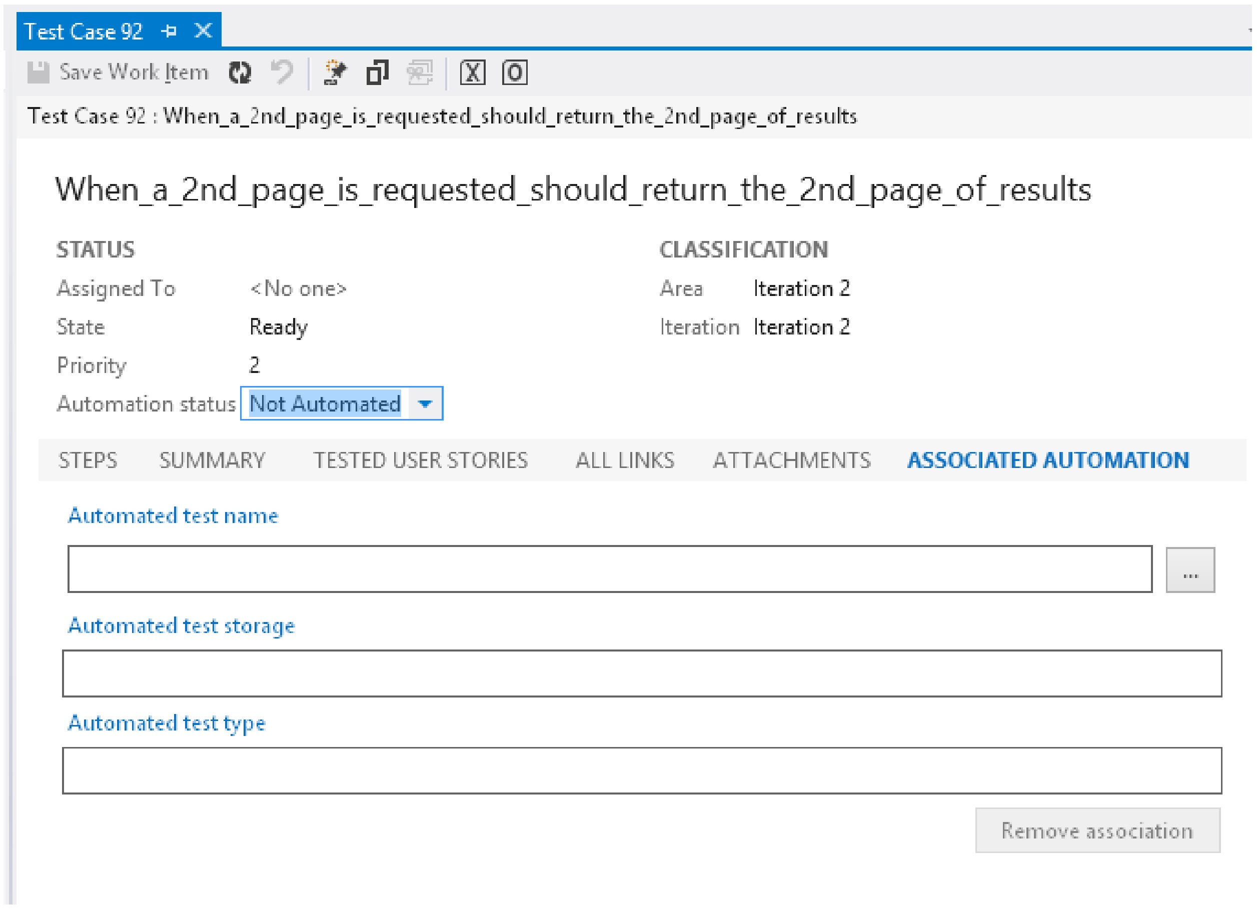

Contents | Results | Properties
New
Open Items (0)
Contents
New
Add requirements
Continuous Delivery Pipeline Testing
Automated Acceptance Tests (4)
Automated Build Verification Tests (3)
Manual Acceptance Tests (2)
Manual Build Verification Tests (6)
Test suite: Automated Acceptance Tests (Suite ID: 104)
Default configurations (1): Windows 8
State: In progress
Open
Add
New
Assign
Configurations
Order
Drag a column header here to group by that column.

Order	ID	Title
1	59	When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results
2	60	When_saving_items_should_return_a_faithful_representation_of_the_items
3	314	When_readings_requested_for_a_client_service_should_return_client_specific_
4	315	When_trying_to_store_a_null_reading_service_should_throw_exception

Task 3: Associate Automated Tests with Test Cases

In this task you associate each automated test to a corresponding test case. For more information about associating an automated test to a test case, see How to: Associate an Automated Test with a Test Case.

1. In Visual Studio, open the **TreyResearch** solution. Build it to so that you can see the tests in Test Explorer.
2. In Team Explorer, select **Work Items**. Search the titles for all the test cases that must be associated with the automated tests. For example, search for the test case **When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results**.
3. Click **OK** to open the work item.
4. To add the automated test, select the **ASSOCIATED AUTOMATION** tab.



Test Case 92 : When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results

When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results

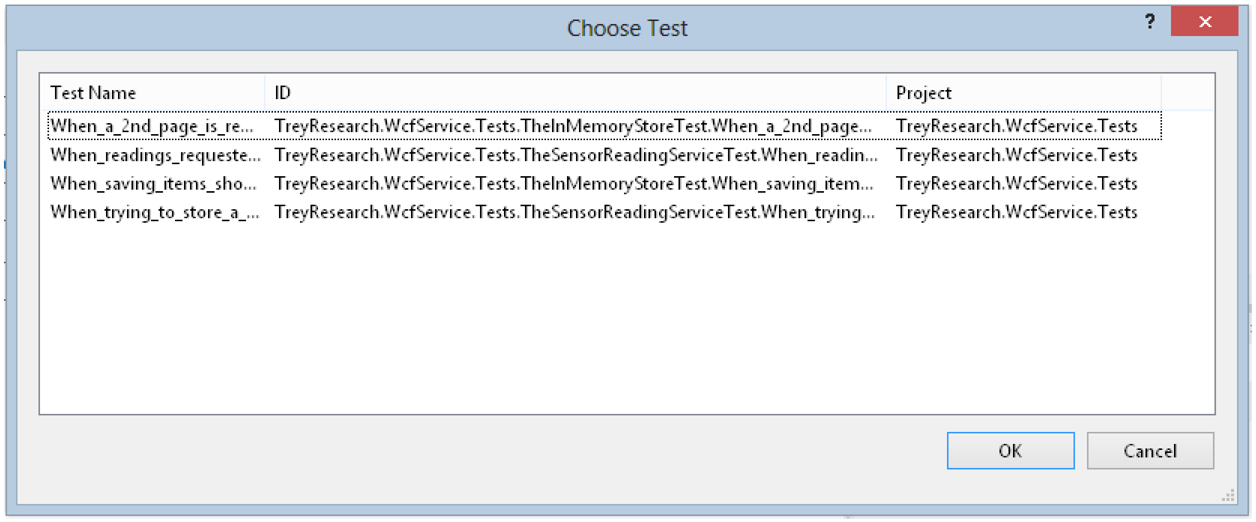
STATUS

Assigned To	<No one>
State	Ready
Priority	2
Automation status	Not Automated

CLASSIFICATION

Area	Iteration 2
Iteration	Iteration 2

5. To find the automated test name, select the **ellipses (…)**. The **Choose Test** dialog box appears. All the tests in the solution are shown together with their associated test projects.


Choose Test

Test Name	ID	Project
When_a_2nd_page_is_re...	TreyResearch.WcfService.Tests.TheInMemoryStoreTest.When_a_2nd_page...	TreyResearch.WcfService.Tests
When_readings_requeste...	TreyResearch.WcfService.Tests.TheSensorReadingServiceTest.When_readin...	TreyResearch.WcfService.Tests
When_saving_items_sho...	TreyResearch.WcfService.Tests.TheInMemoryStoreTest.When_saving_item...	TreyResearch.WcfService.Tests
When_trying_to_store_a_...	TreyResearch.WcfService.Tests.TheSensorReadingServiceTest.When_trying...	TreyResearch.WcfService.Tests

6. Select the **When_a_2nd_page_is_requested_should_return_the_2nd_page_of_results** test case. Click **OK**.

7. Repeat steps 2 through 6 for all the test cases that you want the pipeline to run. For this lab, complete steps 2 through 6 for all test cases in the **Automated Build Verification Tests** and **Automated Acceptance Tests** test suites.

NOTE: In How to: Associate an Automated Test with a Test Case, there is a section, **Set Up Your Test Plan to Use Your Team Build**, which is not in this HOL. To run automated tests from within MTM, you must complete the steps in that section, and first associate a specific build to your test plan. This is so that the test runner can find the assemblies that contain the tests in the build drop folder.

This HOL does not run the automated tests from within MTM. Instead, it customizes the build definition based on the Lab Management template. The test assemblies are loaded from the drop folder and are already defined for the build. This is why you do not assign a build to the test plan as the documentation recommends.

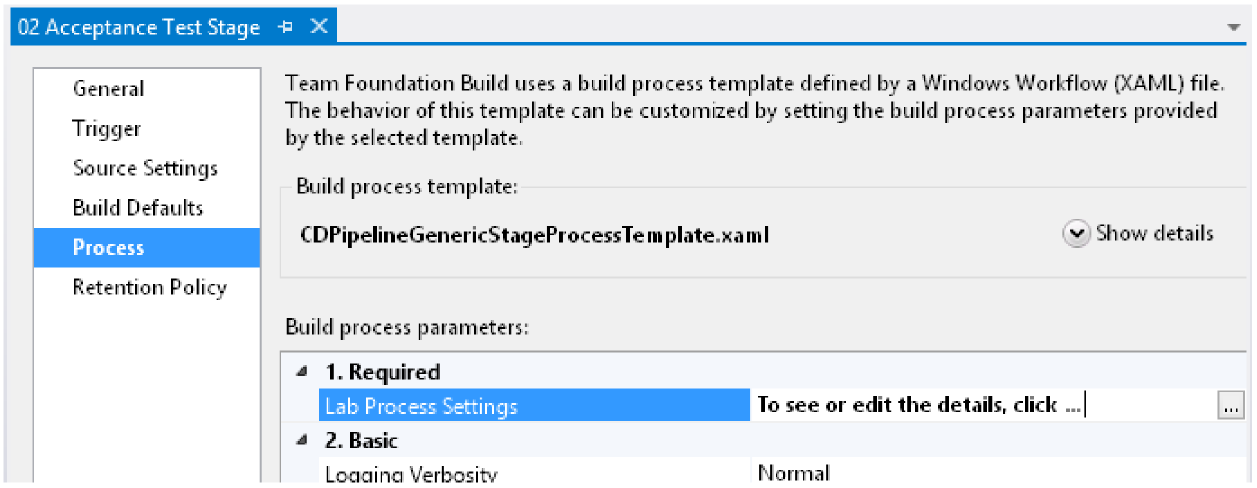
Exercise 5: Configuring and Testing the Subsequent Stages

In this task you configure the remaining pipeline stages to run automated tests. You also verify that the tests run.

Task 1: Set Up the Subsequent Stages to Run Automated Tests

In this task, you configure the remaining pipeline stages to run automated tests.

1. In Team Explorer, edit the build definition for **02 Acceptance Test Stage**. Select the **Process** tab. Go to the **Lab Process Settings** parameter and click the ellipses **(…)**.



Team Foundation Build uses a build process template defined by a Windows Workflow (XAML) file. The behavior of this template can be customized by setting the build process parameters provided by the selected template.

Build process template:

CDPipelineGenericStageProcessTemplate.xaml

Build process parameters:

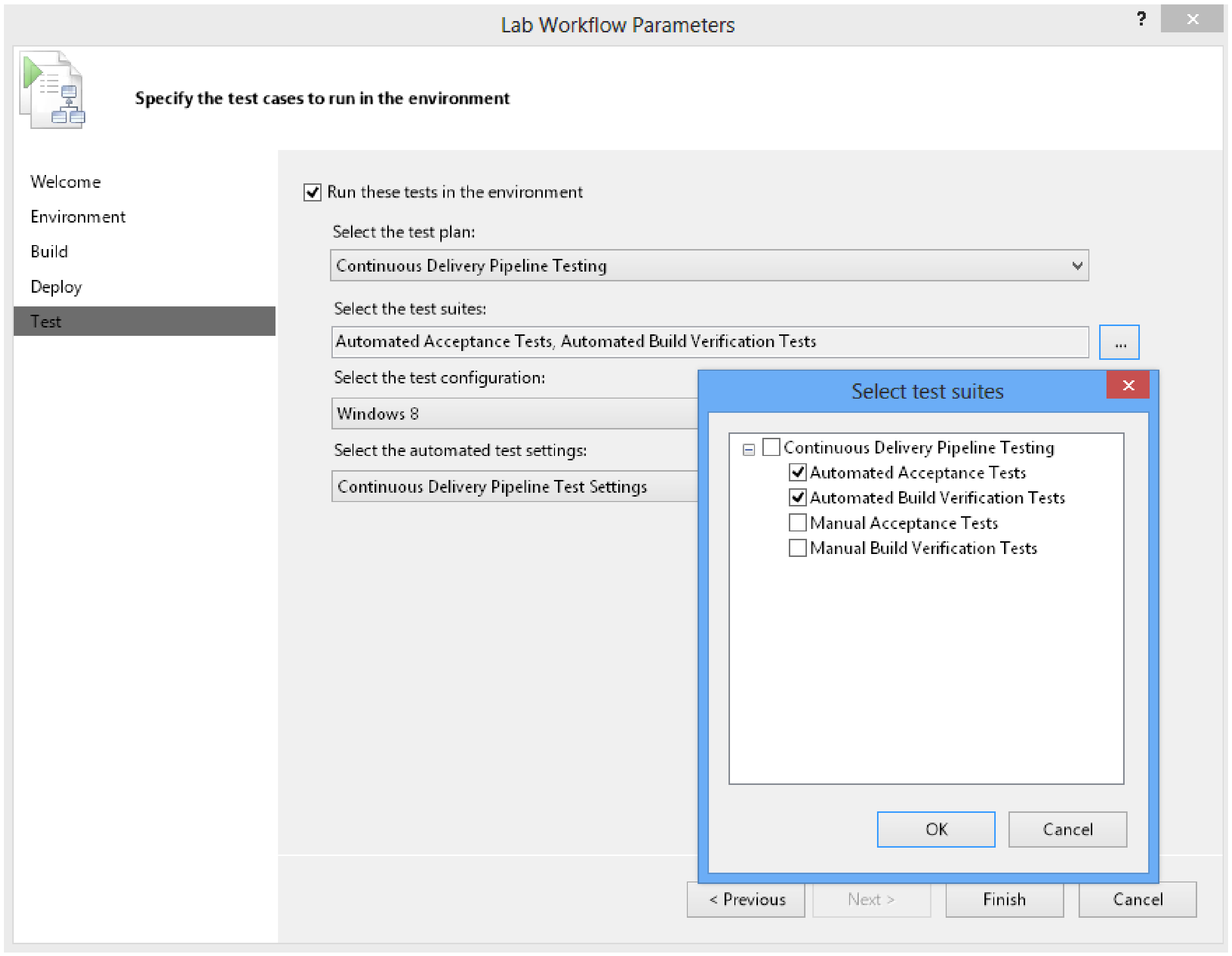
1. Required
Lab Process Settings | To see or edit the details, click ...
2. Basic
Logging Verbosity | Normal

2. The **Lab Workflow Parameters** wizard opens. Select the **Test** tab.

3. Make sure that the **Run these tests in the environment** option is selected.

4. Select **Continuous Delivery Pipeline Testing** from the **Select the test plan** drop-down list.

5. Go to the **Select the test suites** drop-down list and click the ellipses **(…)**. The **Select test suites** dialog box opens. Select **Automated Acceptance Tests** and **Automated Build Verification Tests**.
6. Select **Windows 8** from the **Select the test configuration** drop-down list.
7. Select **Continuous Delivery Pipeline Test Settings** from the **Select the automated test settings** drop-down list.


Lab Workflow Parameters

Specify the test cases to run in the environment

☑ Run these tests in the environment

Select the test plan: Continuous Delivery Pipeline Testing

Select the test suites: Automated Acceptance Tests, Automated Build Verification Tests

Select the test configuration: Windows 8

Select the automated test settings: Continuous Delivery Pipeline Test Settings

Select test suites

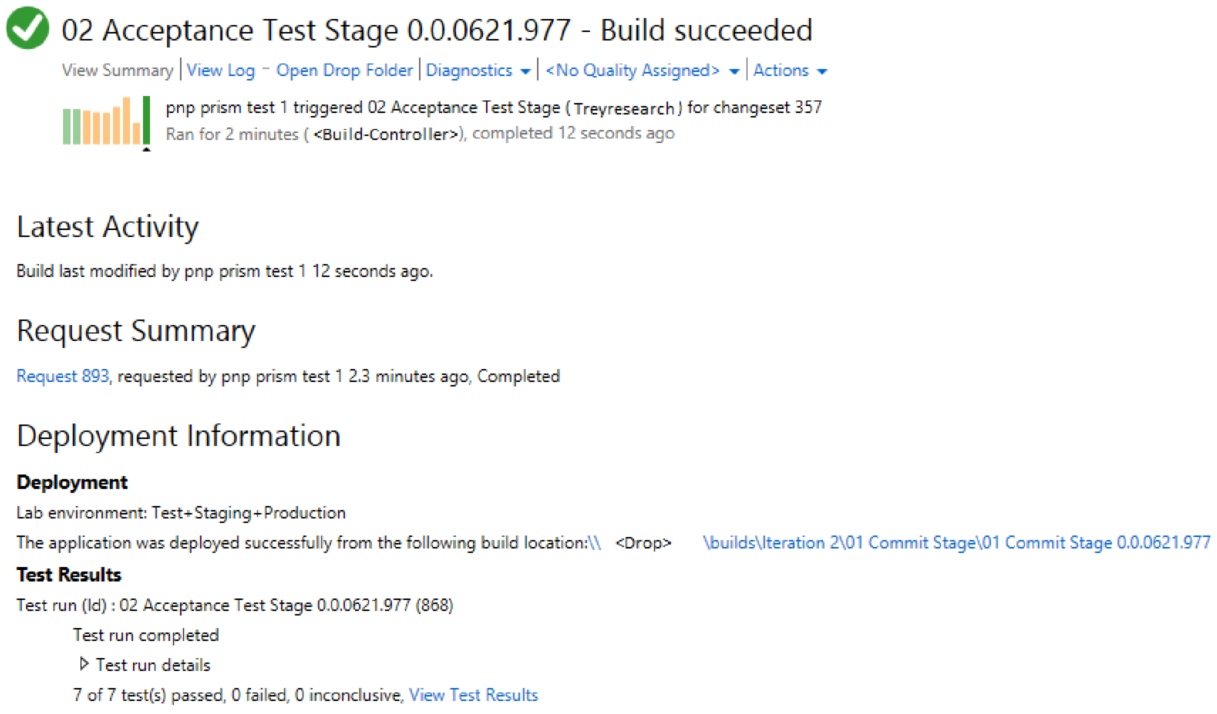
☐ Continuous Delivery Pipeline Testing
☑ Automated Acceptance Tests
☑ Automated Build Verification Tests
☐ Manual Acceptance Tests
☐ Manual Build Verification Tests

8. Click **Finish**. Save and close the build definition.
9. Repeat steps 1 through 8 for **03a Release Stage** and **03b UAT Stage**, however you only need to select **Automated Build Verification Tests** for these stages.

Task 2: Verify the Configurations

In this task you verify that the pipeline stages are configured correctly and that the automated tests run.

1. Trigger **01 Commit Stage**. It will trigger **02 Acceptance Test Stage** after it completes. When the acceptance test stage completes, open the build summary. You should see that the test cases within the test suites have run.



View Summary | View Log - Open Drop Folder | Diagnostics | <No Quality Assigned> | Actions

02 Acceptance Test Stage 0.0.0621.977 - Build succeeded

pnp prism test 1 triggered 02 Acceptance Test Stage (Treyresearch) for changeset 357
Ran for 2 minutes (<Build-Controller>), completed 12 seconds ago

Latest Activity

Build last modified by pnp prism test 1 12 seconds ago.

Request Summary

Request 893, requested by pnp prism test 1 2.3 minutes ago, Completed

Deployment Information

Deployment

Lab environment: Test+Staging+Production

The application was deployed successfully from the following build location:\\ <Drop> \builds\Iteration 2\01 Commit Stage\01 Commit Stage 0.0.0621.977

Test Results

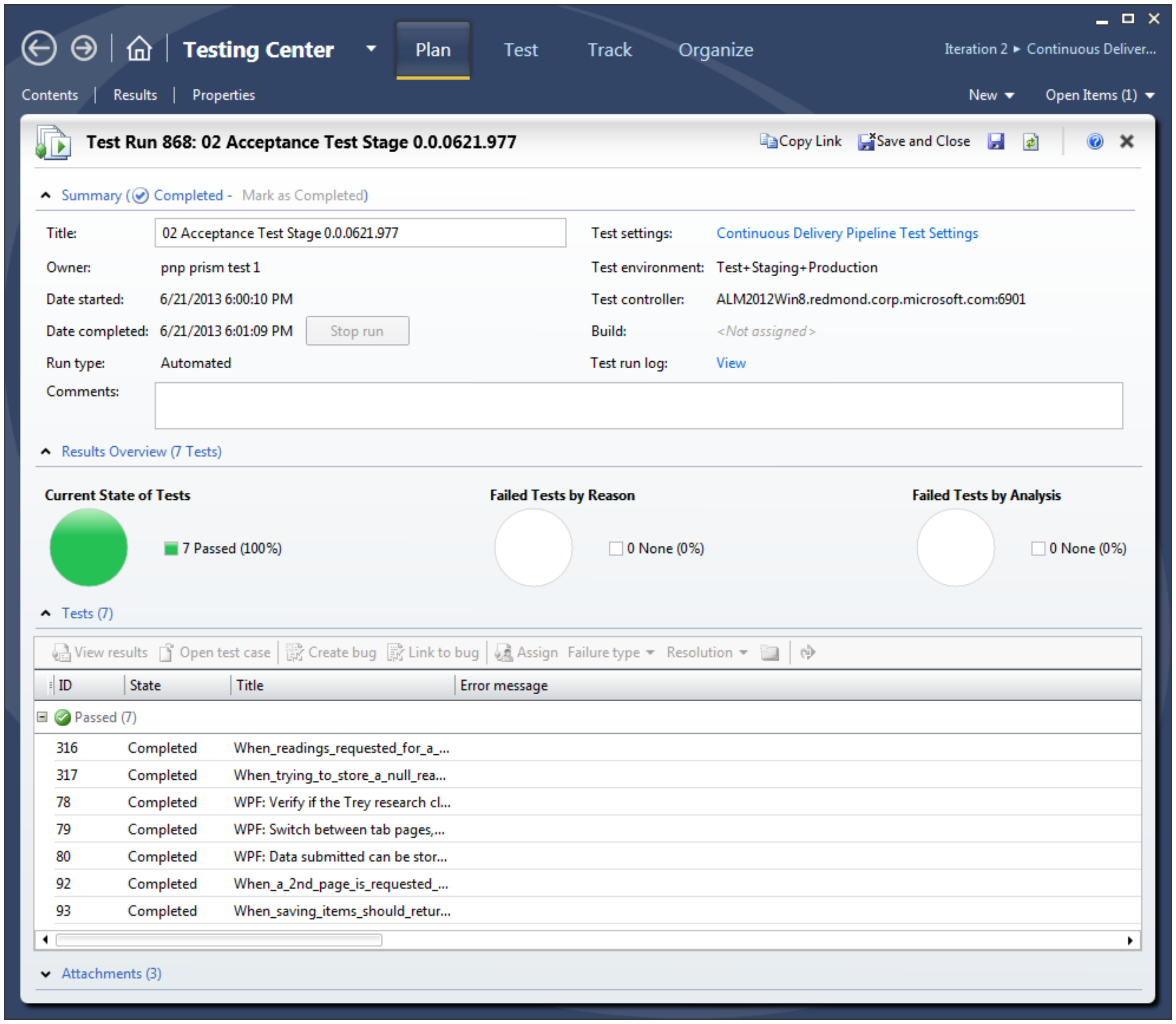
Test run (Id) : 02 Acceptance Test Stage 0.0.0621.977 (868)

Test run completed

▷ Test run details

7 of 7 test(s) passed, 0 failed, 0 inconclusive, View Test Results

2. Click **View Test Results** to see detailed information about each test and its result.



Test Run 868: 02 Acceptance Test Stage 0.0.0621.977

Summary (Completed - Mark as Completed)

Title:	02 Acceptance Test Stage 0.0.0621.977	Test settings:	Continuous Delivery Pipeline Test Settings
Owner:	pnp prism test 1	Test environment:	Test+Staging+Production
Date started:	6/21/2013 6:00:10 PM	Test controller:	ALM2012Win8.redmond.corp.microsoft.com:6901
Date completed:	6/21/2013 6:01:09 PM	Build:	<Not assigned>
Run type:	Automated	Test run log:	View
Comments:			

Results Overview (7 Tests)

Current State of Tests: 🟩 7 Passed (100%)

Failed Tests by Reason: ☐ 0 None (0%)

Failed Tests by Analysis: ☐ 0 None (0%)

Tests (7)

Passed (7)

ID	State	Title	Error message
316	Completed	When_readings_requested_for_a_...	
317	Completed	When_trying_to_store_a_null_rea...	
78	Completed	WPF: Verify if the Trey research cl...	
79	Completed	WPF: Switch between tab pages,...	
80	Completed	WPF: Data submitted can be stor...	
92	Completed	When_a_2nd_page_is_requested_...	
93	Completed	When_saving_items_should_retur...	

Attachments (3)

3. Run either of the manually triggered stages, such as **03b UAT Stage**. Look at the test results.

Summary

In this HOL you made it possible for the pipeline to run automated tests. You modified the Lab Management environment so that it has access to a user interface. You then set up automated testing for the commit stage. The commit stage differs from the other pipeline stages because it only runs unit tests. Other stages in the pipelinc can run a variety of tests. You used a filter to categorize the unit tests and modified the commit stage build definition to run automated tests.

You next set up automated testing for the remaining pipeline stages. You first defined the test settings. You then used MTM to create test suites and test cases. After that, you used Visual Studio to associate the automated tests with the corresponding test cases. When that was done, you modified the build definitions of the pipeline stages. Finally, you verified that the automated tests run correctly.

Copyright