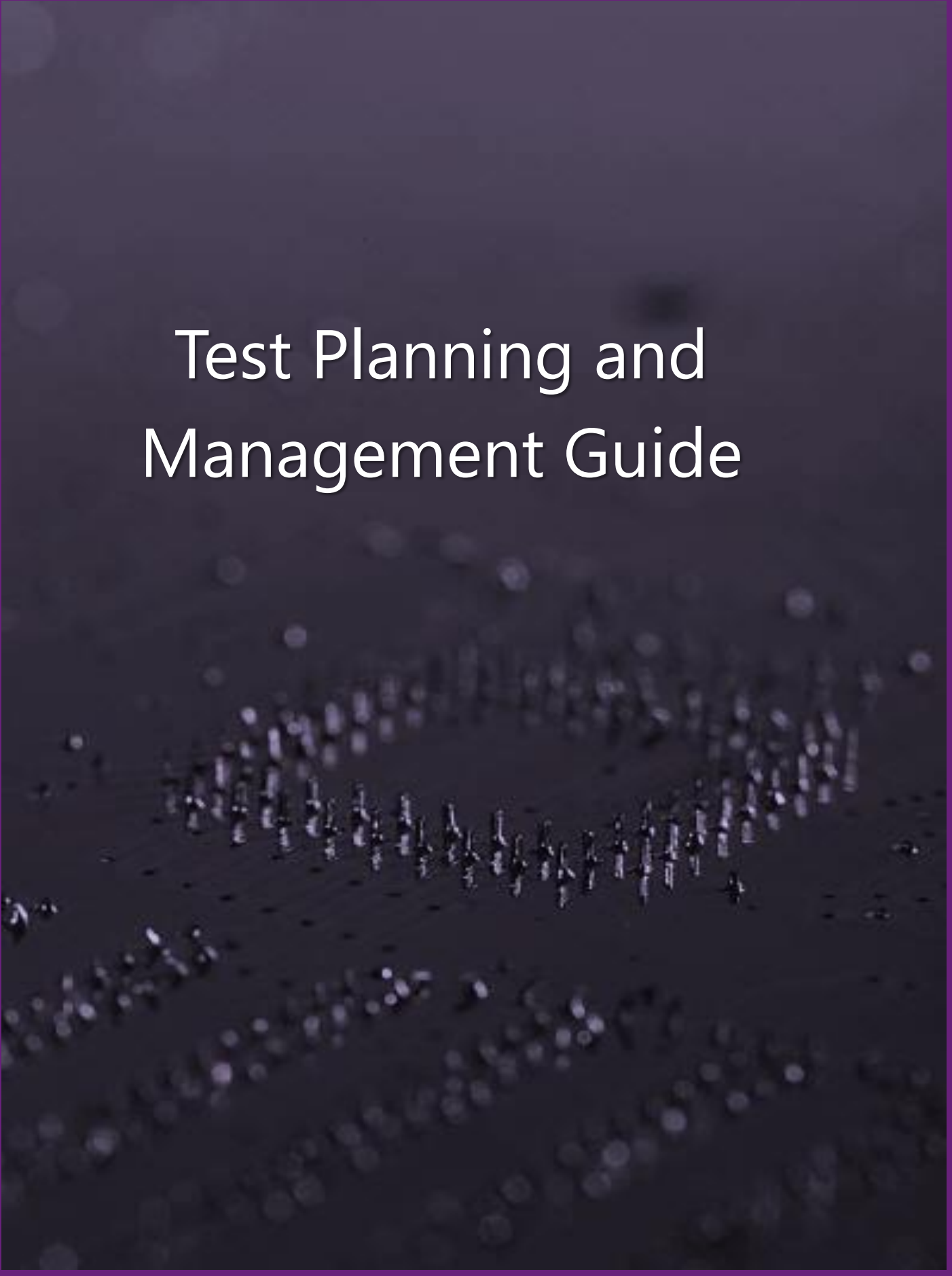


Test Planning and Management Guide

V2.0.1

Visual Studio ALM Rangers

Microsoft

Visual Studio

The MIT License (MIT)

Table of Contents

Foreword

Application Lifecycle Management with Team Foundation Server (TFS) is all about leveraging an integrated toolset to manage your software projects, from planning and development through testing and deployment. With the goal to help teams ship software with high quality, we've built great experiences for manual testing with Microsoft Test Manager (MTM) and Test Hub in TFS Team Web Access. As you will explore in this guide, Test Hub can be looked upon as the evolution of Microsoft Test Manager for manual testing scenarios. With Test Hub in specific and Test Case Management in general, we've made many improvements in the recent releases. Cross platform support has never been better – you can author and run manual tests on any platform with all major browsers supported. We now have first class customization support for customizing test artifacts. One of the key benefits of customization is the flexibility to align the workflows and fields of artifacts used for tracking activities with the business processes used by your organization. This concept can be extended to your test lifecycle, by customizing test plan and test suite work items. Testers spend considerable effort in creating test cases, and creating test cases with the Excel like 'Gird' in Test Hub makes test authoring activity a breeze. Tracking test progress for your team is now possible with lightweight charts, which display test status metrics in real time. You can review your tests with external stakeholders by exporting test plans or test suites.

As testing evolves into an integral phase of ALM, you'll have a lot of questions around effective test planning. What are best practices, based on project size or development methodology? How do I structure my test plans? How to I organize my test cases across different types of test suites? How do I generate appropriate reports per release? How do I carry forward test assets to future sprints and releases?

The Test Planning and Management guide addresses those exact questions and provides a set of best practices, that encompass the entire workflow — from creating test plans to generating test reports for each release. And who better to create the guidance than the experts who use the product extensively across projects of varying nature? The Rangers. Their rich experiences of working across different projects with different software development methodologies, their in-depth understanding of the testing domain, and expertise in using MTM and Test Hub have been invaluable in shaping this guide. This guide covers various perspectives of test planning such as organizing test cases into suites, managing test plans across sprints and releases, setting up regression testing and acceptance testing scenarios, creating insightful reports that reflect test progress, and more. In addition to the core focus on 'Test Planning', this guide also gives you a flavor of new features available in Test Hub and MTM. This guide has a great blend of both, a descriptive summary about different practices for test planning and hands of labs that walk you through the practical aspects of test planning step-by-step. I'd like to thank the Rangers team on accomplishing this comprehensive piece of guidance that will help teams getting started with journey in software testing.

Manoj Bableshwar – Program Manager, Visual Studio Test

Introduction

We guide testers to better plan and execute the project team's testing activities and to enable them to meet testing performance, reporting and configuration management requirements with minimal overhead.

While these goals above can be stated simply and concisely, the practical reality of managing large numbers of test cases over multiple sprints, releases and branches can easily become overwhelming.

Constraints:

- Assumes the Microsoft Visual Studio SCRUM methodology.
- Presents practical guidance, not a general reference.
- Scenarios addressed are simple.

The goal is to equip software testers with a firm understanding of the principles and practices that will enable them to develop better solutions to the testing challenges of their particular projects.

Intended audience

This guide is concerned primarily with one persona, Christine the **Test Lead**. See ALM Rangers Personas and Customer Profiles[1] for more information.

Visual Studio ALM Rangers

The Visual Studio ALM Rangers includes members from the Visual Studio Product group, Microsoft Services, Microsoft Most Valuable Professionals (MVP) and Visual Studio Community Leads. Their mission is to provide out-of-band solutions to missing features and guidance. A growing Rangers Index is available online.

Home	aka.ms/vsarunderstand
Solutions	aka.ms/vsarsolutions
Membership	aka.ms/vsarindex

Contributing ALM Rangers

Bob Hardister, Dan Marzolini, David V. Corbin, Derek Keeler, Doug Owens, Hassan Fadili, Mattias Sköld, Michael Pedersen, Niel Zeeman, Oscar Garcia Colon, Tim Star, Willy-Peter Schaub

[1] http://vsarguidance.codeplex.com/releases/view/88001

Test Clients

> **NOTE** We briefly introduce the Test Hub within Team Web Access in this section, which is the cross-platform, web based interface for doing Test Case Management with Team Foundation Server. Test Hub offers a simplified user experience for most of the capabilities offered in Microsoft Test Manager.
>
> Throughout this guide we primarily use Test Hub, and refer to Microsoft Test Manager for scenarios in which specific capabilities like cloning test plans is yet to light up on Test Hub.

Test Hub within Team Web Access (TWA)

The Test Hub in the Team Web Access (TWA) introduced an alternative to the Microsoft Test Manager (MTM), integrating the management of test plans in the web client. You can create Test Plans, Test Suites, Test Cases and Steps within Team Web Access in one place, as shown in Figure 1.

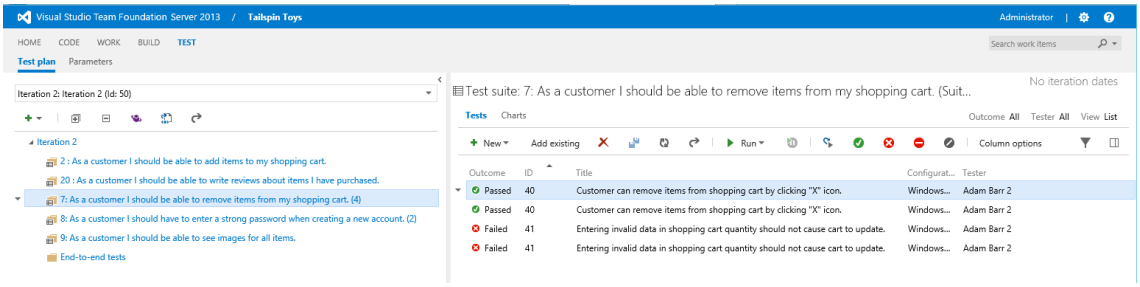


Figure 1 - The Microsoft Team Web Access UI

The test hub allows you to export test plans/suites in an email to be sent, to print, or visualize using lightweight charts.

The following table shows a scenario comparison between MTM and the Test Hub:

Scenario		MTM	Test Hub
Test Planning	Create test plans and test suites	Yes	Yes
	Manage test plan run settings	Yes	No
	Manage configurations	Yes	No
Test Authoring	Author individual tests using test case work item	Yes	Yes
	Author tests with Excel like grid	No	Yes
	Copy paste tests to/from Excel	No	Yes
	Create and manage shared parameters for data driven testing	No	Yes
	Setting up user acceptance testing for multiple users	No	Yes
Test Execution	Run tests on any platform (Windows, Linux, Mac) with Web based Test Runner	No	Yes
	Do rich data collection while performing tests, such as: image action log, video recording, code coverage, etc.	Yes	No
Analyze and Review Tests	Browse test results	Yes	Yes[2]

[2] Coming in TFS 2015.

Scenario		MTM	Test Hub
	Create charts with various pivots like priority, configuration, etc., to track test progress	No	Yes
	Export test plans and test suites for reviewing	No	Yes
	User Acceptance Testing – Assign tests and invite by email	No	Yes

Table 1 – Scenarios comparison between MTM and the Test Hub in TWA

The following table shows a feature comparison between MTM and the Test Hub:

Feature	Action	MTM	Test Hub
Test plans	• Create	Yes	Yes
	• Add requirements	Yes	Yes
	• New suites	Yes	Yes
	• Export	No	Yes
	• Charts	Yes	Yes
	• Charts	Limited	Yes[3]
	• Configurations	Yes	No
	• Properties	Yes	Yes
	• Run settings	Yes	No
Test suite	• Create	Yes	Yes
	• Run	Yes	Yes
	• Run with options	Yes	No
	• Export	No	Yes
	• Run in client		Yes
Test case	• Create	Yes	Yes
	• Author test cases using Excel like Grid	No	Yes
	• Add existing	Yes	Yes
	• View results	Yes	Yes
	• Set state: Active, Passed, Fail, Blocked, N/A	Yes	Yes

Table 2 – Point feature comparison between MTM and the Test Hub in TWA

Figure 2 and Figure 3 highlight the features within the Test Hub to manage test plans, test cases, test case information and the new export and charting capabilities.

[3] Custom charts are available on Test Hub.

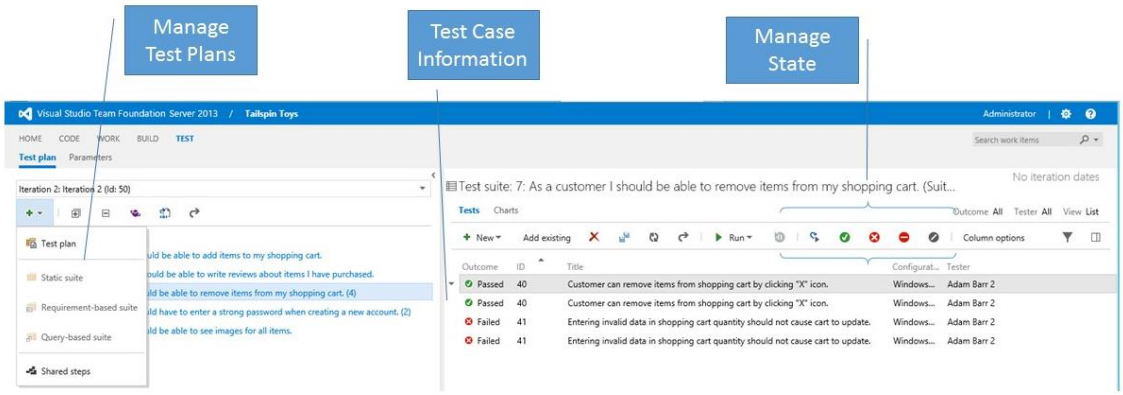


Figure 2 – Team Web Access

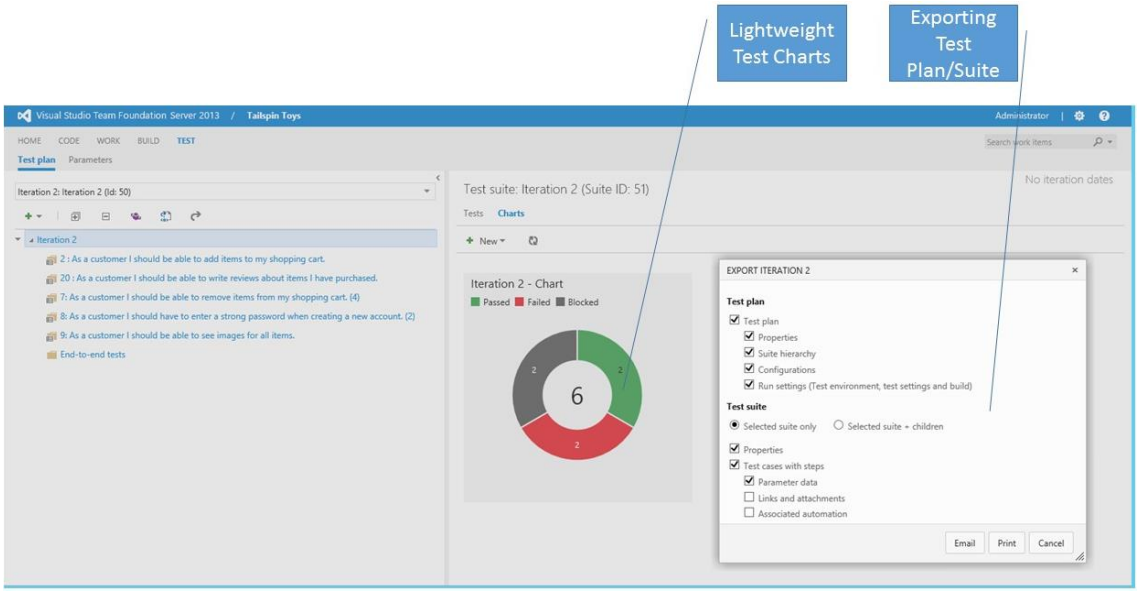


Figure 3 – Export feature

Which client should you use … Microsoft Test Manager or Test Hub in Team Web Access?

Test Hub in Team Web Access (TWA) is much refined and simplified improvement over Microsoft Test Manager (MTM) for Test Case Management with TFS.

For the tester persona, who has to author lots of manual test cases, the Excel-like gird for writing test cases is particularly useful. The lightweight test runner is great to run tests, if you don't do data collection. If you need to capture action logs or video, you can still do test planning and authoring in Test Hub, and switch to MTM for running tests[4].

[4] Using 'Run in Client' feature in Test Hub, launches the Test Runner in MTM.

Test Hub currently lacks create functions performed by Test Lead/Test Manager personas, namely, managing test settings, setting up configurations, or cloning test plans and test suites. 'Further functionality will be released to the Test Hub in upcoming TFS updates, watch the Developer Tool Blogs [5] for details.

Use Table 1 – Scenarios comparison between MTM and the Test Hub in TWA, Table 2 – Point feature comparison between MTM and the Test Hub in TWA or Testing your application using Microsoft Test Manager [6], for an overview of the features and differences.

[5] https://www.visualstudio.com/news/news-overview-vs

[6] https://msdn.microsoft.com/en-us/library/vstudio/jj635157(v=vs.120).aspx

Test Planning and Management

The need for Test Planning Management

Team Foundation Server provides a rich set of features to manage and support project testing activities. The need for test planning and management arises when a project's test team is faced with one or more of the following challenges:

- Provide testing reports for each sprint.
- Maintain a snapshot of the test cases at release.
- Provide testing reports by branch.
- Minimize the test plan and work item management overhead.

To successfully meet these challenges, the software test lead must develop a solution for organizing and managing the project's test artifacts, which are:

- Test plans
- Test suites
- Test cases
- Shared steps
- Shared parameters

The test plan contains vital information, such as

- Configurations to test
- Environments and it's settings for manual and automated testing
- Builds to test

Test Artifacts

The following picture shows how test suites and test cases are displayed within a test plan in Microsoft Test Manager (MTM). The test cases displayed are within the selected test suite.

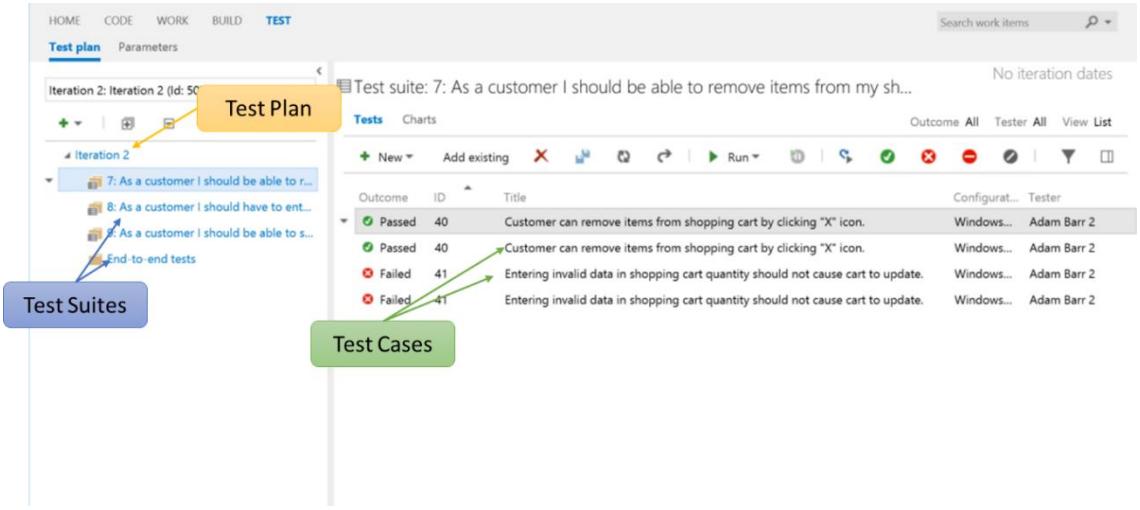


Figure 4 – The TFS Web Access Test Hub

- **Test plans** and **test suites** have been changed to Team Foundation Server work items as of TFS 2013 update 3.
- **Test cases** are work items, but the steps within a test case cannot be edited in Visual Studio. Test steps must be edited in Microsoft Test Manager or other tools specifically designed for that purpose (see references below).
- **Shared steps** are work items and the steps within them must be edited the same way that test case steps are edited. A shared step can be used in multiple test cases.

- **Shared parameters** are also work items and includes values (test data) that can be used in steps and shared steps in a test case. Shared parameters can be used in multiple test cases and it is possible to map columns names to names local to the test case.

NOTE

Unlike other work items such as Features, User Stories, and Tasks, that are typically linked with work item links for managing relationships, relationships between test plans and test suites are not managed by work item links. Though Test Hub (and MTM) have a fit-for-purpose UI to show test suite hierarchy, few scenarios such as the following are blocked without query capability:

- show me list of all suites that belong to a plan,
- show me list of test cases that are present in a suite,
- show me the entire hierarchy of all suites and test cases for a given plan,
- show me the list of all suites a test case belongs to

For most of such scenarios, the approach has been to provide fit-for-purpose UIs - Test Hub (and MTM) have a UI to show test suite hierarchy. For example, with TFS 2013 Update 4, the 'Associated test suites' pane was added that shows all test suites to which a test case belongs. Primary reason for not adding links is that the Test data is distributed across two different stores, the WIT store (Test plans, Test suites and Test cases) and the Test store (test points, test runs, test results etc.). This data could be modified in either of these stores (from the WIT form, Excel add-in, Web access, Test hub, MTM etc.). There is no easy, consistent and reliable way of keeping both these stores in-sync for the changes initiated; there are always some corner cases in which the data would get out-of-sync and be rendered unusable.

The following illustration shows the test case UI form in Microsoft Test Manager. The test steps and shared steps are displayed on the **Steps** tab.

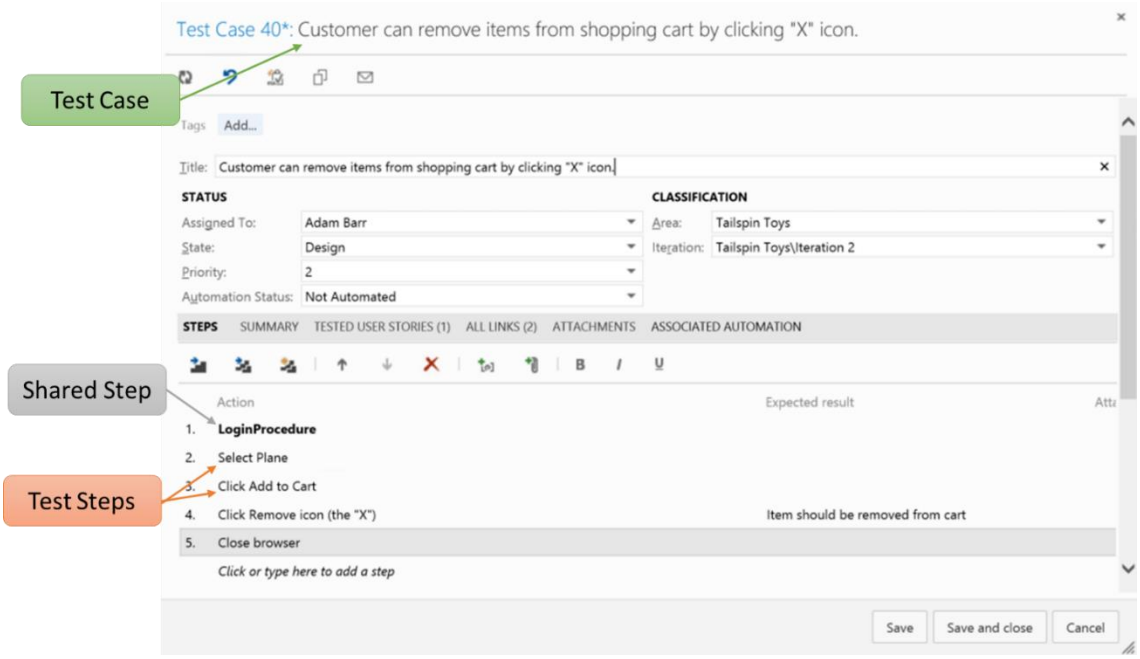
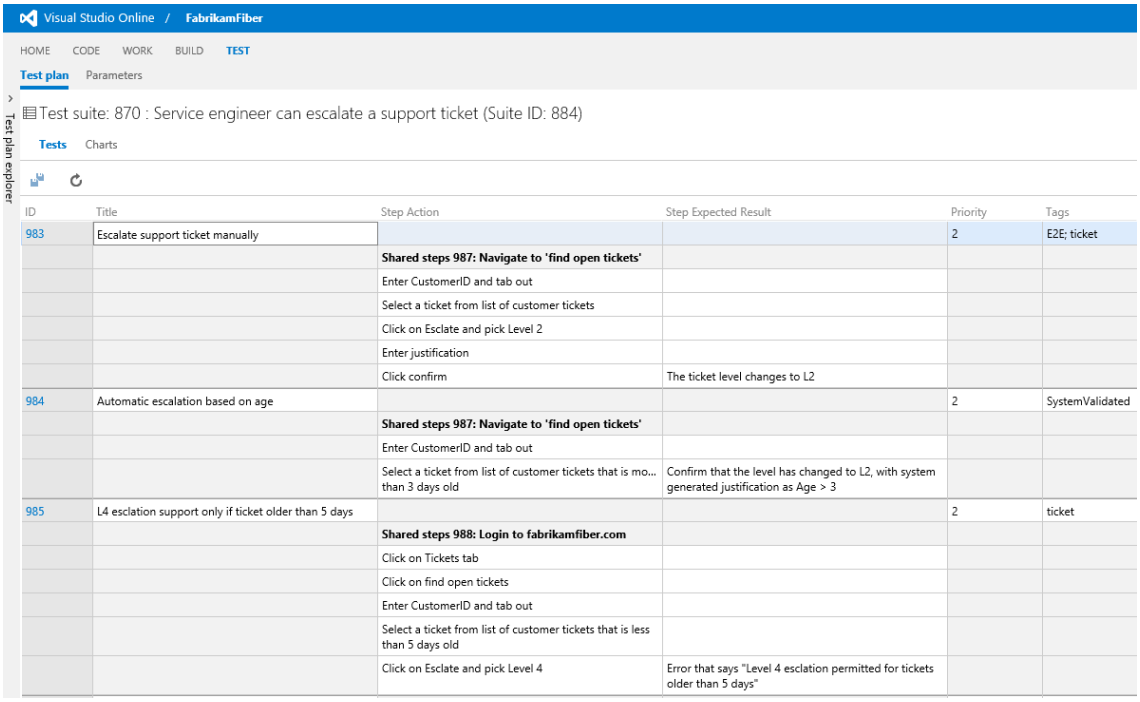


Figure 5 – The TFS Web Access Test Hub Test Case UI

The next illustration shows the Excel-like grid in Test Hub, which simplifies the writing of tests.



ID	Title	Step Action	Step Expected Result	Priority	Tags
983	Escalate support ticket manually			2	E2E; ticket
		Shared steps 987: Navigate to 'find open tickets'			
		Enter CustomerID and tab out			
		Select a ticket from list of customer tickets			
		Click on Esclate and pick Level 2			
		Enter justification			
		Click confirm	The ticket level changes to L2		
984	Automatic escalation based on age			2	SystemValidated
		Shared steps 987: Navigate to 'find open tickets'			
		Enter CustomerID and tab out			
		Select a ticket from list of customer tickets that is mo... than 3 days old	Confirm that the level has changed to L2, with system generated justification as Age > 3		
985	L4 esclation support only if ticket older than 5 days			2	ticket
		Shared steps 988: Login to fabrikamfiber.com			
		Click on Tickets tab			
		Click on find open tickets			
		Enter CustomerID and tab out			
		Select a ticket from list of customer tickets that is less than 5 days old			
		Click on Esclate and pick Level 4	Error that says "Level 4 esclation permitted for tickets older than 5 days"		

Figure 6 – The Test Hub in TWA UI

Test suite types

Static and dynamic test suites

There are three different types of test suites. **Static** suites, **Requirements** based suites and **Query** based suites. While the first type is **static**, the last two are **dynamic**, in the meaning that the content (test cases) contained in the suites is dynamically populated each time you open the tab or refresh the view.

There are several benefits and use cases where dynamic suites are good and provide great value, but there are also some drawbacks from using dynamic suites. In short, dynamic suites are great when building and administering your test plan and executing your tests, but have drawbacks when it comes to documenting/storing the result of the test effort.

WARNING	**Dynamic suites and Baselines** Dynamic suites are dynamic and problematic if your organization requires that you maintain test cases and test results in their exact state upon the release of the software. Consider **cloning** the test plan and its requirements to fully document the state of the requirements and the test plan.

Static suites

A static suite is a placeholder in the test plan and contains a fixed set of test cases and other test suites. Static suites are great when you structure your test plan to enable aggregated reporting and then you need to provide long term documentation.

You use static suites to structure your test plan like this.

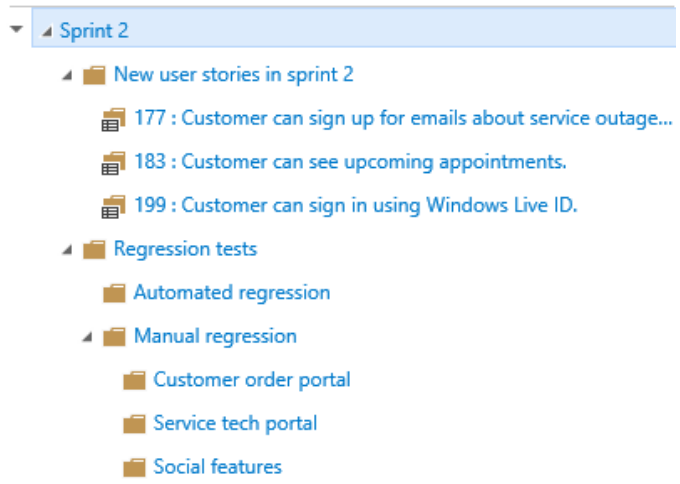


Figure 7 - Static suite example

If you're using static suites and want to have traceability between requirements and test cases, you need to manually link the test case to the requirement by adding a link of type test/Tested between the test case and the corresponding requirement.

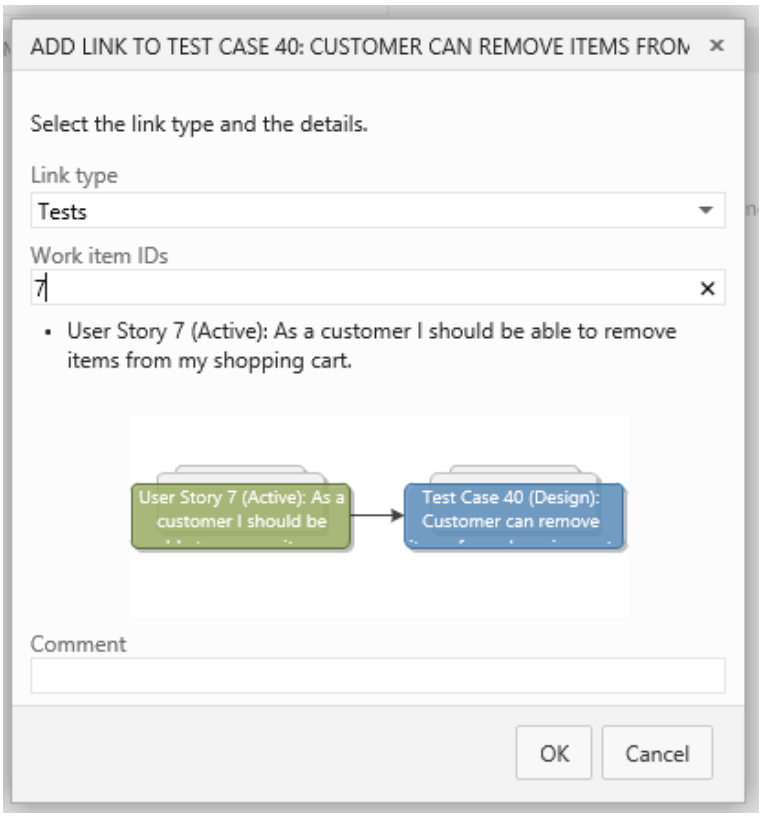


Figure 8 – Manual linking test cases to requirement to achieve traceability between requirement and test cases then not using Requirement Bases Suites.

Requirements based suites

Requirements based suites add requirements (work items that's a part of the requirements category) to the test plan as a suite, along with all test cases linked to the requirement. If you add a new test cases to the requirements based suite (i.e. the requirement), the test cases automatically get linked as Tests/Tested by to the requirement. This gives you traceability between requirement and test cases. Further, when you file bugs they will also get linked both to the test cases and the requirement, providing traceability between bugs and requirements.

The traceability between requirements and test cases and bugs, gives you the possibility to report on the aggregated status on your requirement, covering work, test progress and bug status.

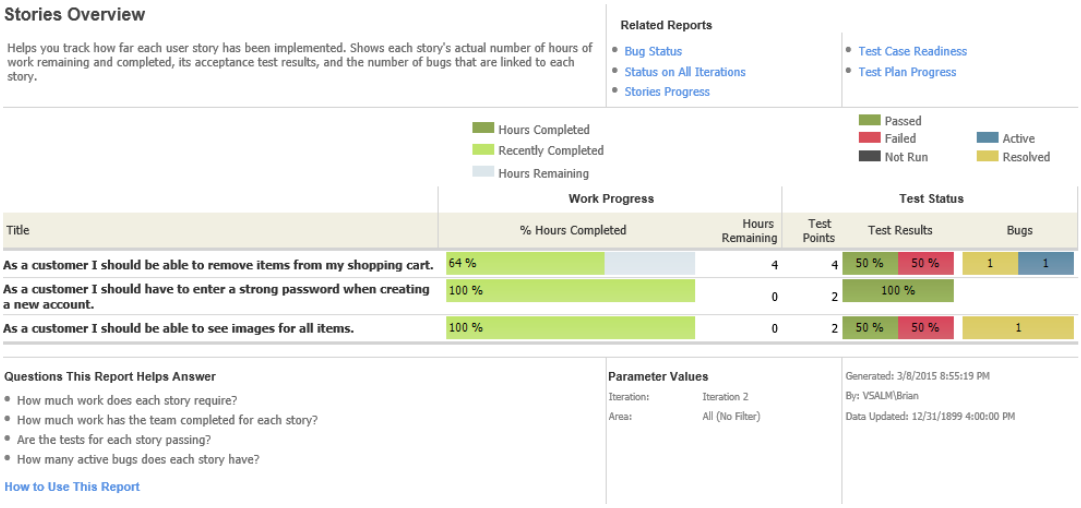


Figure 9 - Aggregated status report example

Requirement Based Suites aren't the only way to accomplish traceability between requirement and test cases, but using Requirement Bases Suites provides it automatically.

Another benefit of Requirements based suites is that it's always showing ALL test cases for the requirement. If a test case gets added to a requirement, either manually by linking them, or by adding the test case in another test plan, the new test case will automatically show in the Requirement based suite. Now in a regression test you might not have the time to do regression testing of all test cases, and instead of removing the test case from the Requirement based suite, the recommended approach is to document the decision not to test by setting test outcome to "not applicable".

WARNING	Removing a test case from a Requirement Based suite will remove the traceability between the Requirement and test case across ALL test plans and suites and create an orphan test case. Instead set the test outcome to "Not Applicable" instead.

Query based suites

Query based suites dynamically fetches test cases that meet the suite's query criteria. This is very powerful for handling a large number of test suites and test cases. However, the query conditions must be setup carefully to ensure expected contents. For example, using iteration path could work well for new test cases in the current sprint. But if that suite is copied to a new plan for the next sprint or release, the iteration path must be updated. A more practical use of a query based suite would be when you want to have a suite that contains all test cases where priority equals one or perhaps you are using tags in TFS to annotate that a test case need only be executed in a specific environment such as Release to Market (RTM), Generally Available (GA), or User Acceptance Testing (UAT).

These are scenarios where query based suites really shine. For more information on how to create query based suites, see Create and Manage Query-Based Test Suites[7] and Plan Manual Tests using Team Web Access[8].

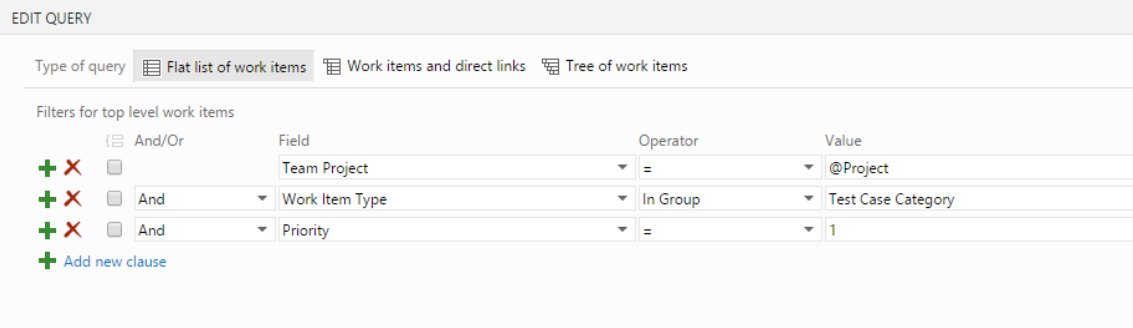


Figure 10 – Edit query example

Test plan

The Test Plan has changed to become a fully-fledged work item in TFS. The fields and workflow can be changed to meet the business process requirements. It is can also be queried like any other work item. The standard test plan contains the following information

- Title
- Area Path
- Iteration Path
- Assigned To
- State
- Start & Finish Date
- Description
- Attachments
- Links
- History
- Tags

In addition to those fields, the following settings is related to the test plan, but not stored as work item information. This information is currently only accessible through MTM.

- Manual Run settings
- Automated Run settings
- Configurations
- Build Settings

Run settings

Manual and Automated run settings controls how much data (like system information, action logs or screen and/or voice recording) MTM collects during test execution, and the environments for manual and automated testing.

Configurations

You might need to test your system in different configurations, for example operating system, product package or browser.

By creating configurations you can select, manage and track witch test cases is assigned to and executed for your different configurations. A configuration is just a name, but you can describe your test configurations by defining different configuration

[7] http://aka.ms/Yuvani

[8] http://aka.ms/i6rciu

variables and link different variable values to a test configuration. (For more info, please refer to Test configurations: specifying test platforms [9])

Build settings

If your testing software built with Team Foundation Build, you can select the Build and a filter for your test plan. By doing so MTM can help you understand the changes between your current build and any other build, by presenting a list of affected work items and even individual code check-ins.

[9] https://msdn.microsoft.com/en-us/library/dd286643.aspx

Test Planning and Management Objectives

The objectives of test planning and management are to determine the minimal update of your test artifacts to ensure that test tracking and reporting requirements are met. The update typically takes place when:

- Going to the next sprint.
- Moving code from one branch to another.
- Going to the next release.

A key consideration when you plan for test release management is the classification of test artifacts. Classification is determined fundamentally by the Team Project that contains the test artifacts. Within a Team Project, there are fields that uniquely differentiate items from other items of the same type for tracking and reporting purposes. As with work items in general, typical reasons for classification are the need to differentiate between:

- Sprints
- Phases
- Releases
- Sub-teams
- Components
- Baselines

The following table summarizes the classification aspects of test plans, suites and test cases with respect to test planning and management.

Type	Classification / Related Fields	Comments
Test Plan	• Title • Area Path • Iteration Path • Assigned To • State • Start & Finish Date • Description • Attachments • Links • History • Tags	The Test Plan has changed to become a fully-fledged work item in TFS. The fields and workflow can be changed to meet the business process requirements. It is can also be queried like any other work item
Run Settings	• Manual Run settings • Automated Run settings • Configurations • Build Settings	These settings are related to the test plan and are not stored as work item information.
Test Suite	• Title • Area Path • Iteration Path • Assigned To • State • Test Suite Type (read only) • Description • Attachments • Links • History • Tags	The Test Suite has changed to become a fully-fledged work item in TFS. The fields and workflow can be changed to meet the business process requirements. It is can also be queried like any other work item
Test Case	• Title • Assigned To • State • Reason	The test case is a work item and can be queried like other work item types. Test steps cannot be edited from Visual Studio, only though Test Manager and through the test hub in TWA.

Type	Classification / Related Fields	Comments
	• Priority • Automation Status • Area Path • Iteration Path	
Shared Step	• Title • Assigned To • State • Reason • Priority • Area Path • Iteration Path	The shared step is a work item and can be queried like other work item types. Steps in the Shared Steps work item cannot be edited from Visual Studio.
Shared Parameters	• Title • Assigned To • State • Area Path • Iteration Path	The shared steps (set) is a work item and can be queried like other work item types. Columns and values in the Shared Parameters work item cannot be edited from Visual Studio or Microsoft Test Manager. They can only be edited from TWA, but they can be used and mapped in test cases using MTM.

Table 3 - Status and classification aspects of test plans, suites and test cases

Alternatives for Work Item Classification

Work item classification for test planning and management is dependent on how work item classification is managed for the team project as a whole. The two primary classification fields within a team project are:

- Iteration Path
- Area Path

Both fields support an extensible tree hierarchy, which provides a very powerful classification mechanism. The Team Project name is the tree root. Any number of nodes and sub-nodes can be created and managed.

Iteration Path

Use the iteration path to classify work items for a point in time. The field can be used to classify work by one or more of the following:

- Release
- Phase
- Sprint

Security on iteration path is limited to permissions for creating, editing, organizing and viewing permissions on tree nodes. It is possible to have some combination of releases, phases and sprints in the iteration path. However, the iteration path will have multiple meanings. This will be addressed in more detail in the section below on "Using Iteration and Area Path: Two Models."

Area Path

Use the area path to classify work items for reporting and isolation. The field can be used to classify work by:

- Sub-team
- Product component
- Product baseline

Work items on a different area path can be targeted, included or excluded from reports. For example, work items for the prior release can be moved to an area path that identifies the work item release baseline. This can provide a

permanent, static record of the work. Reports can target that area path at any future point to support a project audit or do historical analysis.

Whether for baselines, sub-teams or components, the area path is ideal for isolating work items for reporting and for controlling access permissions to work items on a specific area path. In addition to the permissions available to iteration path, area path also has the following permissions:

- Edit work items in this node
- Manage test plans
- View work items in this node

Using Iteration and Area Path: Two Models

As noted above, how work items are classified will determine how you update your test artifacts when you proceed to the next sprint, the next release or when you branch code. For example, take two fundamentally different ways to do work item classification, the **Declarative Model** vs. the **Latest Model**.

The "Declarative Model"

In the Declarative Model, active work is assigned an explicit version, starting at the beginning of the project. For example, if a project team is working on sprint 1 of the 1.0 release, the iteration path would be ***[iteration root]\Release 1\Sprint01***. When the 1.0 version is released and work on the next version begins, the iteration path for the work will be ***[iteration root]\Release 2\Sprint01***.

The following diagrams show an example of how iteration path and area path nodes could be set up to support the Declarative Model of work item classification.

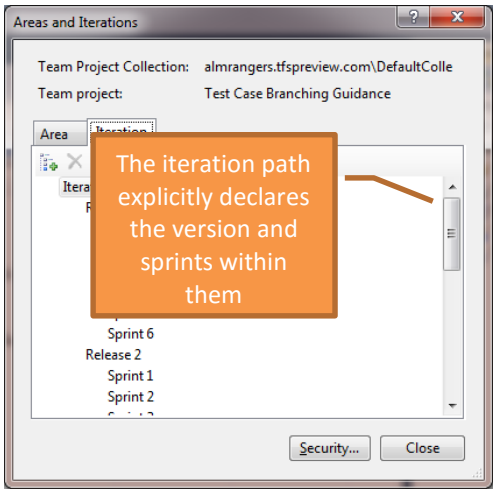


Figure 11 – Example of an Iteration Path hierarchy when using the Declarative Model

Area path might not be needed at all when you use the declarative model. This is because iteration path is declaring both the version (that is, the release) and the time period (that is, the sprint). This leaves area path to be used for any required reporting distinctions. For example, using area path to split out work by sub-team enables reporting for the entire team or by sub-team.

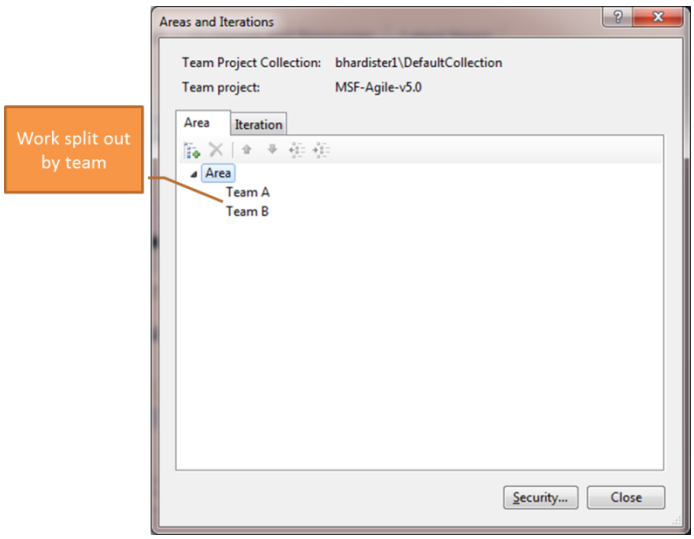


Figure 12 – Example of an Area Path hierarchy when using the Declarative Model

The "Latest Model"

In the Latest Model, the iteration path is not used to designate the release version, only the sprint. Active work is always on the root area path. For example, if a project team is working on sprint 1 of the 1.0 release, the iteration path would be ***[iteration root]\Sprint01*** and the area path would be ***[area root]***. When the 1.0 version is released after sprint 5, the area path for the completed work is changed to ***[area root]\1.0***. Work on the next version begins in iteration path ***[iteration root]\Sprint06*** and area path remains the same as before, ***[area root]***.

The following diagrams show an example of how iteration path and area path nodes could be set up to support the Latest Model of work item classification.

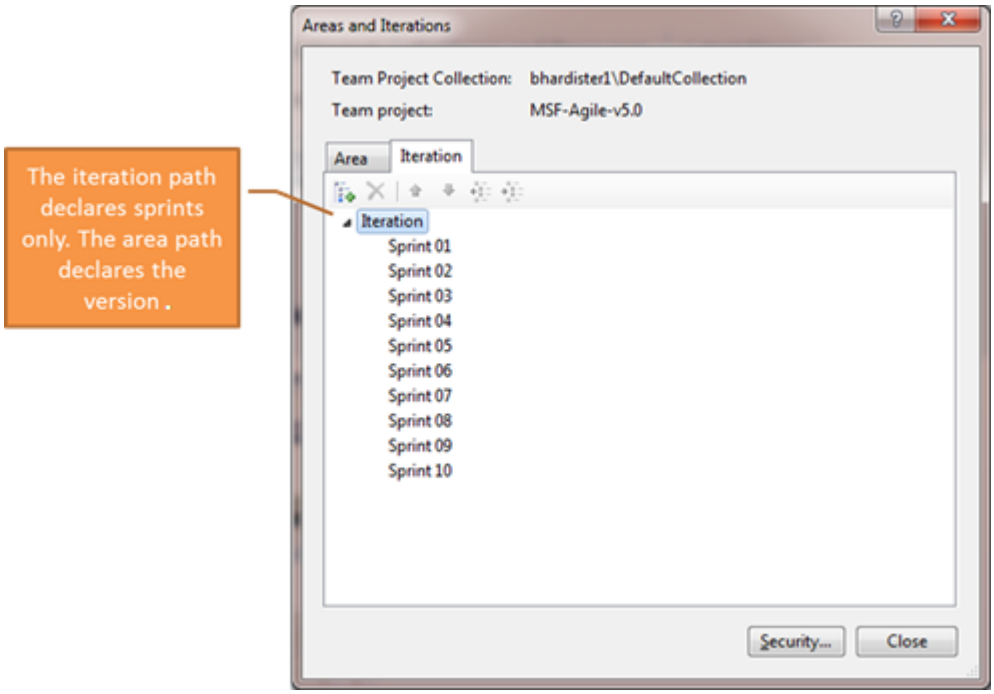


Figure 13 – Example of an Iteration Path hierarchy when using the Latest Model

Area path is required when using the Latest Model so that the version of the software can be identified. The Latest Model uses a double key to uniquely identify any work item:

- Iteration path – time period
- Area path – version

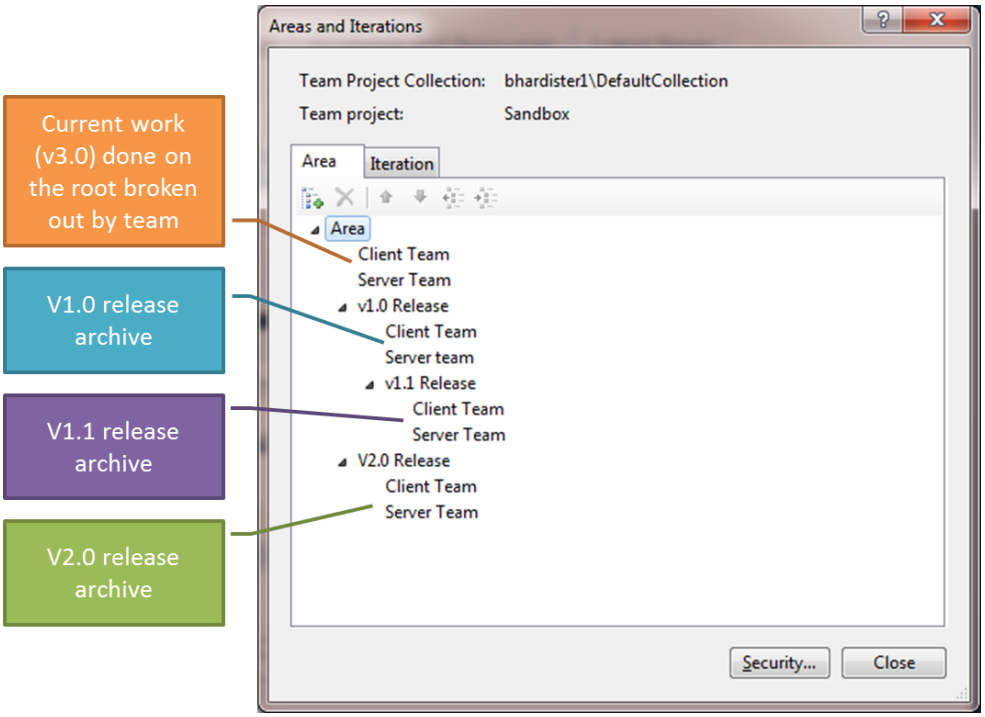


Figure 14 – Example of an Area Path hierarchy when using the Latest Model

Comparison of the Declarative Model vs. Latest Model

The pros and cons of the two models are as follows:

	Declarative Model	Latest Model
Pros	• Simple	• Uses the same area and iteration path for active work across releases: developers working on the latest code always work on the root area path. • Flexible and easy to maintain because assignment of work to an archive , baseline) occurs after the release is completed when all project changes and decisions have already been made • Uses explicit assignment to the archive area path, which requires active review of work items prior to committing them to the baseline. • Flexible and easy to maintain because area path and iteration path each have only a single meaning, so they can be extended and changed without coming into conflict with other meanings as the Declarative Model does. • Work item security is available to specific area path nodes.
Cons	• Less flexible because iteration path fields means both the version and the time	• More difficult to understand

Declarative Model	Latest Model
period (release number and sprint number). • More work to maintain. Changes in the project organization, structure are more likely to require changes to iteration path that then must be updated to all work items. • After a release work items are typically "left behind" without any change to iteration or area path, which provides no explicit indication of a baseline gate review and approval.	

Table 4 - Comparison of the Declarative Model vs. Latest Model

In conclusion, organizations that must support baseline auditing and that must comply with more rigorous configuration management rules should consider using the Latest Model. The advantages of this model are its explicit review and assignment of work to baselines and the security features that are available to the area path.

NOTE	The approach used for test planning and management will differ depending on the work item classification. The following sections will cover the three test release management scenarios addressed by this document using the Declarative Model for work item classification. Adapt the guidance in these sections to develop your own test release management plan.

Defining the Test Mix

When planning a test effort there is often the need to both verify new or changed functionality, as well as verifying that the complete system behaves as expected. There is different techniques and methods for achieving both goals.

For verifying new functionality, designing and creating test cases provides a big overhead that's takes time away from testers to execute tests. Exploratory testing lets testers use their skills to simultaneously learn, design and execute tests to be more efficient.

To verify the existing functionality in a structured way, manual testing using test case is a common approach. But only doing manual regression testing of existing functionality will soon be very costly and time consuming. Test automation can efficiently lower the cost and time needed to verify that the systems behavior hasn't changed.

A key success factor in test planning and managing is to have the right mix of test techniques and methods to provide long term success.

Exploratory Testing

Exploratory testing lets testers spend more time testing and less time designing and documenting their tests. The problem with this has traditionally been, if I find a defect, how can I document it or make it reproducible.

With Exploratory testing in MTM, you get excellent support for performing and documenting your exploratory testing. MTM keeps track of all activates and automatically provides an image log then filing a bug.

MTM can also export part of your exploratory session as a traditional test case with test steps, this is often used as a quick and efficient way of creating a draft for a manual test case

Manual Testing

Manual testing using test cases can be done in a very light exploratory way, as well in an extremely precise and documented manner with test data, shared steps and recorded for fast forward execution.

Regression testing

As regression testing a complete system often takes much effort and time, especially manual regression testing. It's a good idea to categories and prioritize your regression test cases. By doing so you have a distinct way to define, select and communicating the test effort spent on regression testing.

Manual vs. Automated Testing

One of the challenges many teams face is the decision on which types of tests to automate and which to keep as manual processes. With the right test mix you can let test cases mature and make small investments in each step to minimize the risk and investment with instant return of investment in each step.

Other types of testing

There is other kinds of test that's normally performed as a part of the total quality assurance process.

Type	Description	Advantages	Disadvantages
Unit Tests	Written by the developer concurrent with writing the code, these tests are focused on the smallest element of executable code (typically a method or property in Object Oriented languages).	With the fine granularity, they provide pinpoint identification of the code triggering the failure. The concurrent development nature provides immediate validation with maximum capturing of the developers intentions.	Being written by the same person, if there is any misunderstanding of the "specification", that is likely to be reflected in both the program code and the test code.
Component / Integration Tests	These tests are utilized to validate that the individual elements work properly in conjunction with other elements, without requiring an entire system to be in place.	Provides increasing strategic value in validating functionality	Becomes increasingly difficult to pinpoint exact cause of failure, unless prior tests have validated functionality at a more granular level [at which point failure are likely at the integration point itself].
Acceptance Tests	In the most general sense, these tests verify that the work product will be acceptable to the people who will be consuming it. The two most common types of acceptance tests are Story Acceptance Tests (typically defined in the "Definition of Done" for the User Story) and User Acceptance Testing.	Since the tests are defined by the person who will be accepting the work, these test provide a closed feedback loop that mitigates communication errors.	Specifically referring to UAT, there is often a large time gap between the development of the code and the execution of the tests. This is likely to increase the remediation costs for any defects found compared to tests which are executed in a more timely fashion.

Table 5 – Types of testing

Regression Testing

A strategy for managing regression testing should be developed as a part of the initial test management plan. This is because the way in which test suites are setup will set the direction for how regression testing takes place. For example, when cloning or reusing test cases for regression the suite type remains the same. Gaining a good understanding of suite types, suite cloning, suite reuse and regression testing will make the importance of having a good plan for managing regression tests before the project starts more apparent.

The organization of test suites greatly impacts how well a test plan facilitates test execution, reporting, and plan maintenance. Factors to consider include the:

- Type of test suite to use
- Depth and granularity of test suites
- Hierarchical structure of test suites

Test suites contain test cases or other test suites. As described above, there are three types of test suites available: static, requirements based and query based. Additional information on suite types is available in the guidance section Organizing Test Cases Using Test Suites [10]

Using Suites for Regression

Regression testing means re-running test cases from existing test suites to build confidence that software changes have no unintended side effects. Here are some important principles to keep in mind:

- Regression testing assumes no change to the requirements
- Test results will report against the requirement when using requirement-based suites
- Test cases must be manually added or removed from static suites
- Test cases will automatically be added or removed from query-based suites according to whether query criteria are met

Alternatives for Organizing Regression Test Suites

Approach	Description	Pros/Cons
Single Plan	Use a single test plan that has regression and non-regression suites	One reporting result set. Managing a single comprehensive plan can be more complex.
Separate Regression Plan	Keep all regression suites in a separate regression test plan. New test cases belong to suites in a different test plan.	It's easy to manage the separate plans according to their type. However, there is no single results rollup to one plan.

Table 6 - Alternatives for organizing regression Test Suites

Creating Regression Test Suites

Our recommendation is never to create regression test suites from scratch. Always reuse or clone them:

- Reusing a test suite creates a new suite, but reuses the existing test cases.
- Cloning a plan or a suite creates a new suite and duplicates the test cases, shared steps, and optionally, the requirements linked to them

Additional information on reusing or duplicating test suites and test cases is available in the guidance section Copying and Cloning Test Suites and Test Cases [11].

[10] http://aka.ms/L99mgu

[11] http://aka.ms/lnlugl

Consider the following before creating regression test suites:

Requirement	Approach
Baseline required	Do not reuse test cases since a snapshot of all work items for a release must be retained and unchanged. Rather, use the Test Plan Clone feature or the TCM suites /clone command line to clone specific suites. The command line provides a **/clonerequirements** option that will duplicate the requirement work items in the cloned plan. Otherwise, if this option is not used, requirement based suites will be converted to static suites.
No baseline required	Reuse test cases in the new regression suites. Results are by suite and by plan. When reusing test cases the suite type remains the same from old to the new suite.

Table 7 - Regression Test Suites considerations

NOTE

When reusing test cases consider the resulting query type in the new suite

Static suites operate exactly as before. A requirement based suite will use the same requirement work item used as the old suite. Query based suites are exactly as before, but the criteria may need to be changed. For example, the iteration path in the old suite could be incorrect for the new suite.

Moving from Sprint to Sprint

In most software development projects that apply an agile approach, the development is done in multiple sprints or iterations. Each sprint is usually targeted to deliver an increment of software functionality that includes specified requirements (user stories or product backlog items) and fixed bugs, as well as both manual and automated regression testing. Sprint or Iteration testing should be a mix of exploratory, structured manual and automated testing.

This section of the guide explains test planning principles and best practices for moving to the next sprint. To accomplish this, a sprint-to-sprint scenario from an example project will be used. The profile of the example project is as follows:

- Our persona, Christine, is the test lead on the project.
- The project uses the Scrum template.
- The project is using declarative work item classification where the iteration path is ...\Release 1\Sprint 1.
- Area path is not being used.
- Christine is using one test plan per iteration.

The Sprint-to-Sprint Scenario

Our example project has just completed sprint 1. The software is now ready for release and the project is moving on to sprint 2.

Christine, the project's test lead, must now create a test plan for sprint 2 that includes:

- The regression testing of completed work from sprint 1
- New test cases for the planned implementation work in sprint 2

In addition, Christine must account for unfinished and deferred testing. First, there are test cases where more testing is needed. Second, there are test cases allocated to sprint 1 where the user stories being tested have been deferred to sprint 2.

While the project requires that testing results and reporting for each sprint be maintained, there's no requirement for maintaining a baseline of the test cases for each sprint. However, in the next section of the guide, "Release to Release," this requirement will be addressed.

NOTE	A "**baseline**" of a test case is the exact state or snapshot of a test case work item at a particular point in time. Team Foundation Server does provide work item history, but does not support work item versioning. Therefore, to maintain a testing baseline is to make no changes to the baseline test plan and its test cases after the baseline is set.

Christine needs to:

- Complete this work as quickly and efficiently as possible.
- Provide sprint 1 reports that are accurate, complete and ready for the upcoming sprint 1 retrospective.
- Ensure that the test plan for sprint 2 is set up to provide the correct test reporting.
- Be sure that the reporting for sprint 1 will not be affected by sprint 2 activities.

Recommended Practices

In the example project, the first step is to determine how user stories assigned to sprint 1 will be verified and which test cases from sprint 1 will be run in regression during sprint 2. The following questions should be considered:

- What are the new requirements to be verified in sprint 2?
- How the new requirements should be tested?
- Is sprint 1 testing on track?

- Are there test cases we need to invest in to lower the cost of future regression testing?
- How is test progress at the end of the sprint to be reported?
- How should regression test cases within the test plan be organized?

The following sections cover typical scenarios related to managing test artifacts across sprints. They include actions and reports that you might find useful when building your own test release management plan.

> **NOTE** The recommended practices are set at the conceptual level. They are intended to convey the nature of the solution, but not the exact steps for performing the work. Please refer walkthroughs, page 46, and demo videos for procedures and an example of executing sprint-to-sprint test case management.

Close Sprint 1

At the end of a sprint 1, use the Test Hub or MTM to update and close out the sprint 1 test plan.

Action	Guidance
Review Work Status	Verify: • All test cases are in the correct iteration and area path • Test plan dates, iteration and area path is correct Identify: • Test cases suitable for regression testing • Verified test cases not to be regressed in sprint 2. • Requirements awaiting approval where work, testing, or bugs, has been deferred to sprint 2..

Table 8 - Update and close out the sprint 1 test plan

Sprint 2 Planning

During sprint 2 planning, you need to create a test plan for sprint 2 and then add suites and test cases to the plan so that:

- New requirements for sprint 2 are tested
- Selected regression test cases can be run against user stories completed during earlier sprints.
- Automated test cases can be executed.

Use the Test Hub or MTM to create and set up the sprint 2 test plan.

Action	Guidance
Create the Sprint 2 Test Plan	• Include the sprint number in the plan name.[12] • Set the sprint test plan testing state to **In planning**. • Assign the start and end dates to the test plan so they match the sprint start and end dates. • Set the test plan iteration path to \project\Release1\Sprint2.
Assign new requirements	• Create a Static suite named **Sprint 2 requirements.** • Add the new requirements to the test plan using Requirements Based Suites.
Assign Test Cases for regression	• Create a Static suite named **Regression Testing**. • Create a Use the **Copy suite from another test plan** feature to copy all the suites from the sprint 1 test plan into the sprint 2 test plan (except for those suites where you don't plan to execute any of the test cases in sprint 2).

[12] Starting with 2012 Visual Studio Update #2, you can simply use the "Test Plan Clone" capability to jump-start your test plan creation process. The clone operation applies the values of configurations, test settings and other properties present on the source test plan to the destination test plan.

Action	Guidance
	• Mark any test cases from the suites that you do not plan to execute in sprint 2[13] as Non Applicable • Change the iteration path of the remaining test cases in the sprint 2 test plan from \project\Release1\Sprint1 to \project\Release1\Sprint2.
Assign Test Cases for automated regression	• Create a Static suite named **Automated Regression Testing**. • Use the **Copy suite from another test plan** feature to copy all the suites from the sprint 1 test plan into the sprint 2 test plan (except for those suites where you don't plan to execute any of the test cases in sprint 2). • Mark any test cases from the suites that you do not plan to execute in sprint 2 as Non Applicable • Change the iteration path of the remaining test cases in the sprint 2 test plan from \project\Release1\Sprint1 to \project\Release1\Sprint2.

Table 9 - Create and set up the sprint 2 test plan

NOTE

Currently shared parameter cannot be cloned so to create a copy of an existing parameter set do the following:

- Create a new shared parameter set
- Add the parameter names (column headers) matching the existing data. Optionally you can change the names.
- Copy the data in the grid of the original shared parameter set.
- Paste the data into the grid of the new shared parameter set.

For more advanced copy and editing, paste the data into Excel before pasting into the new shared parameter set.

Sprint 2 Execution

For new requirements you can start doing exploratory testing, as its get you fast into testing and learning the new requirement and implementation with minimal overhead. Once you get an understanding of the requirement and implementation, you can use exploratory testing to test and create the test cases you plan to re-execute for formal verification of the requirement.

You can lower the future costs of testing regression cases by detailing them, providing better steps and concise expected results, specifying test data, or recording the test execution and fully automating the case..

NOTE

Remaining test work reporting issues

Remaining test work is any test cases in a current sprint test plan (sprint 1 in the example above) in a state of "ready" that are not-run or failed.

Remaining work will continue to display in the SQL Server Reporting Services (SSRS) Stories Overview report and similar reports unless it is removed from old test plans. This may be acceptable. If you wish to avoid this:

- Move (rather than copy) not-run and failed test cases to the new test plan.
- Change the iteration path of test case and the user story it tests to the next sprint.

[13] With Visual Studio Update #2, you can simply mark these test cases with "Not Applicable" outcome instead of explicitly removing them from the suite (either static or requirement suite)

Working with multiple branches

This scenario describes test planning and management when working with multiple branches within the same release. The Visual Studio ALM Rangers publish best practices about how to handling and isolating code changes in different situations. Those recommendations are published in the Version Control (ex Branching and Merging) Guide [14]. The branching guide covers different branch plans suitable for different situations.

Figure 15 - Basic Branch Plan from the Visual Studio ALM Rangers Branching Guide

This guide will address test planning and management for the basic branch plan, which involves branching for development and branching for release. The example project will be used to illustrate test planning and management when working with multiple branches. The profile of the example project is as follows:

Our persona, Christine, is the test lead on the project.

- The project uses the Microsoft Visual Studio SCRUM process template.
- The project is using declarative work item classification. The iteration path is …\Release 1\Sprint 1.
- Area path is not being used.
- The same version of the software is:
    - Developed and integration tested on the DEV branch.
    - System tested on the MAIN branch.
    - User acceptance tested and released on a RELEASE branch.
    - Separate testing reports are required for the DEV, MAIN and RELEASE branches.

While the Sprint to Sprint and Release to Release scenarios take place at the end of a sprint, code branching can happen at any time during a project. When working with multiple branches, the project test lead, Christine needs to:

- Update the test artifacts as quickly and efficiently as possible.
- Provide reports that are accurate and complete for each branch.

This scenario provides guidance on how to manage test artifacts when code is branched and merged.

The Working with Multiple Branches Scenario

In our example project, feature implementation and integration testing is being conducted on the DEV branch. Christine setup a test plan to do integration testing. Several of the features in the DEV branch are ready for system testing and over the next few days will be merged to the MAIN branch for system testing.

Christine should consider the following:

- Which, if any, test cases being run on the DEV branch should also be run on the MAIN branch
- The creation of new test cases to be used for system testing on the MAIN branch

[14] http://vsarbranchingguide.codeplex.com/

Christine must also prepare for user acceptance testing when the RELEASE branch is created. The user acceptance tests are specified by the product manager. Christine will need the following types of test cases for testing against the RELEASE branch:

- A new set of user acceptance test cases.
- A small set of smoke-test cases reused from testing run against the MAIN branch.
- Test cases for any bugs found and fixed during user acceptance testing.

In summary:

- Feature implementation and integration testing is being conducted on the DEV branch.
- System testing is being conducted on the MAIN branch.
- User Acceptance Testing (UAT) is being conducted on the RELEASE branch.

Recommended Practices

When working with multiple branches Christine, the test lead will need to:

- Set up a test plan for each branch as the project requires separate reports for the types of testing run against each branch.
- Share test cases across plans as needed.
- Create new test cases that are unique to a particular plan.

The following sections cover typical scenarios related to managing test artifacts when you work with multiple branches. They include actions and reports that you might find useful when building your own test release management plan.

NOTE The recommended practices are set at the conceptual level. They are intended to convey the nature of the solution, but not the exact steps for performing the work.

Setup a Test Plan for Integration Testing

Use a separate test plan for tests run against the DEV branch in the current sprint. Create this test plan as part of sprint planning as described in the **Sprint to Sprint** section above.

Setup a Test Plan for System Testing

Set up a separate test plan to run tests against the MAIN branch. Typically, this plan will reuse the functional tests run against the features in the DEV branch. This test plan should contain tests that are unique to the integration of features into the system as a whole. Regression tests appropriate to system testing can be included in a regression suite of this plan or in a separate overall regression test plan. See the section above on Regression Testing for options on creating regression test suites.

NOTE Do not duplicate test cases within the same version of the software. This will make reporting and baselining more difficult.

Except for when making a baseline, it is fine to reuse the same test case in multiple test plans. The results of test case execution will be broken out to the plan under which it was run. Having a single test case for the same test steps ensures that conflicting results do not occur and that test step definitions do not diverge and become invalid over time. The MTM shallow copy [15] feature supports this at the suite level.

[15] https://msdn.microsoft.com/en-us/library/vstudio/hh543843%28v=vs.110%29.aspx

Use MTM to create a new test plan to support system testing on the MAIN branch.

Action	Guidance
Create the Test Plan	• Include the branch name and sprint number in the plan name [16]. • Set the test plan testing state to **In planning**. • Assign the start and end dates to the test plan so they match the sprint start and end dates. • Set the test plan iteration path to the current sprint. • Set up distinct test suites for unique vs. shared test cases
Assign Shared Test Cases	• Add required test cases where those test cases are already being used in other test plans of the current sprint.

Table 10 - Create a new test plan to support system testing on the MAIN branch

NOTE	To reuse a single test case, simply add existing test cases that are already used in one plan to another plan. To reuse a whole suite, use the **Copy suite from another plan (shallow copy) feature** shown in the Sprint-to-Sprint section above

Action	Guidance
Assign New Test Cases	• Add test cases from the backlogs that are planned for testing on the MAIN branch during the sprint. • Set the iteration path of these to the current sprint.
Verify Status, State and Links	• Verify that the test plan, suites, test cases, shared steps, and shared parameters are in the correct status and state, and that they are linked to the correct user stories.
Verify Iteration Path	• Verify that all the test cases, shared steps, and shared parameters in the test plan are set to the current print.

Table 11 - To reuse a single test case, simply add existing test cases

When you are ready to start testing builds from the MAIN branch, set the MAIN branch test plan testing state to **In progress**.

[16] As noted in the section above, with 2012 Visual Studio Update #2, the Test Plan Clone capability can be used to jump start this.

Setup a Test Plan for User Acceptance Testing

When the RELEASE branch is created:

- Create a new test plan for the branch.
- Set up test suites in the plan to distinguish between shared test cases, unique planned test cases and bug test cases.
- Create new test cases for testing unique to that branch.
- Reuse test cases from other plans where the same testing is required on this branch.

> NOTE Again, do not duplicate test cases within the same version of the software. This will make reporting and baselining more difficult.

As of TFS 2013, MTM now allows multiple testers to be assigned to a test case similarly to how a test case can be assigned multiple configurations.

> NOTE Consider creating a new Team to host UAT testers to ensure they only have access to test cases relevant to UAT. Refer to the User Acceptance Testing walkthroughs, page 59, for details.

Use MTM to create a new test plan to support user acceptance testing on the RELEASE branch.

Action	Guidance
Create the Test Plan	• Include the branch name and sprint number in the plan name. • Set the test plan testing state to **In planning**. • Assign the start and end dates to the test plan so they match the sprint start and end dates. • Set the test plan iteration path to the current sprint. • Set up distinct test suites for unique vs. shared test cases.
Assign Shared Test Cases	• Assign existing test cases where those test cases are already being used in other test plans of the current sprint.
Assign New Test Cases	• Add test cases from the backlogs that are planned for testing on the RELEASE branch. • Set the iteration path of these to the current sprint.
Verify Status, State and Links	• Verify that the test plan, suites, test cases, shared steps, and shared parameters are in the correct status and state, and that they are linked to the correct user stories.
Verify Iteration Path	• Verify that all the test cases, shared steps, and shared parameters in the test plan for sprint 1 have the correct iteration path.

Table 12 - create a new test plan to support user acceptance testing on the RELEASE branch

The following illustration shows the new Web UI, with a useful feature to assign testers and invite by email:

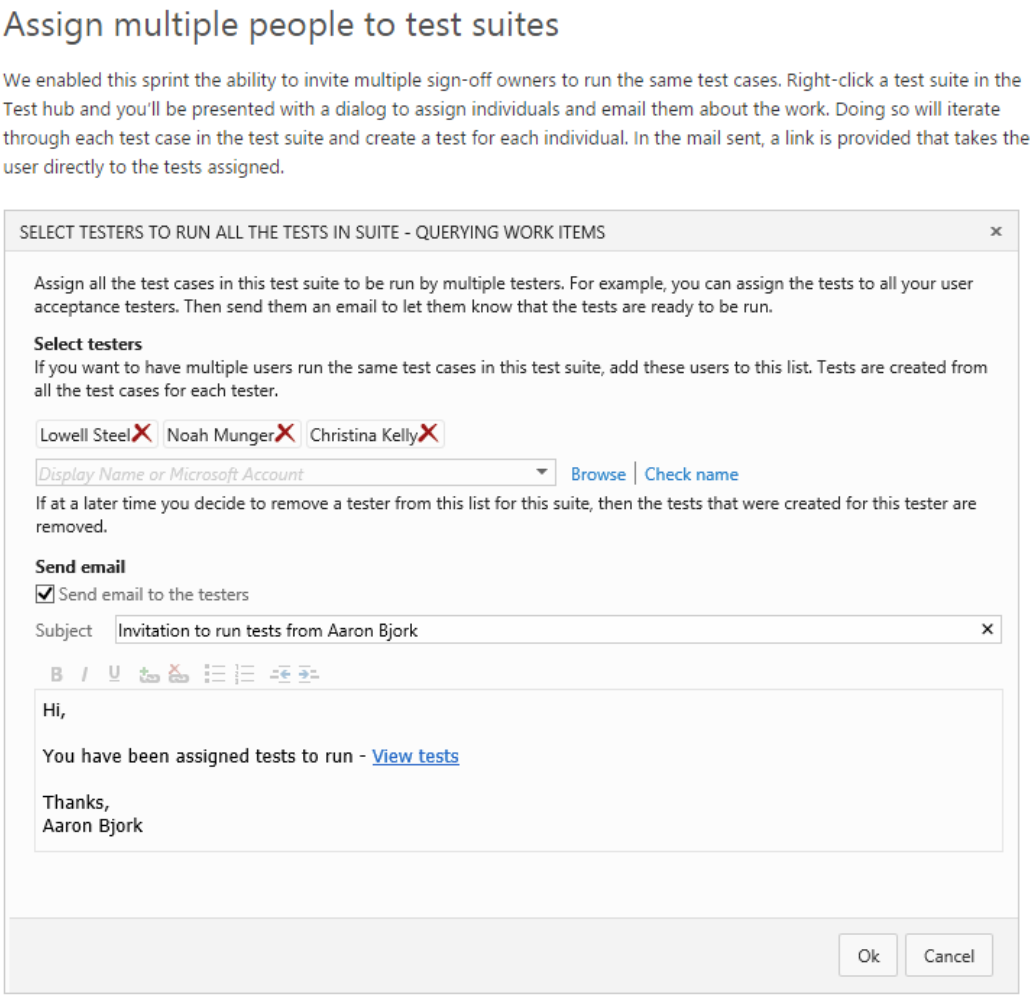


Table 13 - create a new test plan to support user acceptance testing on the RELEASE branch

NOTE	Exploratory tests can also be used as part of User Acceptance Testing.

When the project is ready to start testing builds from the RELEASE branch, set the RELEASE branch test plan testing state to **In progress**.

Moving from Release to Release

In this scenario, **release** means that the version of the software under development is complete. No more development, merging, building, testing or software engineering will take place against this version of the software. The product has **shipped** and the next sprint is the first sprint of the next version of the software.

The purpose of this section is to explain test release management principles and best practices for going to the next release. To accomplish this, a release-to-release scenario from an example project will be used. The profile of the example project is as follows:

Our persona, Christine, is the test lead on the project.

- The project uses the Microsoft Visual Studio SCRUM process template.
- The project is using declarative work item classification where the iteration path is …\Release 1\Sprint 2.
- Area path is not being used.
- Christine is using one test plan per iteration.
- The software is developed, tested and released on the same branch (that is no more than one source control branch is ever used).

In our example project, the product is being released at the end of sprint 2.

Going to the next release has some similarity to going to the next sprint. For example, the following test release management requirements for going to the next sprint are also applicable when going to the next release. The project test lead, Christine, needs to:

- Complete this work as quickly and efficiently as possible.
- Provide release 1 sprint 2 reports that are accurate and complete.
- Ensure that the test plan for sprint 1 of release 2 is setup to provide the correct test reporting.
- Be sure that the reporting for sprint 2 of release 1 will not be affected by release 2, sprint 1 activities.

However, Christine has additional requirements that she does not have for a **non-release** sprint, which are to:

- Provide reporting for release 1 as a whole.
- Maintain a snapshot of the test cases at release.

NOTE

Maintaining an immutable record of artifacts as they exist when the software product is released is often referred to as a release baseline. A release baseline may contain many other things besides test cases. In fact, it often contains all the work items, source code, build definitions, documents, environments and any other engineering artifacts associated with the release. It is recommended not to create a baseline for going from sprint-to-sprint. If a baseline is needed, only create for a release.

The Release to Release Scenario

Our example project has just completed sprint 2 of release 1, and release 1 is being shipped. Our example project is continuing on immediately to the next release, **release 2**. Christine must now create a test plan for sprint 1 of release 2. Again, the test planning and management work involved in going to the next sprint of a new release shares much in common with going to the next sprint within the same release. Christine should consider the following items when determining the contents of the new test plan:

- The regression testing of release 1 work.
- New test cases for the planned implementation work in release 2.
- Incomplete testing and work deferred to release 2.

Recommended Practices

The example project is moving on to release 2 and has begun sprint 1 planning. As the test lead, Christine will need to:

- Review the user stories from release 1 and determine which ones will be regression tested in the first sprint of release 2.
- Close-out release 1 testing artifacts.
- Determine what test cases will be needed for the first sprint of the next release.
- Consider the design of the next test plan to make structural improvements and account for any changes in reporting requirements.

As sprint 2 is the last sprint of the release, consider the sprint 2 test plan as the test plan for the whole release. This will often make sense because the last sprint of the release will likely include all testing required for user acceptance. In our example project, Christine has renamed the sprint 2 test plan to the **Sprint 2 Test Plan - for Release 1**. This was used to verify and accept all the functionality and features of release 1.

The following sections cover typical scenarios related to managing test artifacts across releases. They include actions and reports that you might find useful when you build your own test release management plan.

> NOTE The recommended practices are set at the conceptual level. They are intended to convey the nature of the solution, but not the exact steps for performing the work. Please refer to the Test Planning and Management Release to Release Hands-on Lab for procedures and a hands-on example of executing release to release test case management.

Close Release 1

At the end of a release, review the work status and update test artifacts appropriately so that:

- Reports show release 1 as closed.
- Planning for sprint 1 of release 2 can be easily performed.
- All test artifacts are in the correct state and assigned to the correct iteration path.

Use MTM to update and close out the release 1 test plan.

Action	Guidance
Review Work Status	Identify: • User stories to be re-tested in sprint 1 of release 2. • Test cases to be regressed in sprint 1 of release 2. • Verified test cases. • Incomplete test cases where work has been deferred to sprint 1 of release 2. • Test cases not run in release 1 where work has been deferred to sprint 1 of release 2.
Verify Test Case State and Iteration Path	• All active test cases are in a Ready state. • All active test cases are assigned to the sprint 2 iteration path (...\Release1\Sprint2). • Verify that test cases that have never been executed are assigned to the backlog.
Verify Shared Step State and Iteration Path	• All active shared steps are in an Active state. • All active shared steps are assigned to the sprint 2 iteration path. • Verify that shared steps that have never been executed are assigned to the backlog.
Verify Share Parameters State and Iteration Path	• All active shared parameters are in an Active state. • All active shared parameters are assigned to the sprint 2 iteration
Update the Release 1 Test Plan	• Set the **verified** suites testing state to **Completed**. • Set the release 1 test plan testing state to **Completed**. • Set the release 1 test plan properties state to **Inactive**.

Table 14 - Update and close out the release 1 test plan

When Creating a Baseline is required

The method for creating a baseline will vary depending on the method used for work item classification (see above section on **Alternatives for Work Item Classification**). The following table summarizes the differences depending on whether the Declarative Model or Latest Model of work item classification is used.

Action	Guidance
Declarative	• The baseline will consist of the work items assigned to sprints under the iteration path release 1 node. • After the release these work items become the baseline and are no longer changed.

Latest	• The baseline will consist of the work items assigned to:
	  ○ The area path root.
	  ○ The sprints completed in release 1.
	• These work items become the baseline once their area path is set to release 1 (...\Release1). Work items under this area path node can then be made read-only.

Table 15 - Differences depending on whether the Declarative Model or Latest Model

After this the steps are the same for both models. Any work items to be carried forward to the next release are cloned to the backlog. Typically, this will be active bugs, ready test cases and active shared steps.

In our example project, the Declarative Model of work item classification is being used. However, the principles and practices described in this guide can be applied to the Latest Model as well.

Setup the New Test Plan

During release 2 planning, you need to create a release 2 test plan. The following table describes the basic alternatives for creating a test plan for a new release:

Approach	MTM Feature	Use When	Use	Description
Baseline	Clone Test Plan	You are required to baseline releases and all work items from release 1 must be frozen	Automatically create the new test plan from the **Organize**, **Test Plan Manager** view by selecting the copy-from test plan and click the **Clone** button on the menu bar. Cloning **Test Cases** directly is not available via MTM UI. To do so, use **Create copy and add to suite** feature in MTM by right clicking on Test Case of choice.	Duplicates the test cases and work items linked to them. See guidance on Cloning test plans using Microsoft Test Manager.[17]
Single Plan	Create Test Suites by Referencing Existing Test Cases	You are not creating a baseline for release 1. This is no different than going from Sprint-to-Sprint.	Manually create a new test plan. In the **Plan**, **Contents** view, right-click the parent suite in the new test plan and select **Create test suites by referencing existing test cases**. Then add new test cases.	Reuses the same test cases. The suite is copied into the new plan, but there is no duplication of test cases.
Separate Regression Plan	Same as above	Same as above	Same as above except new test cases are added to a separate test plan	Same as above

Table 16 – Alternatives for creating a test plan

For a new test plan:

Action	Guidance
Set Test Plan Properties	• Include the sprint number in the plan name. • Set the sprint test plan testing state to **In planning**. • Assign the start and end dates to the test plan so they match the sprint start and end dates. • Set the test plan iteration path to \project\Release2\Sprint1. • Create one or more suites for the new test cases developed for release 2 features.
Create Regression Test Suites	• Setup your regression test cases using one of the approaches listed above.

Table 17 - Create and setup the sprint 1 test plan

Action	Guidance
Verify the Cloned Test Cases in the New Test Plan	• Verify the test cases and shared steps in the new suite are new work items. • Verify the relationships and iteration path in the test cases

[17] http://blogs.msdn.com/b/visualstudioalm/archive/2013/05/03/cloning-test-plans-using-microsoft-test-manager.aspx

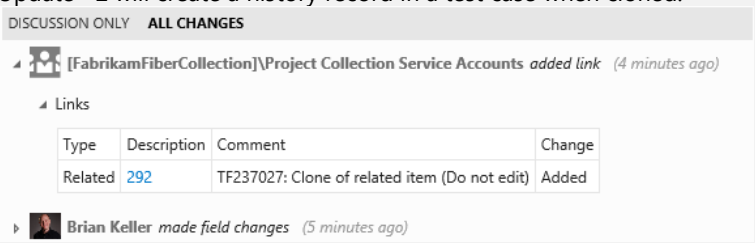
Action	Guidance
When the project is ready to start testing in sprint 1	• Set the sprint 1 test plan testing state to **In progress**.

Table 18 – Verify the cloned test cases

NOTE

Marking regression test cases

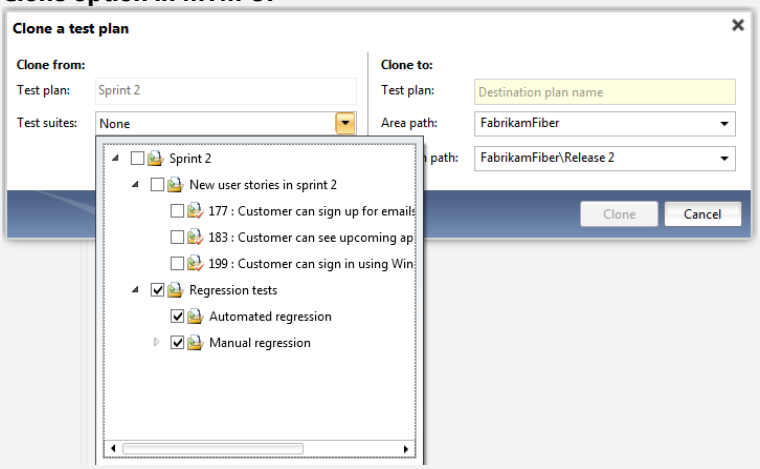
The test cases that are part of the release 1 baseline were duplicated. These duplicates are used in release 2 to support regression testing and to complete any testing not performed in release 1. New features in Visual Studio Update #2 will create a history record in a test case when cloned.



Prior to that you may want to mark these duplicates using a custom reason code, adding a custom indicator field, history comment or in some other way so it will be easy to identify them for assignment to a test plan.

NOTE

Clone option in MTM UI



- When cloning a Test Plan with (RBS, QBS and related Test Cases), **Shared Parameters** of the cloned Test Cases are **NOT** cloned.
- When cloning a Test Plan with (RBS, QBS and related Test Cases), **Shared Steps** of the cloned Test Cases are cloned.

Reporting

Reporting and communicating the current progress, status and findings of the test effort is perhaps the most important task in testing. In order to accomplish this task you can use the Lightweight charting capability of the Test Hub to create charts based on test cases and test outcome. Those charts can then be pinned to the team home page for easy charring and visibility to the team.

There is also other capabilities for reporting. MTM contains a limited set of graphs to illustrate and report on test status in the context of the current work. MTM also got a simple Reporting Tab.

For more advanced reporting needs, there is an option to install and use SQL Server Reporting Services to get additional and rich reports. There are several test reports packaged with the process templates, but you can add reports from community sources such as TFS community report pack, or by creating your own custom reports. This option is however not available for Visual Studio Online.

Tips for tracking and reporting

The Test Charts automatically aggregate information for all sub suites. Using static suites to structure your test plan will make it possible to create and aggregate charts corresponding to the structure. You can use the test charts to either create a view on the aggregated top level, or use them to highlight important suites in in the test plan..

Unfortunately, if you want to provide the detailed charts for all suites in the test plan it will require a lot of duplicating and copy & paste, or use other reporting solutions like SQL Server Reporting Services.

Track and report on test authoring progress

By creating charts on test cases and their status you can report on the progress and readiness of the test plan.

You can also create work item queries to list requirements without test cases to keep track of the number of requirements without test coverage.

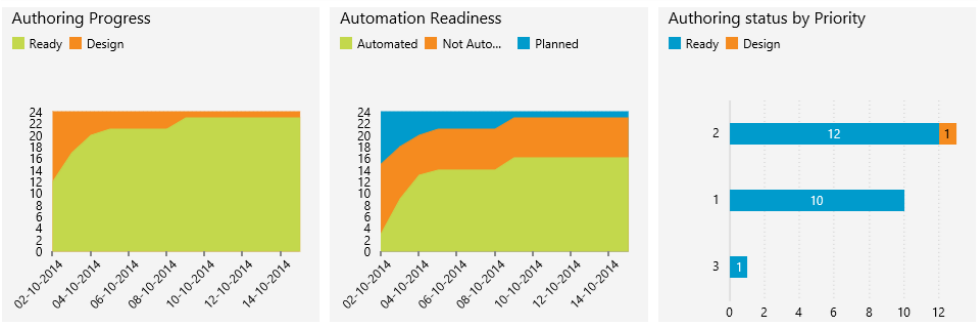


Figure 16 - Track and report on test authoring progress

Track and report on test execution progress

By creating Test Result charts and group them by tester, configuration or priority you can track and report on the progress of test execution.

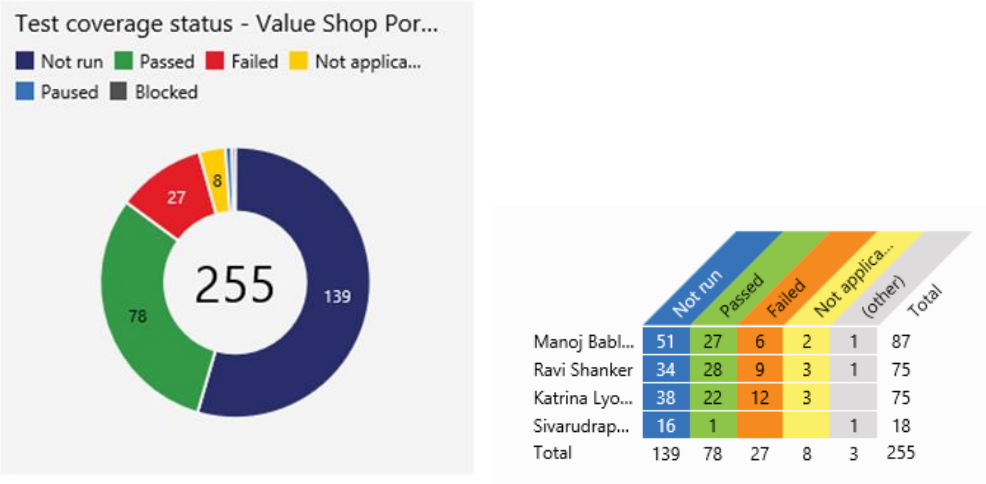


Figure 17 - Track and report on test execution progress

Track and report on test status and outcome

By creating Test Result charts and group them by suite and outcome track and report on the outcome of the testing effort.

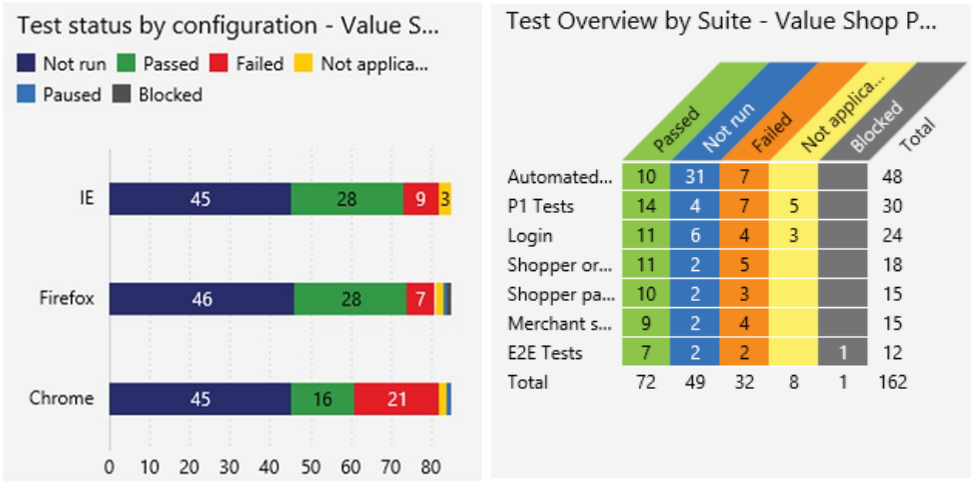


Figure 18 - Track and report on test status and outcome

Track and report on bug status

By creating work item queries on bugs, you can create charts showing bugs by State, Severity & Assigned

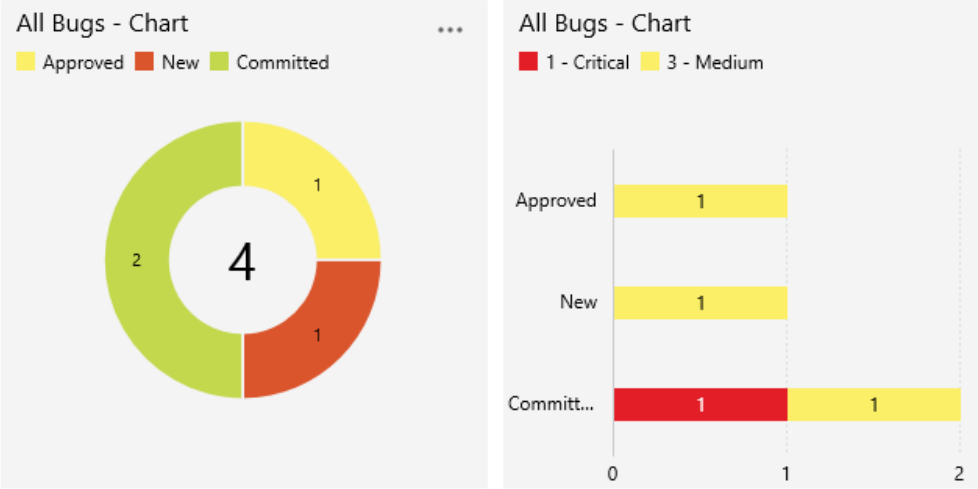


Figure 19 - Track and report on bug status

Advanced reporting

If you have needs for even more advanced reporting there is several other options available for you.

Reporting Solution	TFS On-Premises	VSO	Required knowledge and more info
SSRS	√	x	Using **default** reports: • MSDN Default Reports in Reporting Services[18] • Community report extensions[19] Create **custom** reports : • SQL or MDX and Reporting Service knowledge needed! • MSDN Create, Customize, and Manage Reports for Visual Studio ALM[20] Tabular Store Guidance • TFS Reporting Guide[21] Focused on providing practical guidance and a reference solution to enable VSO users to create a tabular store model and generate reports based on WIT data, and to enable TFS users to create valuable reports using the TFS Data Warehouse, based on real-world scenarios.

Table 19 – Advanced Reporting

[18] https://msdn.microsoft.com/en-us/library/bb649552(v=vs.110).aspx#oob_srs

[19] https://communitytfsreports.codeplex.com/

[20] https://msdn.microsoft.com/en-us/library/bb649552(v=vs.110).aspx

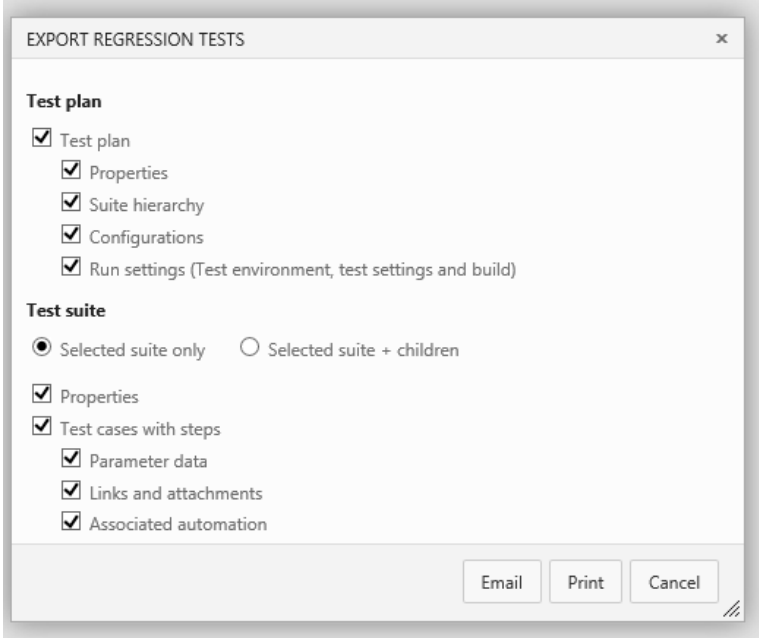
[21] https://vsarreportguide.codeplex.com/

Export

The export feature, introduced with Team Foundation Server 2013 Update 2 or later [22], allows you to export test plans, test suites and test cases using Team Web Access for **offline** reporting, viewing, emailing or printing.

This feature is intended to replace the functionality provided by the Test Scribe [23] extension available on the visual studio gallery for older versions of Microsoft Test Manager.

Using the export feature

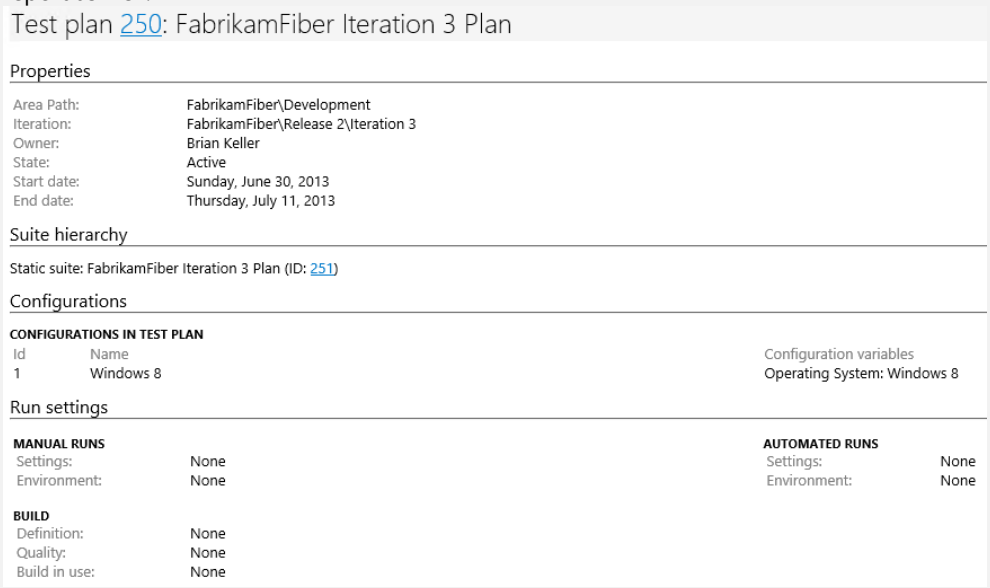
Step	Instructions
1 Select Export ⛶ - Done	• Navigate to the Test Hub and right click on the test plan, to document the entire test plan, or a test plan suit, to document a suite, and select Export. • Alternatively, highlight the **plan** or **suite** and use the export toolbar **button**  . • Once export is selected you will be presented with a dialog box where you can identify which testing artifacts you want to include in your report. 
2 Review Report ⛶ - Done	• The top of the report will contain the information corresponding to the Test Plan selections you made in the dialog above.

[22] http://go.microsoft.com/fwlink/?LinkId=392762

[23] https://visualstudiogallery.msdn.microsoft.com/e2c743ce-d9f1-4ed0-ab08-a4777bef06a4

Step	Instructions

- **Properties** will include information on the plan such as area path and iteration, plan start and end date, and plan owner. A hierarchy of the test suites may follow which includes the suite names and hierarchy and hyperlinks to each suite. A list of all configurations included in the plan or referenced by the suite would come next. Finally environment, and build information may be included in the report as well.



Test plan 250: FabrikamFiber Iteration 3 Plan

Properties

Area Path:	FabrikamFiber\Development
Iteration:	FabrikamFiber\Release 2\Iteration 3
Owner:	Brian Keller
State:	Active
Start date:	Sunday, June 30, 2013
End date:	Thursday, July 11, 2013

Suite hierarchy

Static suite: FabrikamFiber Iteration 3 Plan (ID: 251)

Configurations

CONFIGURATIONS IN TEST PLAN

Id	Name	Configuration variables
1	Windows 8	Operating System: Windows 8

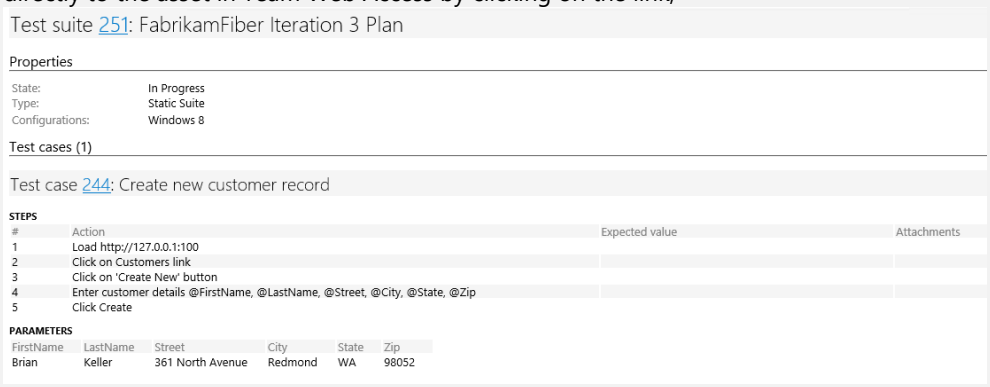
Run settings

MANUAL RUNS		AUTOMATED RUNS	
Settings:	None	Settings:	None
Environment:	None	Environment:	None

BUILD

Definition:	None
Quality:	None
Build in use:	None

- The **test suite** portion of the export may be configured to include the properties of the suite which includes the state and type of suite and also includes which configurations were used as a part of the testing, following that will be a list of all test cases included in the suite. This section would contain a summary of the test case, a list of the test steps, parameter information, and even links to associated bugs or automation. All links are exported as hyperlinks so a consumer of the export may navigate directly to the asset in Team Web Access by clicking on the link,



Test suite 251: FabrikamFiber Iteration 3 Plan

Properties

State:	In Progress
Type:	Static Suite
Configurations:	Windows 8

Test cases (1)

Test case 244: Create new customer record

STEPS

#	Action	Expected value	Attachments
1	Load http://127.0.0.1:100		
2	Click on Customers link		
3	Click on 'Create New' button		
4	Enter customer details @FirstName, @LastName, @Street, @City, @State, @Zip		
5	Click Create		

PARAMETERS

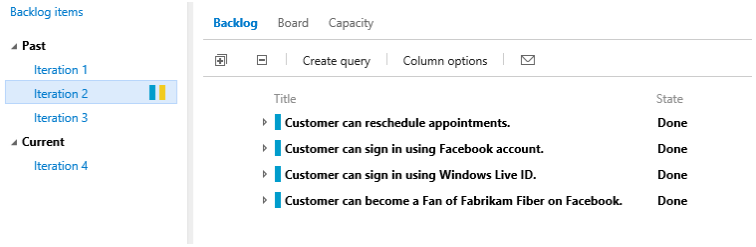
FirstName	LastName	Street	City	State	Zip
Brian	Keller	361 North Avenue	Redmond	WA	98052

Walkthroughs

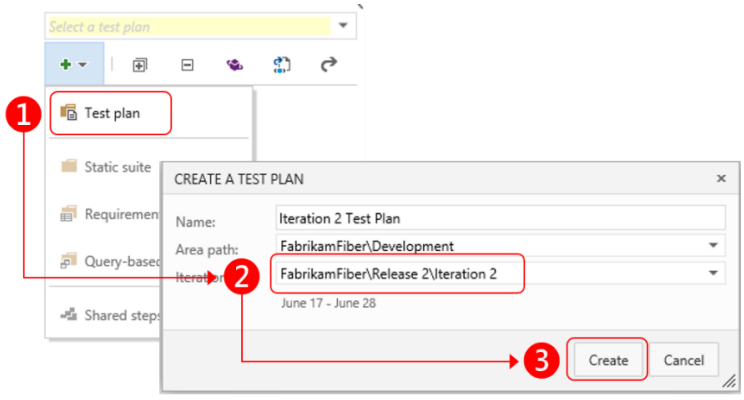
Requirements Based Suite

This walkthrough will show you how to create a requirements based suite of tests to test against your requirements.

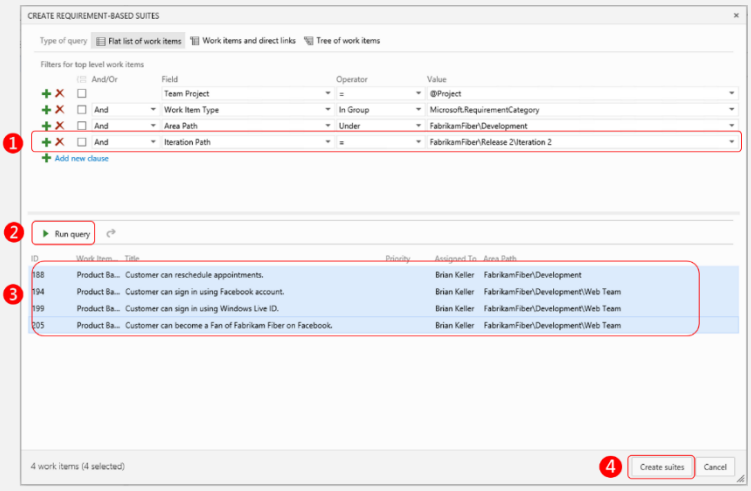
Before we begin

Step	Instructions
1 User Stories ⛶ - Done	Before you can create a suite of tests you need to make sure you have the requirements assigned to an iteration backlog. • If they don't exist, create a series of user stories and place them into your backlog • Move a few stories into an active iteration NOTE In this example, we are going to use **Iteration 2** from the **FabrikamFiber / Fabrikam Fiber Web Team** backlog 

Creating a Requirements Based Test Suite

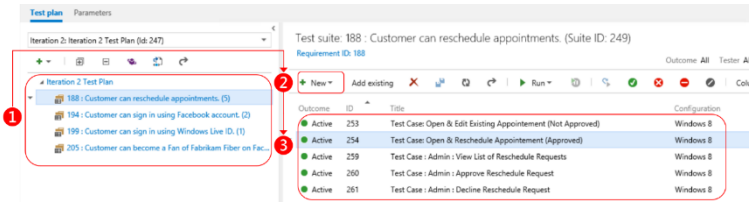
Step	Instructions
1 Create Test Plan ⛶ - Done	• Make sure you are in the **FabrikamFiber / Fabrikam Fiber Web Team** project • Click on the **TEST** hub link • Click the + link and select **Test Plan** • Enter your test plan name and select **Iteration 2** from the iteration list 
2 Create requirement-based suite	• Click the tick icon **(>)** next to the newly created test plan • Select **New requirement based-suite** • Modify the query to include all requirements that are in Iteration 2 • Click Run query • Select all work items

Step	Instructions
⬚ - Done	• Click Create suites button



> **NOTE** If you are testing requirements from sprint to sprint you will want filter the requirements list by iteration path. On the other hand, if you are doing a full regression test, you probably don't. You have the flexibility to create the query of choice to select the appropriate requirements to complete your testing.
>
> *If you add new requirements to your iteration after your test plan is created, you will need to go back into test plan and add the additional requirement(s).*
>
> We do not think test cases are assigned to iterations. Typically user stories and bugs are tracked with iterations. And you use the query editor to pick them to create RBS. QBS are apt for iterations. If test cases are tracked with iterations, then they would disappear from sprint n QBS once they are moved to sprint n+1 QBS is created and test cases updated to point to sprint n + 1.

Step	Instructions
3 Create Test Cases ⬚ - Done	• Under the plan name you will see your added requirements • Add test cases with test steps to each requirement

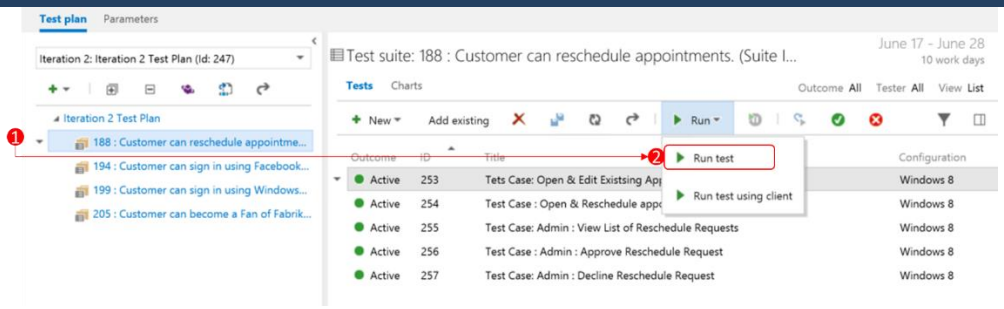


> **NOTE** Try using the "New test case using grid" option when creating test cases. It makes it a lot easier to add multiple test cases in a single batch.

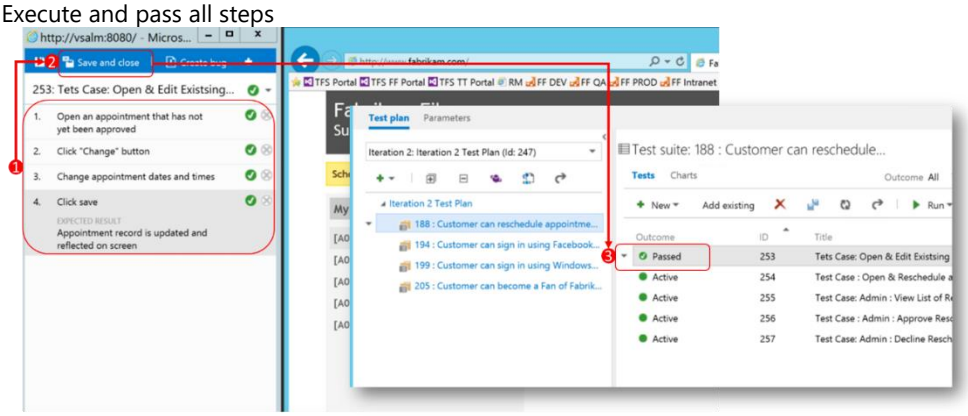
Run Tests

Step	Instructions
1 Run your tests ⬚ - Done	• In your test plan, select a requirement • Select a test case • Click ▶ Run ▾ and select ▶ Run test

Step	Instructions



- The web test case runner will open
- Execute and pass all steps



- Click Save and close
- Notice the first test case is now set to Passed.

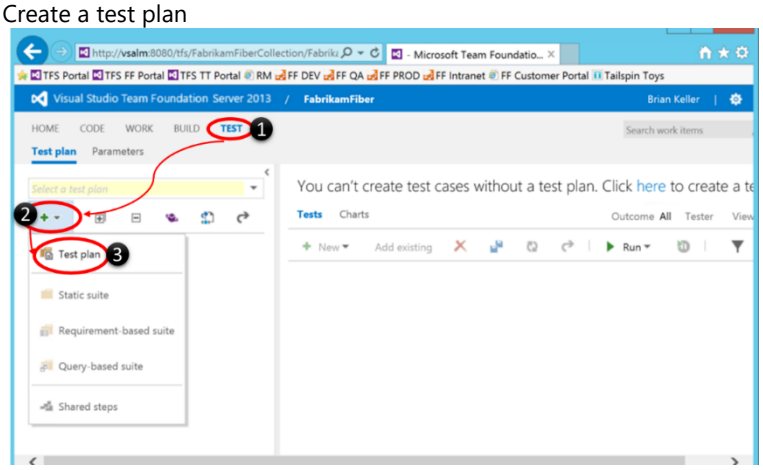
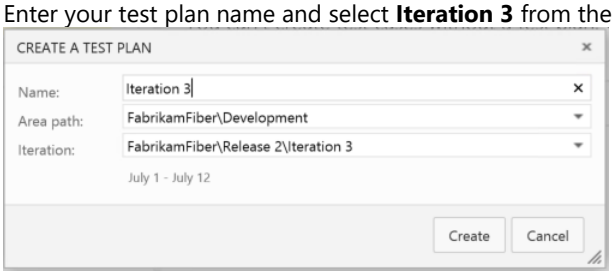
Tips and Tricks

- If you add new requirements to your iteration after your test plan is created, you will need to go back into test plan and add the additional requirement(s).
- Use the Grid view to quickly add test cases and test steps.
- Try sticking with the TFS Test Hub for managing your test cases and use MTM to run the test cases.

Adding Test Plan and Test Suites

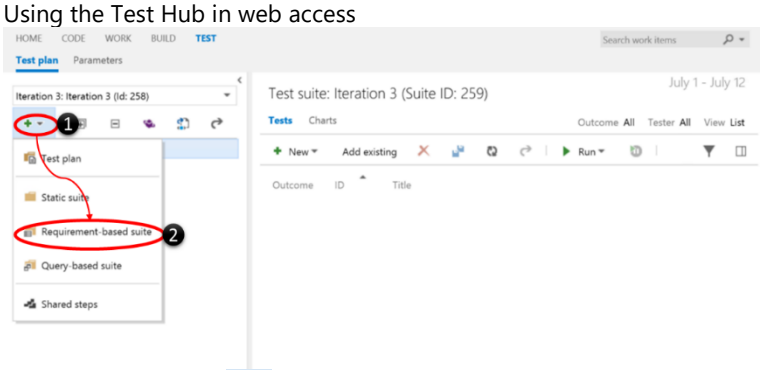
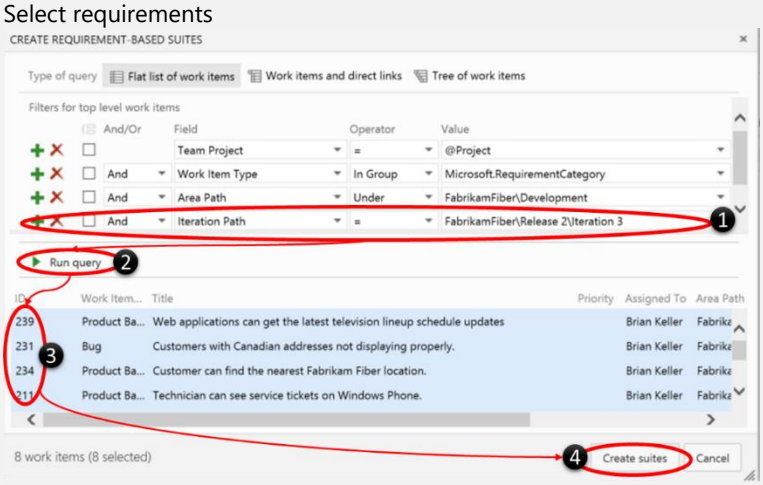
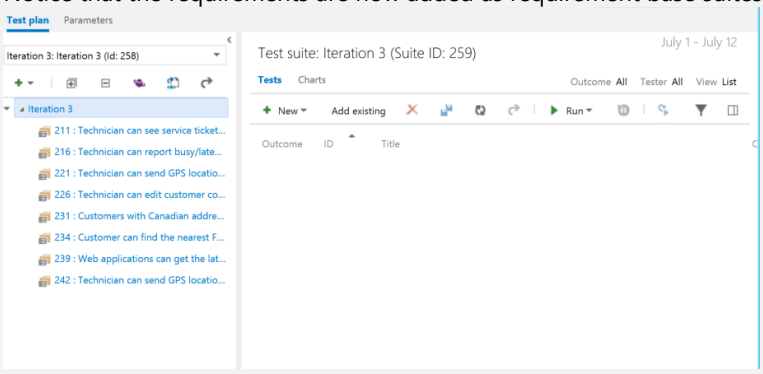
This walkthrough will show you how to create a test plan and adding various types of test suites using Test Manager as well as though the test hub in web access.

Create a Test Plan using Web Access

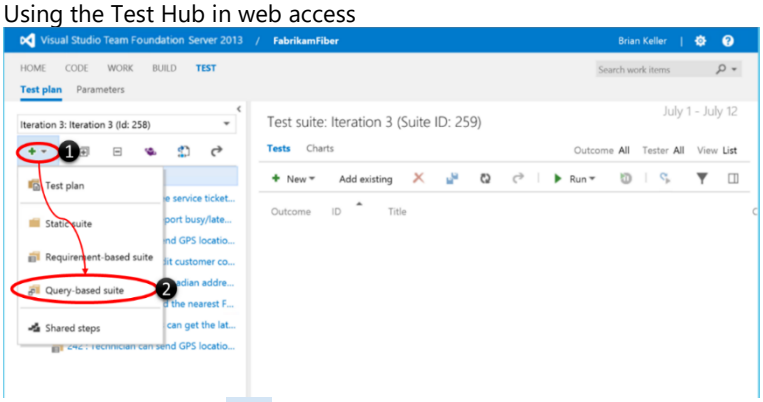
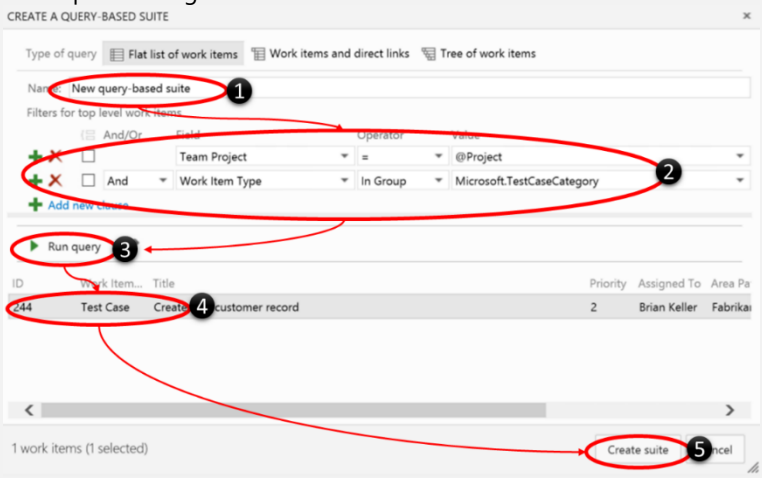
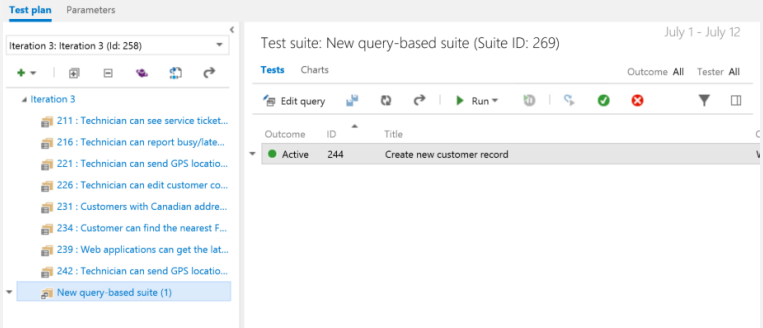
Step	Instructions
1 Create Test Plan ☐ - Done	• Open up a browser and navigate to the Fabrikam Fiber TFS portal (http://vsalm:8080/tfs/FabrikamFiberCollection/FabrikamFiber) • Create a test plan  ○ Select the **Test** tab to navigate to the test hub ○ Select the add ( ) button and then ○ Select "**Test Plan**" to create a new test plan • Enter your test plan name and select **Iteration 3** from the iteration list  • Click "**Create**

NOTE: To configure more advanced settings (such as the Run Settings) you can click the "Open test plan using Microsoft Test Manager" button ( ) to open the test plan in Microsoft Test Manager

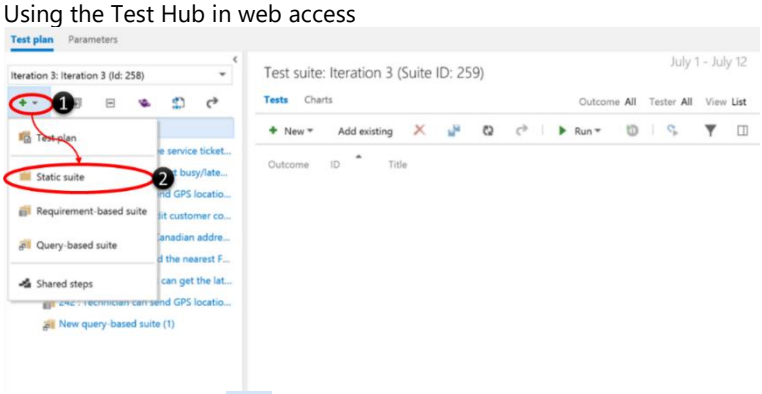
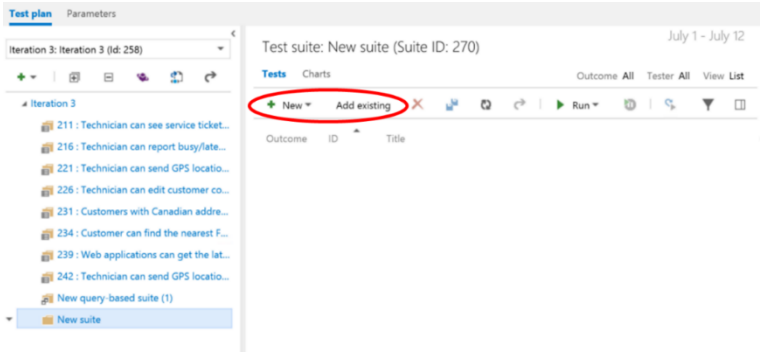
Create Requirements Based Test Suites using Web Access

Step	Instructions
1 Add Requirements based Test Suite ⛶ - Done	• Using the Test Hub in web access  ○ Select Add (+) ○ Select "**Requirements-based suite**" (Requirement-based suite)
2 Select Requirements ⛶ - Done	• Select requirements  ○ Add the iteration path "**FabrikamFiber\Release 2\Iteration 3**" ○ Select "**Run query**" to execute the query ○ Select all the requirements that are returned ○ Select "**Create suites**" • Notice that the requirements are now added as requirement base suites ( ) to the test plan 

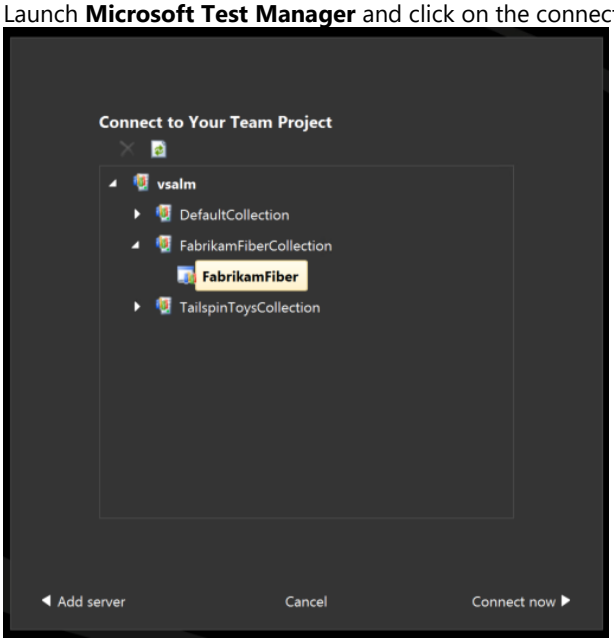
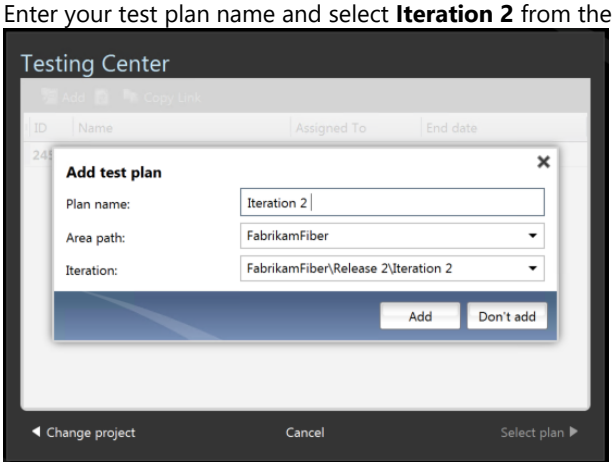
Create Query Based Test Suites using Web Access

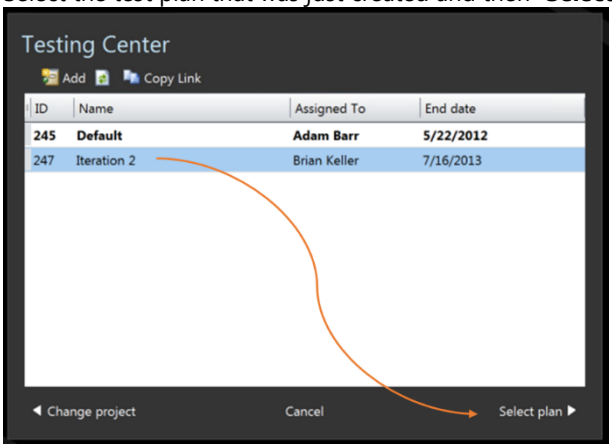
Step	Instructions
1 Add Query based Test Suite ⬚ - Done	• Using the Test Hub in web access  ○ Select Add (+ ▾) ○ Select "**Query-based suite**" (Query-based suite)
2 Select test Cases ⬚ - Done	• Select pre-existing test cases  ○ Give the test suite an appropriate **Name** ○ Modify the criteria to return pre-existing test cases ○ **Run** the query ○ Select all the test cases that has been returned ○ Select "**Create suite**" to create the new test suite • Notice that the query based suite ( ) has been added to the test plan 

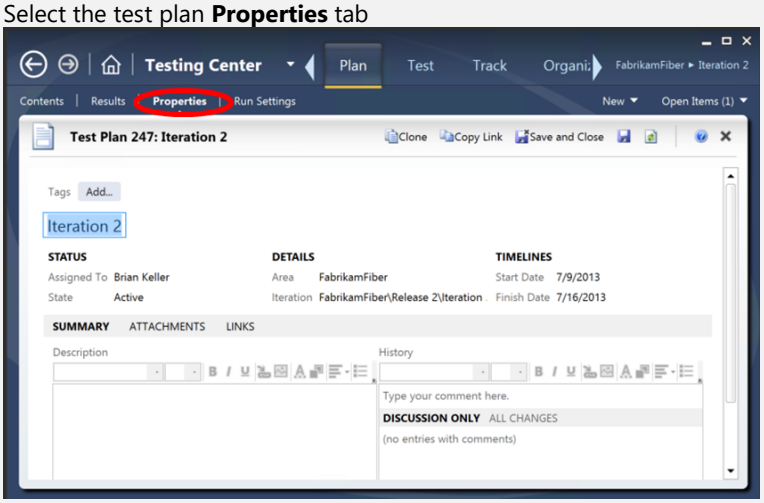
Create Static Test Suites using Web Access

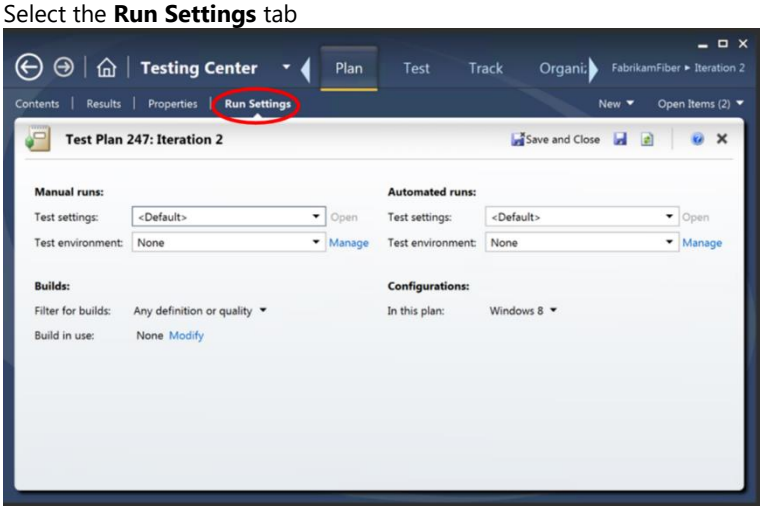
Step	Instructions
1 Add Static Test Suite ⬚ - Done	• Using the Test Hub in web access  ○ Select Add (+) ○ Select "**Static suite**" (Static suite) • Give the suite a descriptive name • Use **New** (New) to add new test cases and **Add existing** (Add existing) to add existing test cases in the selected suite 

Create a Test Plan using Microsoft Test Manager

Step	Instructions
1 Create Test Plan ⬚ - Done	• Launch **Microsoft Test Manager** and click on the connection dialog if it is not already open  • Select the "**FabrikamFiber**" team project and select "**Connect Now**" • Click the **Add** () to add a new Test Plan • Enter your test plan name and select **Iteration 2** from the iteration list  • Select "**Add**"

Step	Instructions
	• Select the test plan that was just created and then "**Select Plan**" 

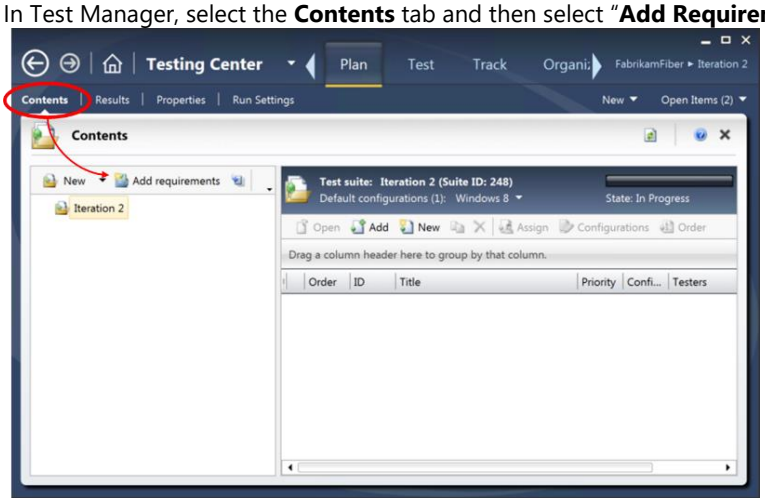
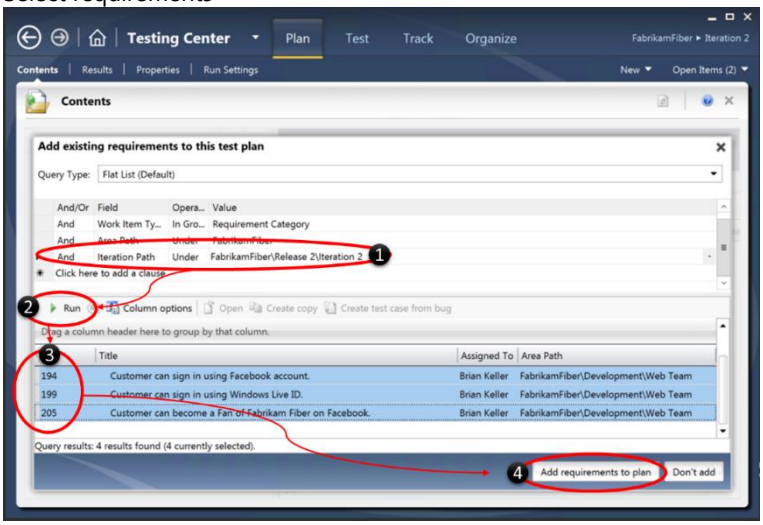
2
Review and modify Test Plan Properties
⬚ - Done
	• Select the test plan **Properties** tab
	 
	• Review and modify the details such as
	  ○ the **Timelines** and
	  ○ Test plan **Summary**

3 Review and modify the Run Settings ⬚ - Done	• Select the **Run Settings** tab  • Review and modify settings such as ○ **Manual run** settings and environments

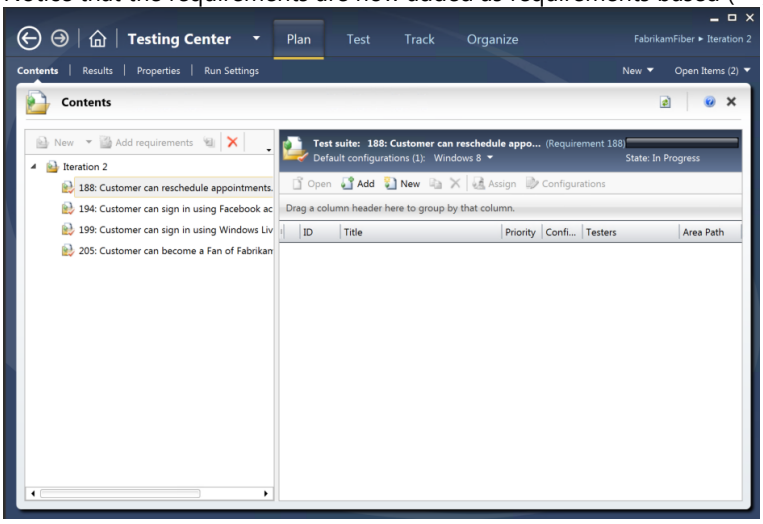
Step	Instructions
	○ **Automated runs** settings and environments ○ Select the **build** that is being testing ○ And the associated **configurations** that need to be verified as part of this test plan

NOTE Run Settings can only be configured using Microsoft Test Manager and not yet though TFS Web Access Test hub

Create Requirements Based Test Suites using Microsoft Test Manager

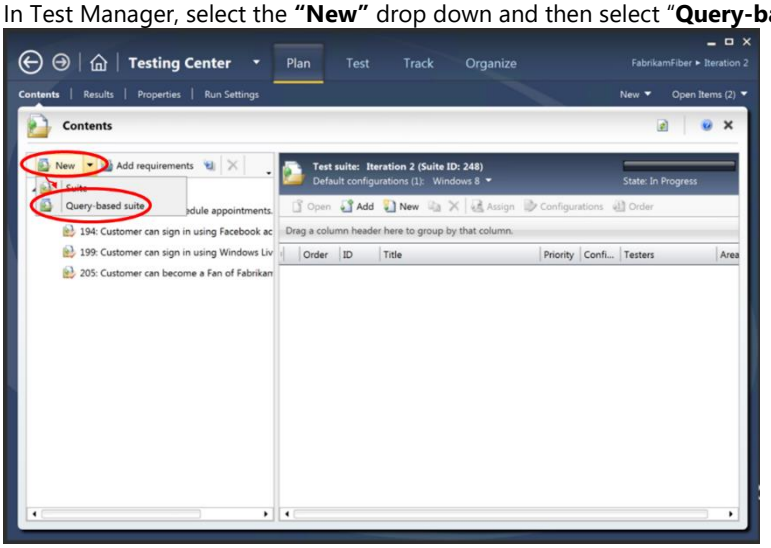
Step	Instructions
1 Add Requirements based Test Suite ☐ - Done	• In Test Manager, select the **Contents** tab and then select "**Add Requirements**"  • Select requirements  ○ Add the iteration path "**FabrikamFiber\Release 2\Iteration 2**" ○ Select "**Run**" to execute the query ○ Select all the requirements that are returned ○ Select "**Add requirements to plan**"

Step	Instructions
	• Notice that the requirements are now added as requirements based ( ) suites to the test plan



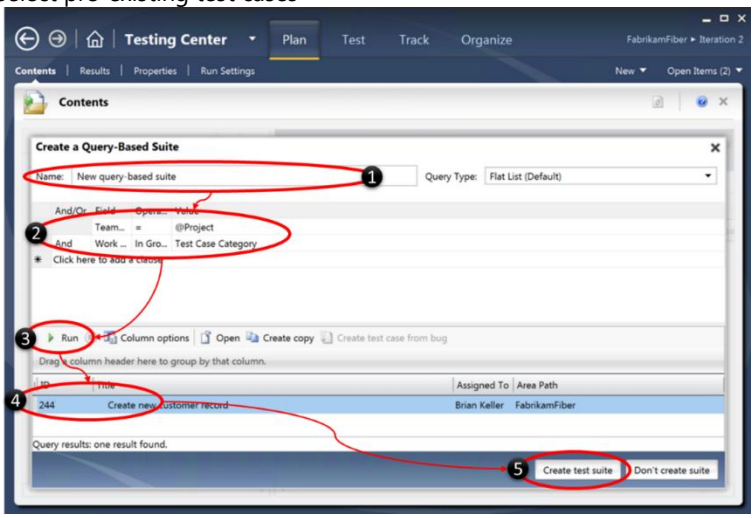
Create a Query Based Test Suite using Microsoft Test Manager

Step	Instructions
1 Add Query based test suite to the test plan ⬚ - Done	• In Test Manager, select the **"New"** drop down and then select "**Query-based suite**"

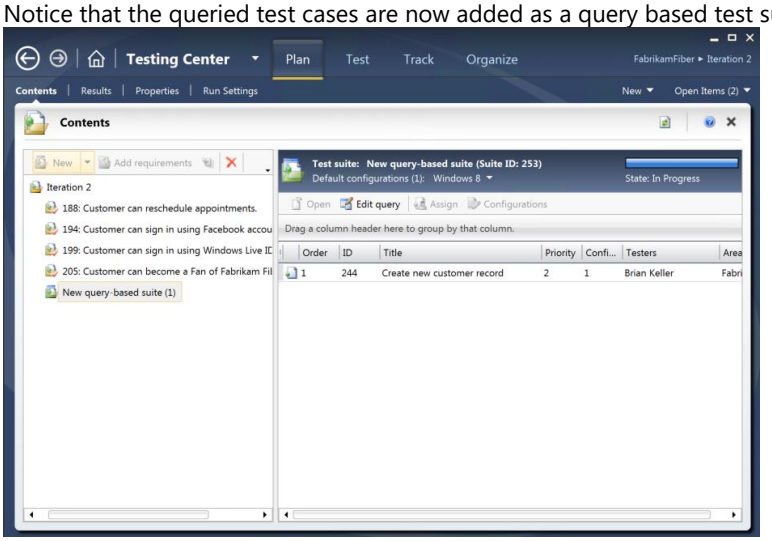


Step	Instructions

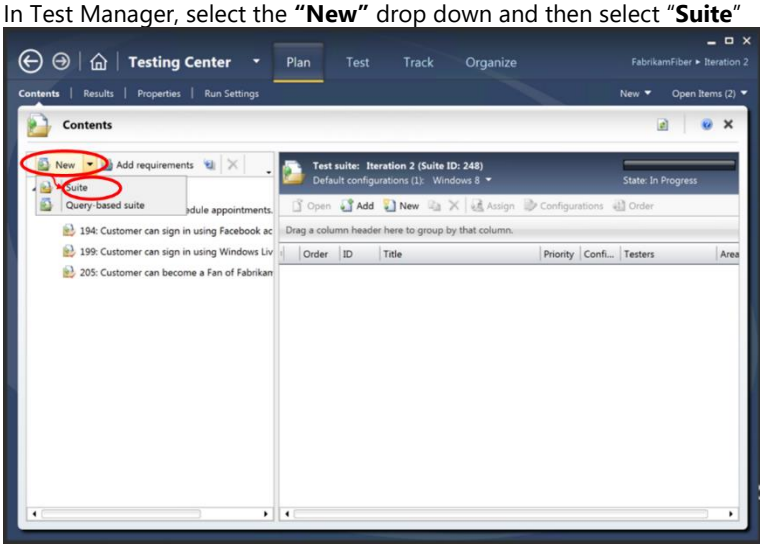
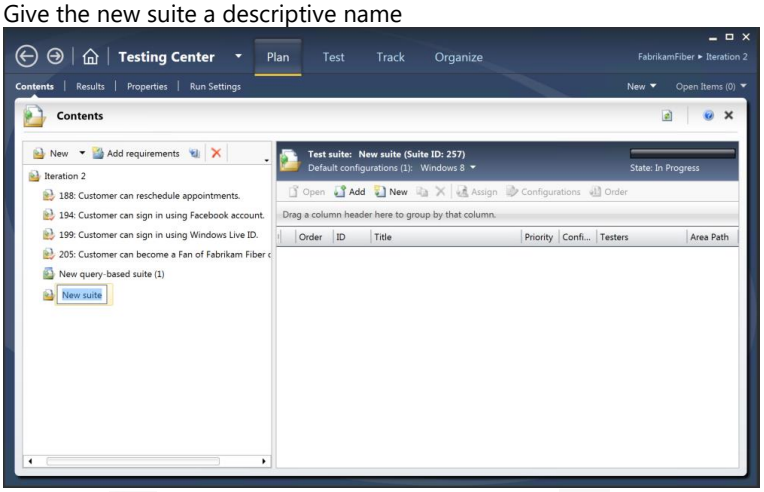
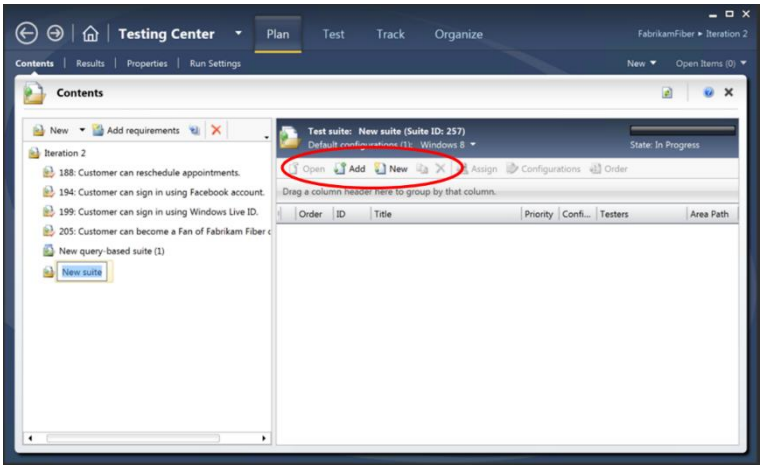
- Select pre-existing test cases



  - Give the test suite an appropriate **Name**
  - Modify the criteria to return pre-existing test cases
  - **Run** the query
  - Select all the test cases that I returned
  - Select "**Create test suite**" to create the new test suite
- Notice that the queried test cases are now added as a query based test suite ( )to the test plan



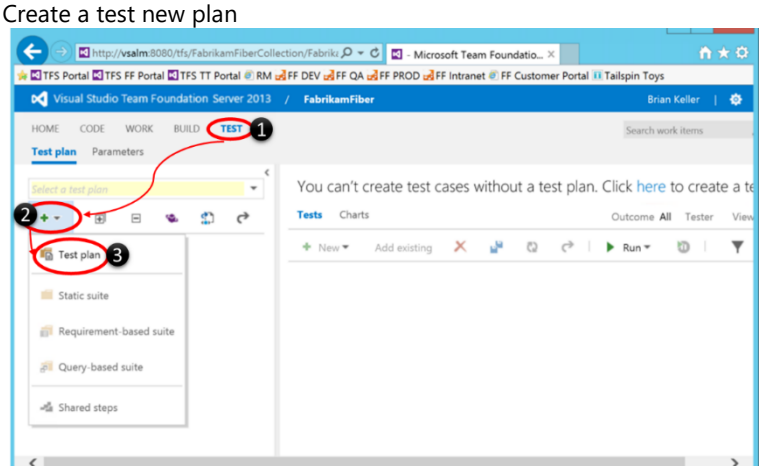
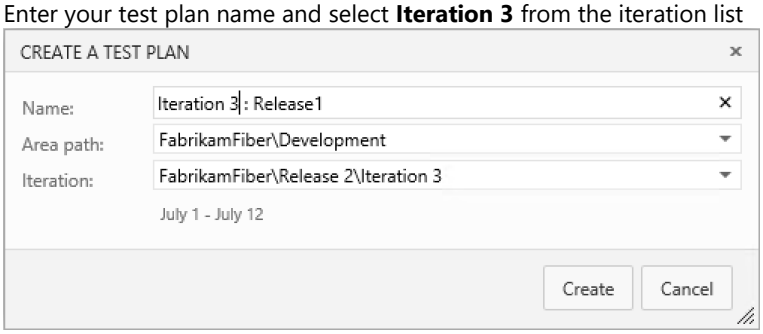
Create a Static Test Suite using Microsoft Test Manager

Step	Instructions
1 Add Static test suite to the test plan ⬚ - Done	• In Test Manager, select the **"New"** drop down and then select "**Suite**"  • Give the new suite a descriptive name  • Use **Add** (Add) to add existing Test cases and **New** (New) to create new test cases in the selected suite 

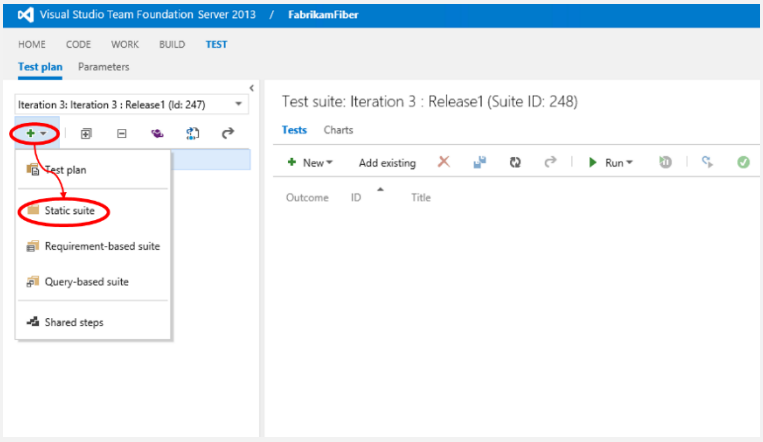
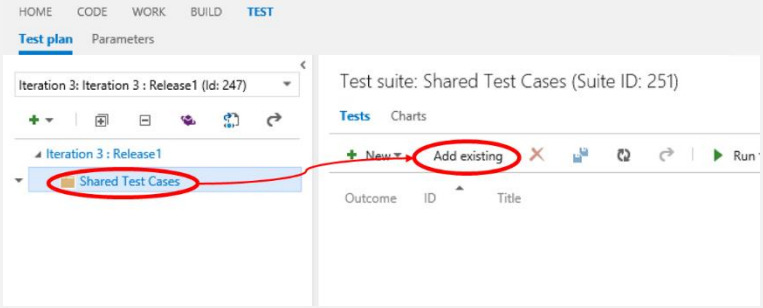
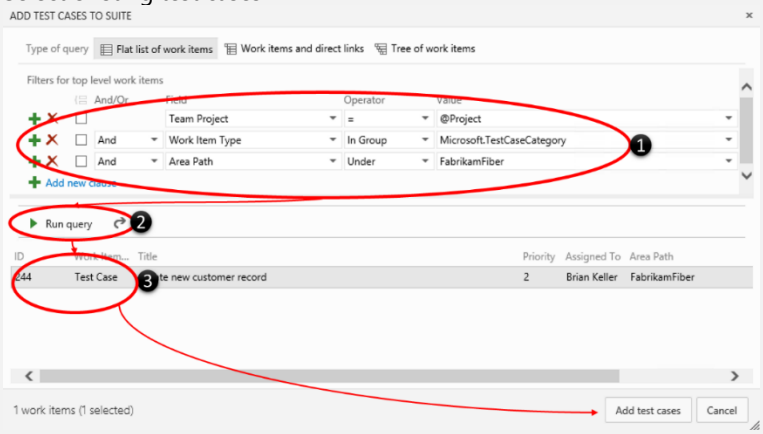
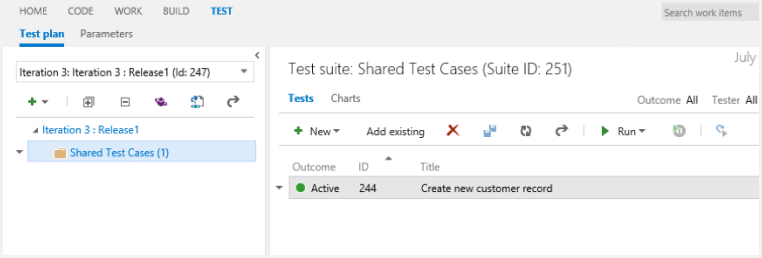
User Acceptance Testing

This walkthrough will show you how to create a test plan that is specifically focused on user acceptance testing of a release using the Microsoft Test Manager (MTM) and Team Web Access (TWA) clients.

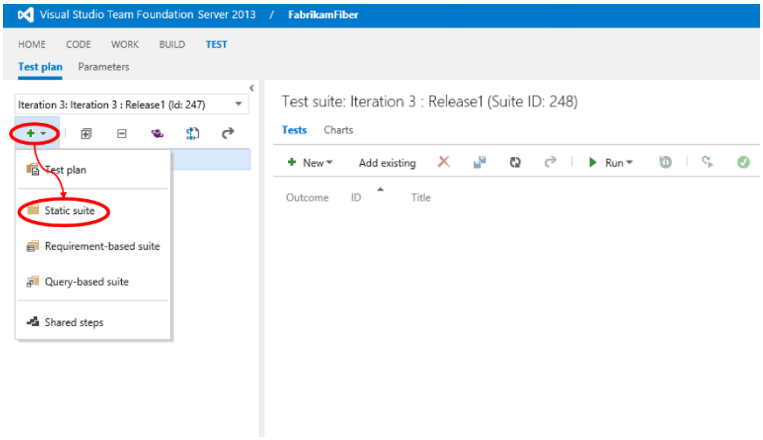
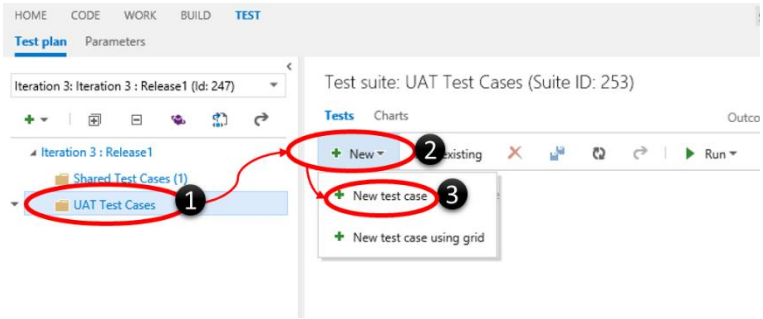
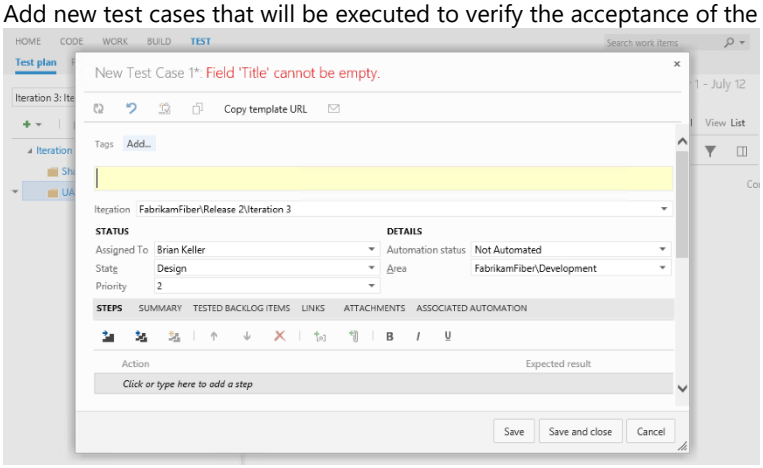
Create a new test plan for the release branch using TWA

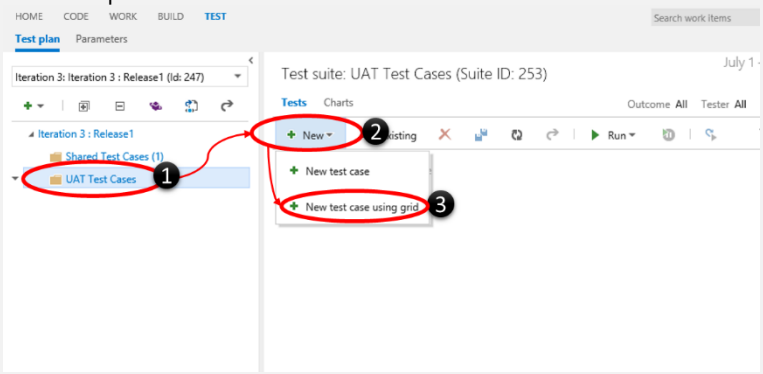
Step	Instructions
1 Create Test Plan ⬚ - Done	• Open up a browser and navigate to the Fabrikam Fiber TFS portal (http://vsalm:8080/tfs/FabrikamFiberCollection/FabrikamFiber) • Create a test new plan  ○ Select the **Test** tab to navigate to the test hub ○ Select the add (@22) button and then ○ Select "**Test Plan**" to create a new test plan • Enter your test plan name and select **Iteration 3** from the iteration list  **GEM** Include the branch name and sprint number in the plan's name • Select "**Create**"

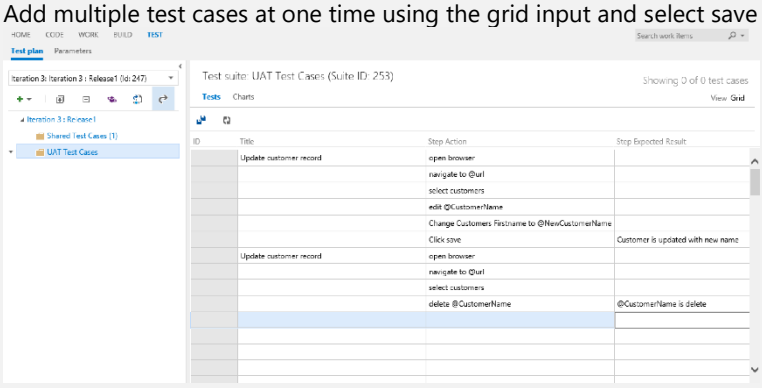
Step	Instructions

2
Assign Shared Test Cases
☐ - Done
	• Select the test plan that was just created and then select add ( ) and then "**Static suite**" then name the test suite "Shared Test Cases"
	 
	• Select the test suite and select "**Add existing**"
	 
	• Select existing test cases
	 
	  ○ Specify the search criteria to find existing test cases
	  ○ Run the query
	  ○ Select all the relevant test cases
	  ○ Select "**Add test cases**"
	• Repeat the above step until all relevant pre-existing test cases have been added to the test suite
	 

Step	Instructions

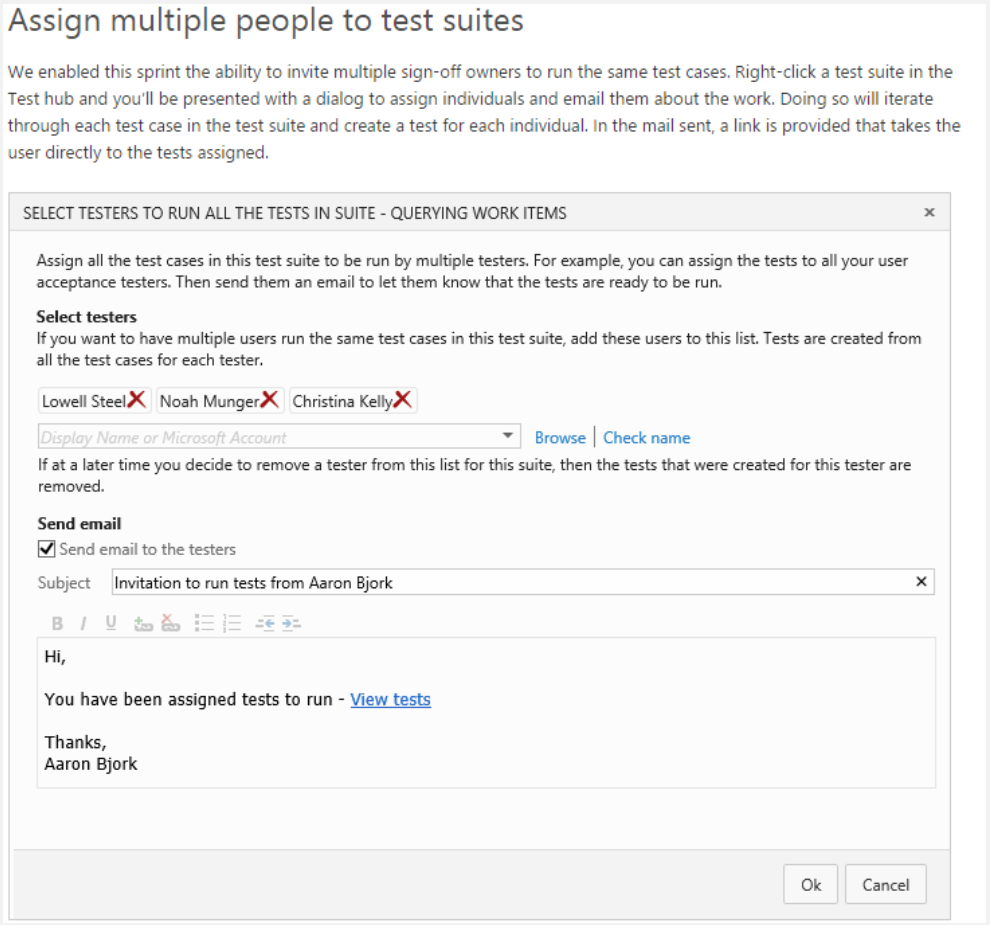
3
Create new UAT Test Cases one at a time
⬚ - Done
	• Select the test plan that was just created and then select add (+) and then "**Static suite**" and name the test suite "UAT Test Cases"
	 
	• Create a new individual test case
	 
	  ○ Select the test suite
	  ○ Select the "**New**" drop down
	  ○ Select "**New test case**"
	• Add new test cases that will be executed to verify the acceptance of the deployment
	 

Step	Instructions
4 Create new UAT Test Cases using a grid ⬚ - Done	• Add multiple test cases at once  ○ Select the test suite ○ Select the "**New**" drop down ○ Select "**New test case using grid**" • Add multiple test cases at one time using the grid input and select save ( )

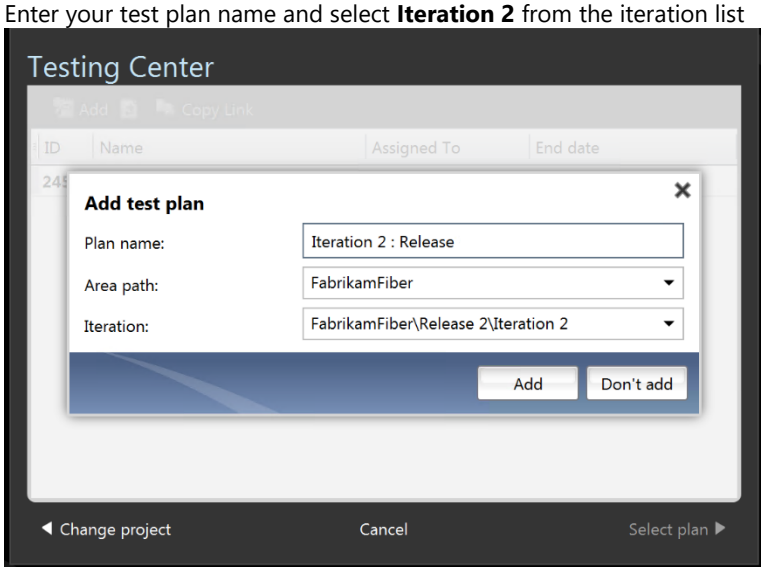


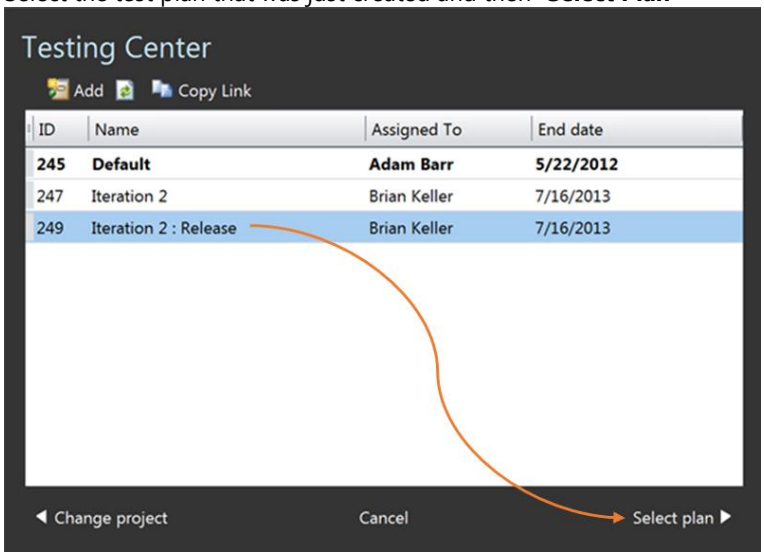
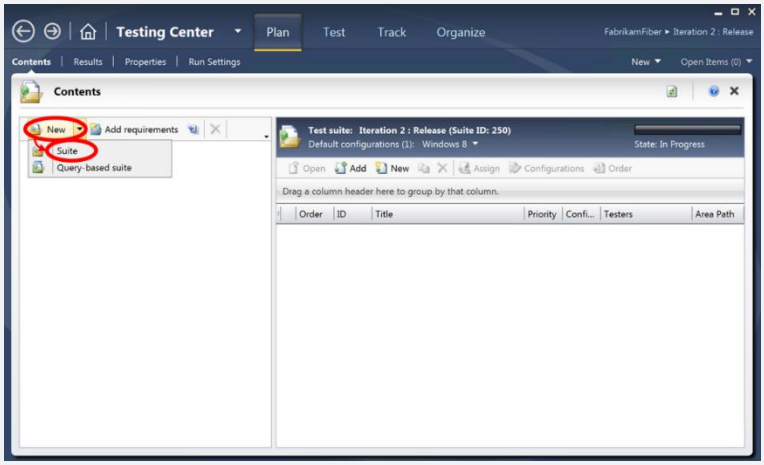
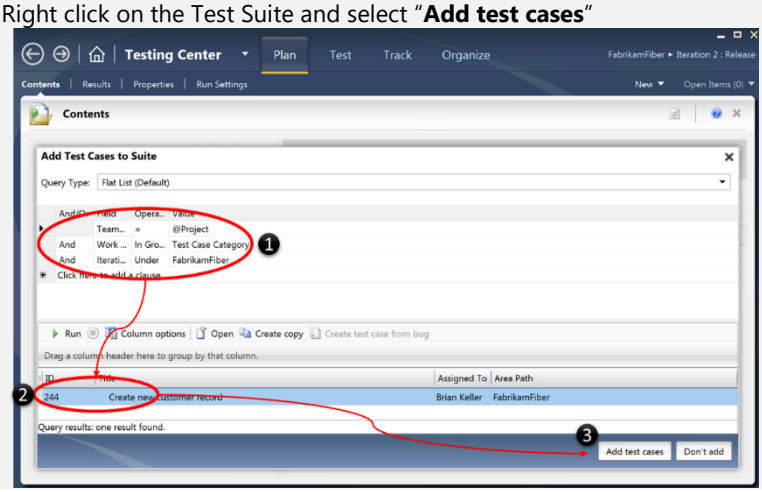
NOTE

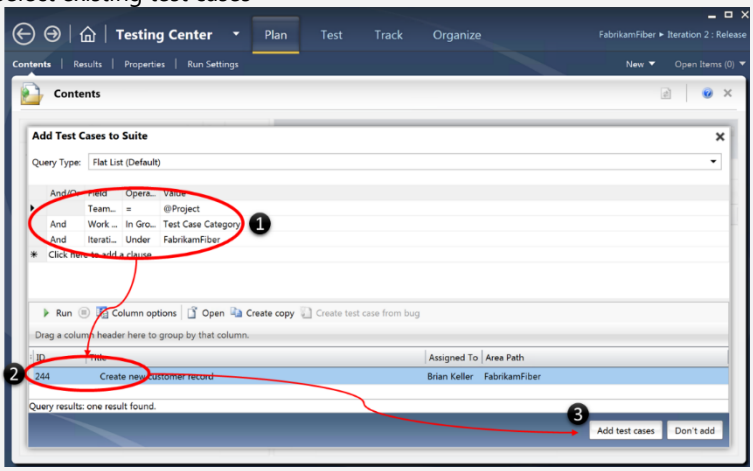
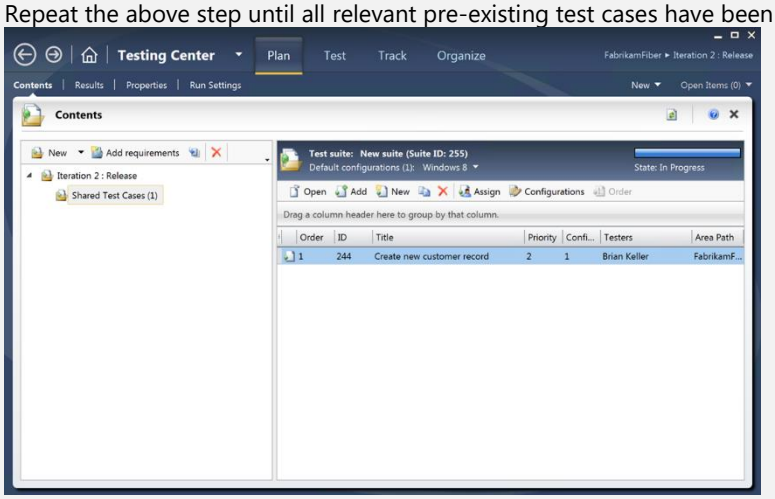
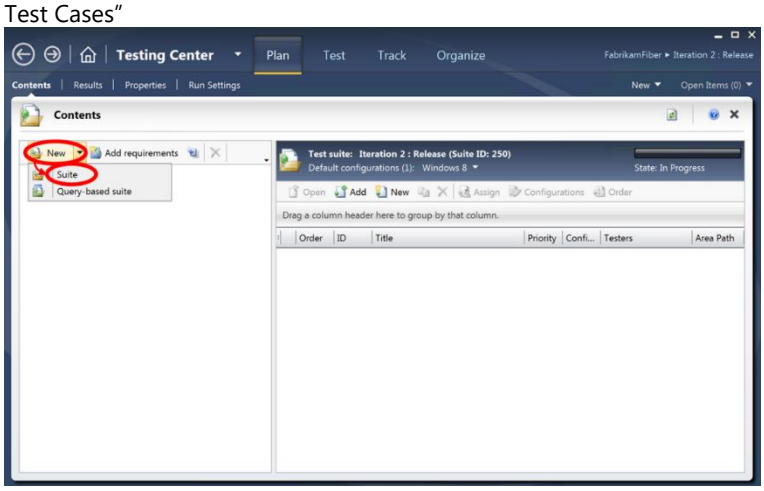
Currently in Visual Studio Online, and in a future release of TFS on premise the next step is to assign multiple testers to the test cases using a form as depicted below:



Create a new test plan for the release branch using MTM

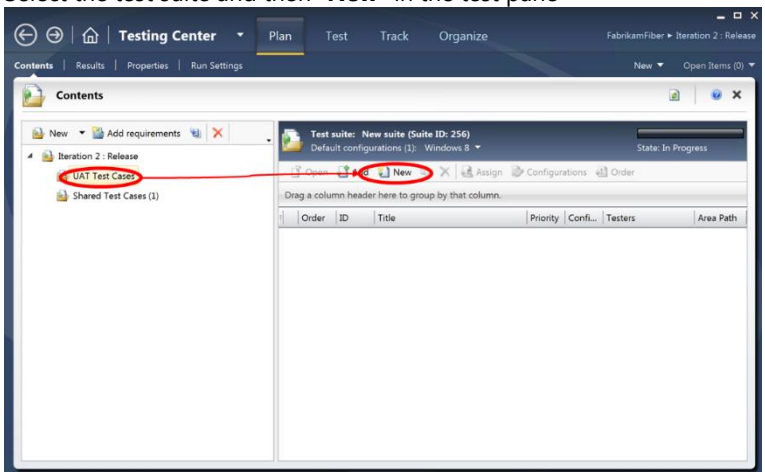
Step	Instructions
1 Create Test Plan [ ] - Done	• Launch **Microsoft Test Manager** and click on the connection dialog if it is not already open  • Select the "**FabrikamFiber**" team project and select "**Connect Now**" • Click the **Add** (Add) to add a new Test Plan • Enter your test plan name and select **Iteration 2** from the iteration list  **GEM** Include the branch name and sprint number in the plan's name • Select "**Add**"

Step	Instructions
	• Select the test plan that was just created and then "**Select Plan**" 
2 Assign Shared Test Cases ⬚ - Done	• In the contents pane on the Plan view, select the test plan that was just created and then select "**New**" and then "**Suite**" and name the test suite "Shared Test Cases"  • Right click on the Test Suite and select "**Add test cases**" 

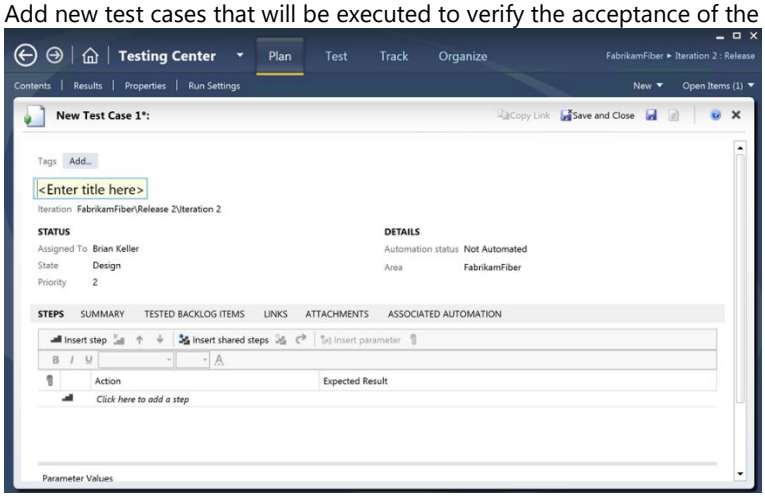
Step	Instructions
	• Select existing test cases  ○ Specify the search criteria to find existing test cases ○ Select all relevant test cases ○ Select "**Add test cases**" • Repeat the above step until all relevant pre-existing test cases have been added to the test suite 
3 Create new UAT Test Cases ☐ - Done	• Create a new suite by selecting the test plan and clicking "**New**" and then "**Suite**" and name it "UAT Test Cases" 

Step	Instructions

- Select the test suite and then "**New**" in the test pane



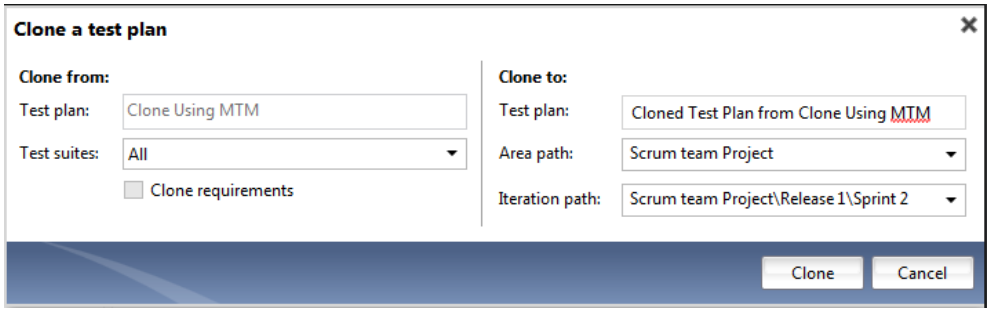
- Add new test cases that will be executed to verify the acceptance of the deployment



Clone Test Plan

This walkthrough will show you how to clone a test plan using MTM and which information is part of a clone action when you go that path.

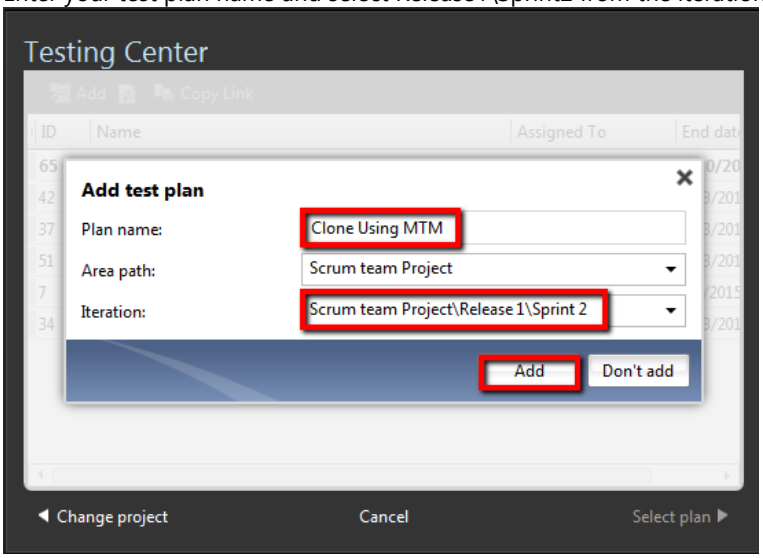
Before we begin

Step	Instructions
1 Test Plan(s) [ ] - Done	Before you can clone a Test Plan using MTM, a (source) Test Plan MUST exist first in MTM. • If they don't exist, create a test plan(s) using MTM and place them into your backlog • Create some test suites and or test cases under a test plan and Move a few of them into an active iteration / sprint NOTE In this example, we are going to use **Release1\Sprint 2** from the **Scrum Team Project** backlog 

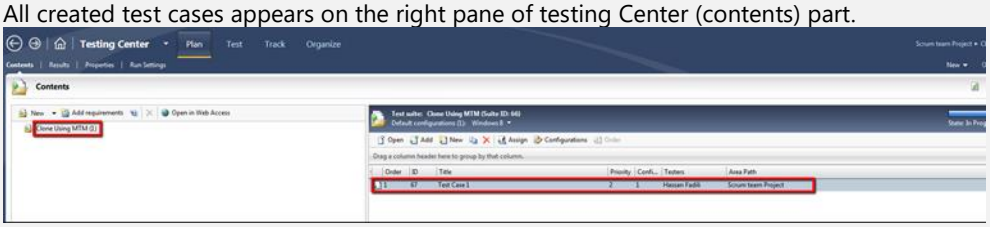
Creating a Test Plan in MTM

Step	Instructions
1 Create Test Plan [ ] - Done	• Start MTM and connect to your TFS instance • Select the desired Team Project Collection • Make sure you are selected the team project of your choice **(Scrum Team Project)** in this case. • Click **Connect now** • Click on the **Add** link in Testing Center, **Add test plan** screen appears

Step	Instructions
	• Enter your test plan name and select Release1\Sprint2 from the iteration list



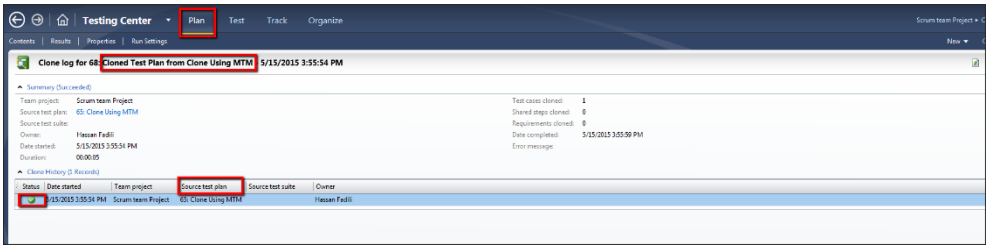
2 Test Cases ⛶ - Done	• Select the added Test Plan (Clone Using MTM) and click icon on the right part of Testing Centre screen to add a test case(s). • Repeat this step in case of need for more creating more test cases under the same test plan. • All created test cases appears on the right pane of testing Center (contents) part.

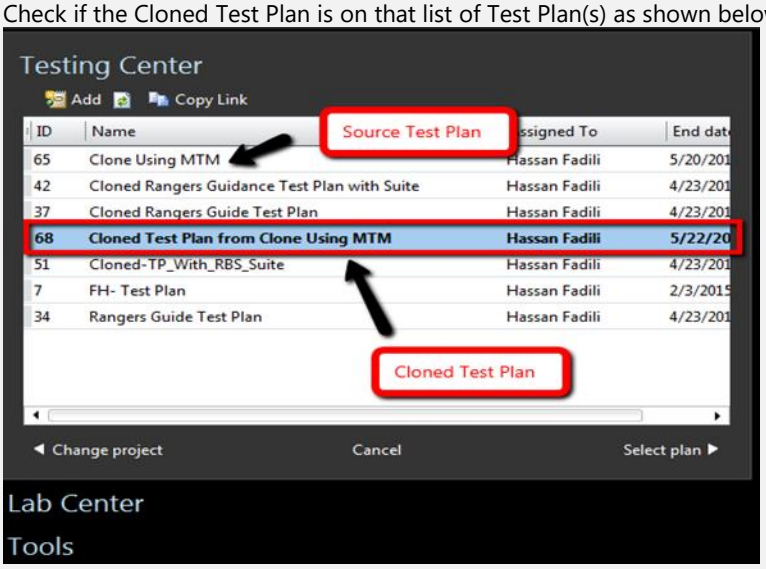
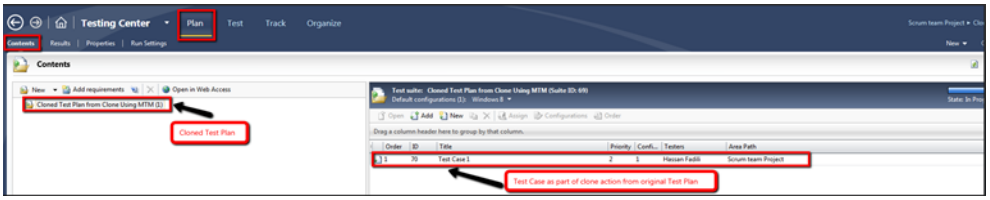


NOTE Many test cases as needed can be added the same way under the same Test Plan. The iteration path of these test cases is automatically set to the same iteration path as configured for test plan. In this case (**Release1\Sprint2**).

Cloning a Test Plan using MTM

Step	Instructions
1 Clone Test Plan ⛶ - Done	• Select the Test Plan created in the previous topic (Create a Test Plan in MTM) • Right click on this test plan, the test plan options appears • Click Clone plan option • Clone a test plan screen appears and enter the desired information by Clone to part (Test plan, Area path and Iteration path. • Click Clone • The clone results are shown below:



Step	Instructions
2 Cloned Test Plan ⛶ - Done	• Select the  icon at the top of Testing Center tool to go back to the overview of Test Plan(s) under the same Team Project • Check if the Cloned Test Plan is on that list of Test Plan(s) as shown below 
3 Check Cloned Test Plan ⛶ - Done	• Select the Cloned Test Plan as shown above and click **Select plan** • Cloned Test Plan is being active in Testing Center including all Test Suite(s) and/or Test Case(s) if any as shown below: 

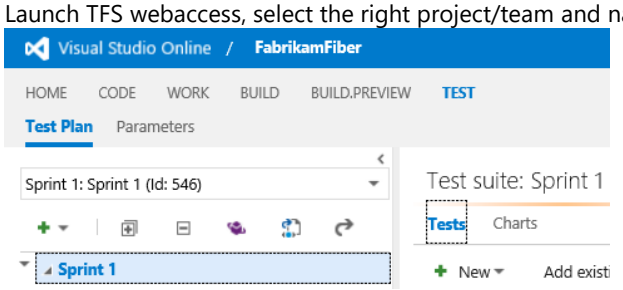
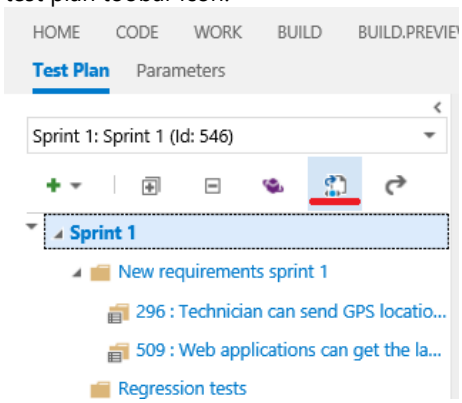
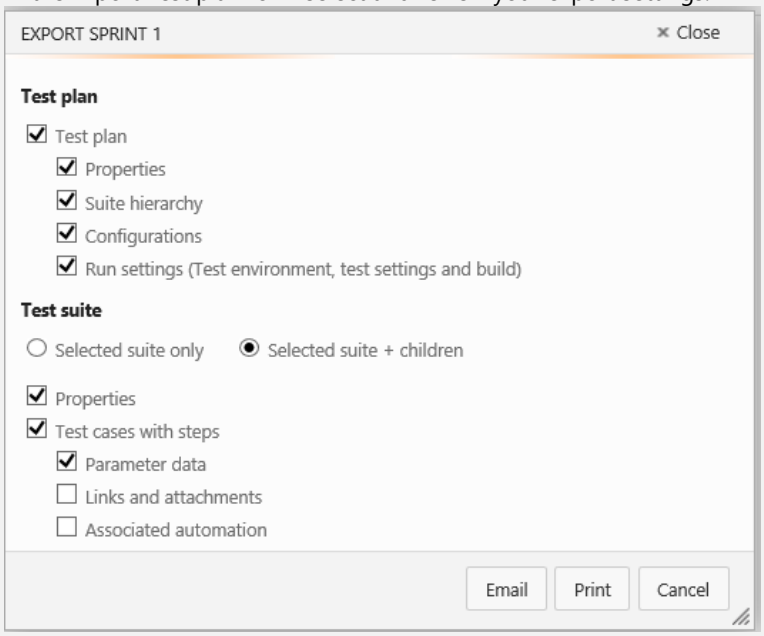
NOTE When a Clone of Test Plan using MTM is done, the Cloned Test Plan and the associated Test Suite(s) and/or Test Case(s) gets new ID's than the original once under the original Test Plan.

Tips and Tricks

- If you clone a test plan using MTM, be aware of the cloned items that those gets new id's.
- To use the cloned Test Plan, make sure you are selected this from the Test plan list and open it in MTM.
- Check if all desired Test Suite(s) / Test Case(s) are copied as part of Clone action of Test Plan.

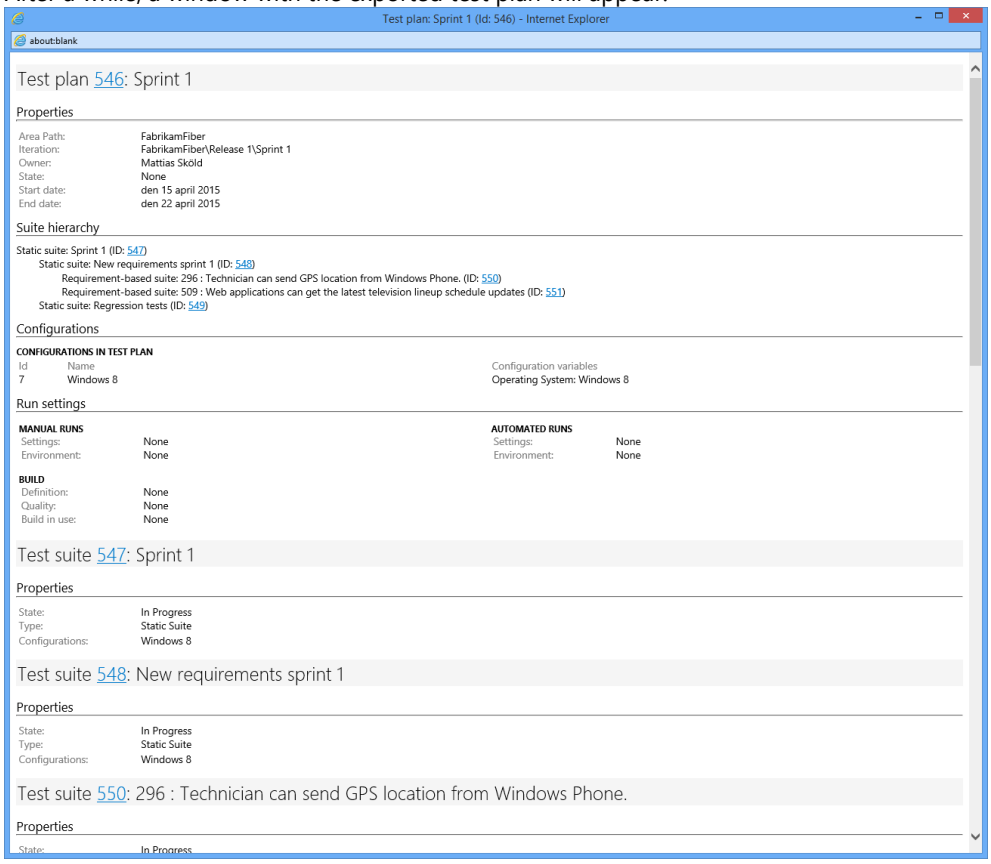
Export a Test Plan

This walkthrough will show you how to export, print or email a test plan or a part of a test plan

Step	Instructions
1 Select Export Test Plan ⛶ - Done	• Launch TFS webaccess, select the right project/team and navigate to the Test Hub  • Select the Test plan root suite, or any part of the plan you want to export, and click the Email or Print test plan toobar icon. 
2 Select what to export ⛶ - Done	• In the Export Test plan form select and review your export settings. 
Export ⛶ - Done	• If you want to Email or Print the test plan, Select the Email or Print buttons. • If you want to export the test plan to other programs, like MS Word, select the Print button.

Step	Instructions

- After a while, a window with the exported test plan will appear.

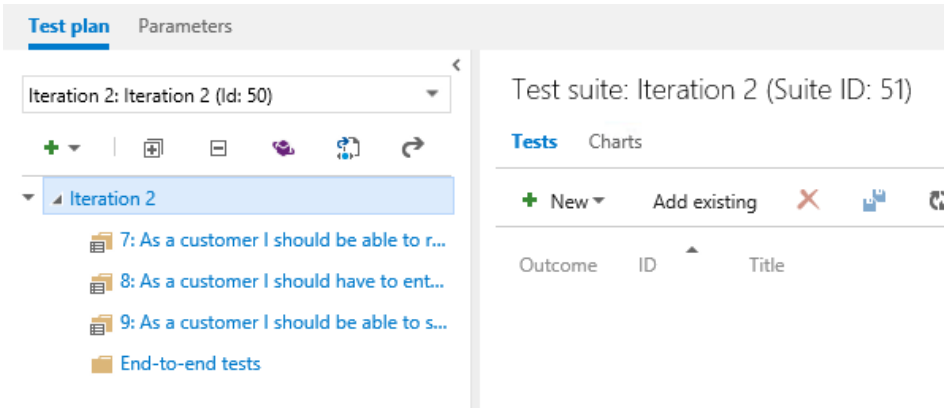


- To export the data to word, click Cancel in the print dialog and right click on the Exported test plan windows,
- Select “Select All”, then “Copy”
- Start word and select Paste

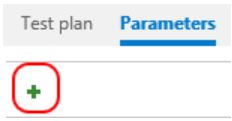
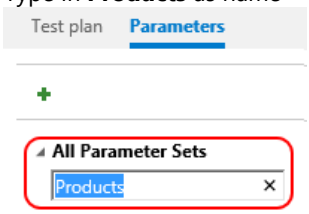
Shared Parameters

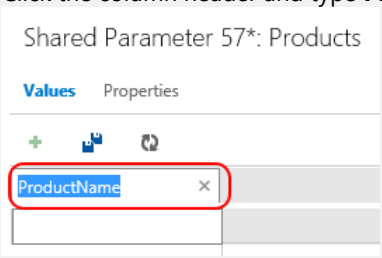
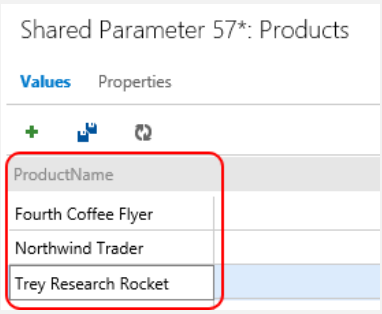
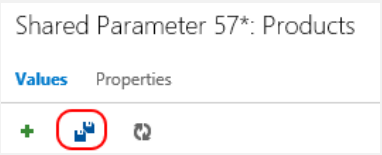
This walkthrough will show you how to create and use shared parameters in a test cases.

Before we begin

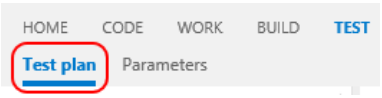
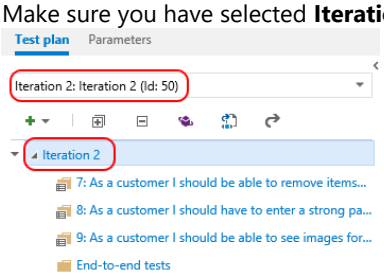
Step	Instructions
1 Test Plan ☐ - Done	Before you can create and use shared parameter, you need to make sure you have a test plan. • If you do not have one - create a test plan. NOTE In this example, we are going to use **Iteration 2** test plan from **Tailspin Toys** 

Creating Shared Parameter Sets

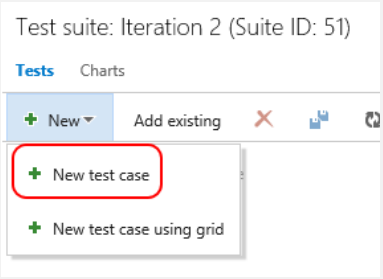
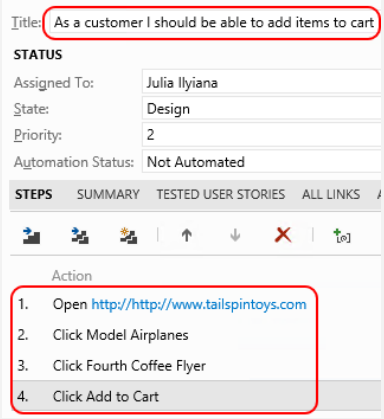
Step	Instructions
1 Create Shared Parameters ☐ - Done	NOTE Shared Parameters can only be created, edited and used from the web access portal – not from Microsoft Test Manager. • Make sure you are in the **Tailspin Toys** Team Project • Click on the **TEST** hub link • Click on the **Parameters** • Click the ✚ link to add a new shared parameter set  • Type in **Products** as name 

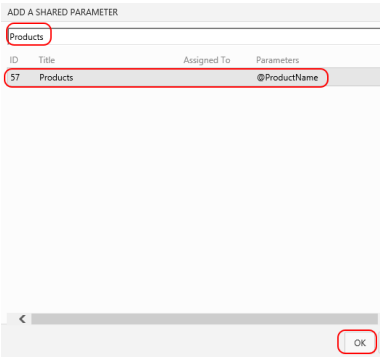
Step	Instructions
2 Add Parameter and Values ⛶ - Done	• Click the column header and type **ProductName**  • Click the value grid in the **ProductName** column • Input these three value in separate cells: **Fourth Coffee Flyer**, **Northwind Trader** and **Trey Research Rocket**  • Click **Save Parameter Set** 

Using Shared Parameters in Test Cases

Step	Instructions
1 Select Test Plan ⛶ - Done	• Make sure you are in the **Tailspin Toys** Team Project • Click on the **TEST** hub link • Click on the **Test Plan** link  • Make sure you have selected **Iteration 2** in the Test plan dropdown 

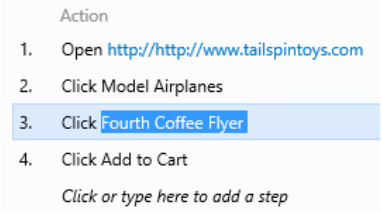
Step	Instructions

2
Create Test Case and Steps
⬚ - Done
	• Select  in the **New** dropdown
	 
	• Add **As a customer I should be able to add items to cart** as Title
	• Add four steps:
	  ○ **Open http://http://www.tailspintoys.com**
	  ○ **Click Model Airplanes**
	  ○ **Click Fourth Coffee Flyer**
	  ○ **Click Add to Cart**
	 

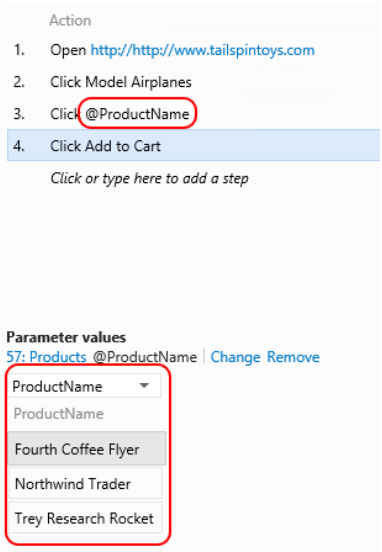
3 Use Shared Parameters ⬚ - Done	• Click **Add a shared parameter set**  • Search for **Products** • Select the line with Title **Products** and click **OK**  • Once Shared Parameters are created and referenced, you can use them in the test cases using the same **@parameterName** syntax that applies to test case parameter values

Step	Instructions

- Select the text **Fourth Coffee Flyer** in step 3



- Type **@ProductName** click enter and notice the parameter value are now loaded from the Shared Parameter Set



- Click **Save**
- Set the State to **Ready**

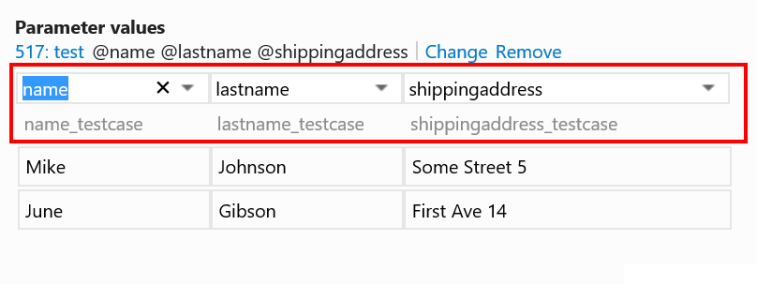
- Click **Save and close**

Tips and Tricks

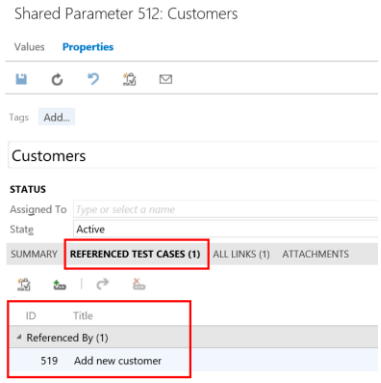
- When cloning work items for e.g. a new release Shared Parameters **cannot be cloned**. Alternatively you can copy the using the grid copy function. To create a copy of an existing parameter set do the following:
  - Create a new shared parameter set
  - Add the parameter names (column headers) matching the exiting data. Optionally you can change the names.
  - Copy the data in the grid of the original shared parameter set.
  - Paste the data into the grid of the new shared parameter set.
- For more advanced copy and editing paste the data into **Excel** before pasting into a shared parameter set.
- If you have existing test cases with parameter values you can use the **Convert to shared parameters** link:

  to convert the value from that test case in a shared parameter set for use in other test cases.
- Often the local parameter names in the individual test cases will not use the same name as the Shared Parameter name. This is especially the case if existing test cases and being updated. Therefore, it is possible to map the Shared Parameter name to

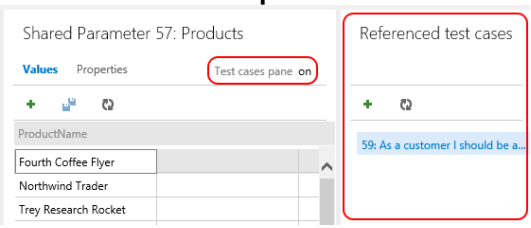
the test case parameter value name:



- To view referenced test cases open the Shared Parameter Set and go to the **Properties** tab:



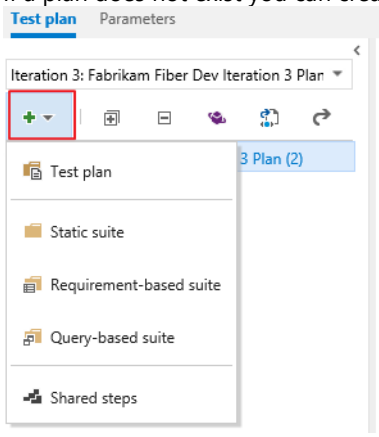
- Or enable the **Test case pane**



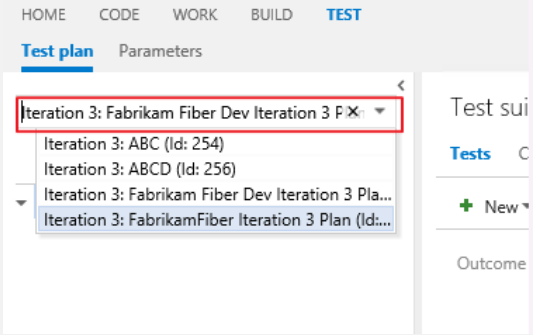
Viewing Test Charts in Team Web Access Lightweight

This walkthrough will show you how to use the new Charts dashboard in Team Web Access.

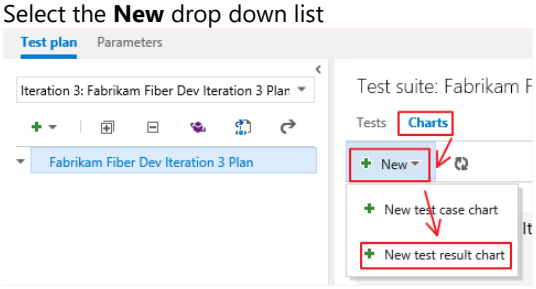
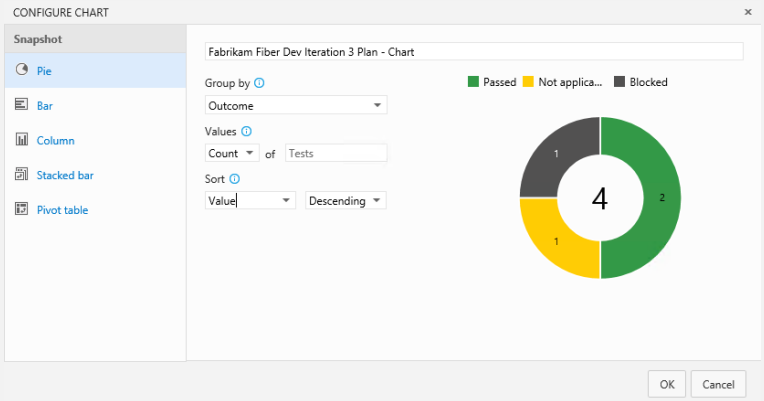
Before we begin

Step	Instructions
1 Test Plan(s) ⬚ - Done	• Before you can effectively explore the chart feature make sure you have a Test Plan that includes several test cases and ideally contains some test run results. • If a plan does not exist you can create one in the test hub by clicking the ✚ **New** dropdown list. 

Navigate to Test Hub

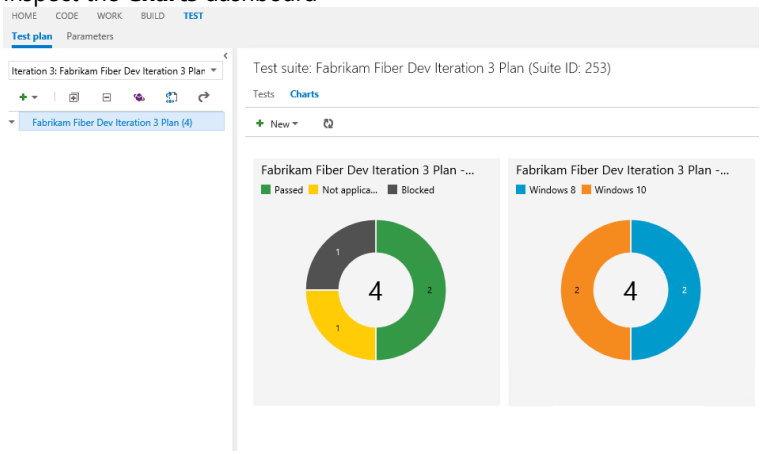
Step	Instructions
1 Navigate to Test Hub ⬚ - Done	• Launch Team web access • Browse to your teams home page • Select the **Test** link
2 Select Test Plan ⬚ - Done	• Select the Test Plan from the drop down list if one is not already selected. 

Configure a Chart

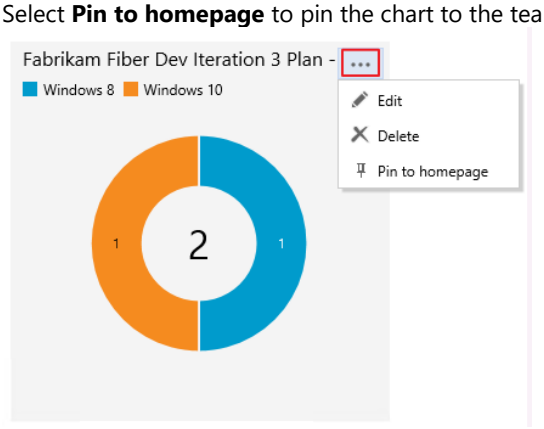
Step	Instructions
1 Navigate To Chart Configuration ⬚ - Done	• Select the **Charts** tab on the right hand pane • Select the **New** drop down list  NOTE Test Case charts are used to display tings about the state of the test **cases** whereas the test result charts are primarily concerned with the **results** of test runs
2 Configure the chart ⬚ - Done	• Select the chart type on the left • Give the chart a title (top textbox on the right hand side) • Select the **Group by** field on the right • Select the **Values** • Select the **Sort** Attribute • Select the sort direction • Notice the chart preview on the far right • Click **OK** 
3 Inspect and edit the chart dashboard ⬚ - Done	• Add a couple more charts

Step	Instructions

- Inspect the **Charts** dashboard



- To Edit the chart click the ellipsis button in the upper right hand corner of the chart
- Select **Edit** to modify the look or style of the chart
- Select **Delete** to remove the chart from the dashboard
- Select **Pin to homepage** to pin the chart to the team home page



NOTE Test charts that are pinned to home page are also accessible to Stakeholders even though they don't have access to the TEST hub

Appendix

References

Information	URL
MSDN Microsoft Test Manager Guide	http://msdn.microsoft.com/en-us/library/ms182409.aspx
Copying and Cloning Test Suites and Test Cases	http://msdn.microsoft.com/en-us/library/vstudio/hh543843.aspx
Visual Studio Test Tooling Guidance	http://vsartesttoolingguide.codeplex.com
Test Case Migrator Tool by Shai Raiten (does handle shared steps)	http://blogs.microsoft.co.il/blogs/shair/archive/2011/03/20/test-case-migrator-between-projects-wpf-metro.aspx
Test Scribe Reporting Tool	http://visualstudiogallery.msdn.microsoft.com/e79e4a0f-f670-47c2-9b8a-3b6f664bf4ae
Test Scribe Extended	http://testscribeextended.codeplex.com
SQL Server Reporting Services	http://msdn.microsoft.com/en-us/library/ms159106.aspx
Microsoft Test Case Management Reporting FAQs	http://blogs.msdn.com/b/vstsqualitytools/archive/2011/10/14/test-case-management-tcm-reporting-frequently-asked-questions-part-1.aspx http://blogs.msdn.com/b/vstsqualitytools/archive/2011/11/11/test-case-management-tcm-reporting-frequently-asked-questions-part-2.aspx
Community TFS Report Extensions	http://communitytfsreports.codeplex.com
Identifying Code Change Impact on Tests	http://msdn.microsoft.com/en-us/library/dd264992.aspx
Repeat a test with different data (shared parameters)	https://msdn.microsoft.com/library/dd997832.aspx

In Conclusion

This concludes our journey with the goal to equip software testers with a firm understanding of the principles and practices that will enable them to develop better solutions to the testing challenges of their particular projects.

We hope you have found this guide useful and wish you success in your software adventures.

Sincerely