

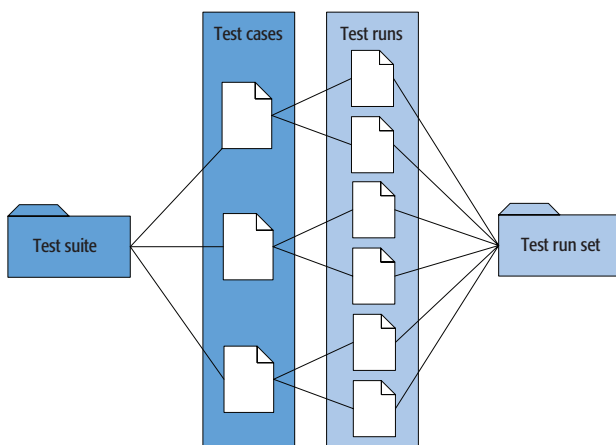
# TestTrack TCM Best Practices

This document addresses some of the common activities in TestTrack TCM and offers best practices for each. These best practices are designed with TestTrack TCM users in mind, but many of them may apply regardless of the tool. These best practices are guidelines, and your testing process and business policies should dictate whether you follow a practice or not.

## Test components

Before you begin using TestTrack TCM, it is important to understand the differences between test cases and test runs and how they are organized.

Think back to the manual tracking process you used before TestTrack TCM. You may have created a checklist of items to test. When a new product version required testing, you printed the checklist and wrote the test results on it. The items you used to create and perform the test are all represented in TestTrack TCM: the electronic version of the checklist is a test suite, each line in the electronic checklist is a test case, the printed checklist is a test run set, and each item in the printed checklist is a test run. The following diagram shows the relationship between these components.



Relationship between test components

## Test cases versus test runs

Test cases are the core test component in TestTrack TCM. Test cases contain all information about a test, including the description, scope, conditions, detailed steps, expected results, scripts, and other test data.

Test runs are instances of test cases that are generated at a milestone in the testing cycle, such as when a build is provided by the development group. Test runs are assigned to testers to perform the test and then the test results are entered in the test run. A test run contains all information from the related test case, but also includes the results of a specific instance of the test. A single test case can have one or more related test runs, which are performed against different builds or configurations.

Keep the following in mind:

- Test cases are reused for future testing efforts. Test runs are only used once during the testing cycle they are generated for.
- Test runs are modified to indicate the test results. Test cases generally remain static unless they need to be modified because of an application change or incorrect information.

## Test suites

A test suite is a group of related test cases. In TestTrack TCM, folders or custom fields are used to represent test suites. Test cases can be grouped based on the testing cycle, test purpose, functional area, or other criteria. A single test case may be part of one or more test suites.

## Test run sets

A test run set is a group of related test runs. When you generate test runs, you can add the new test runs to a test run set. When testing multiple builds in parallel, test run sets allow you to report on each test initiative and determine the current status.

## Preparing for testing

### Develop a test plan

The backbone of a successful testing effort is a detailed test plan. Write a plan before starting each testing cycle to communicate the testing process to the development and quality assurance

teams. The test plan ensures that everyone understands what will be tested, how it will be tested, when testing will occur, who will perform tests, and the criteria used to determine when testing is complete and the product is ready for release.

The test plan should also identify the team members responsible for performing various tasks in TestTrack TCM. Document who will write test cases, review test cases, and generate test runs. Also provide details about when these activities should occur during the testing cycle to eliminate duplicate effort and confusion about responsibilities.

The test plan can also be used as a guide as you configure TestTrack TCM before testing begins. For example, the build schedule can help you determine the test run sets that need to be configured.

### Identify test suites

As an application evolves, the number of test cases increases. Develop a list of test suites to group related test cases and keep them organized.

To identify a test suite, consider the groups of tests that will be run during the entire testing cycle. If possible, the same suite names should be used for each major product release. At a minimum, use a suite for all regression tests and a suite for all new product feature tests. Following are examples of test suites that can be used to group test cases.

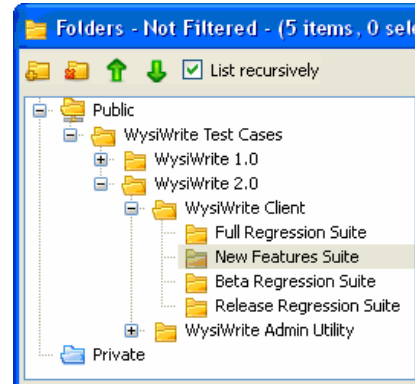
- **New Features Suite**—Includes all new test cases written for new features introduced in the product version. If you are testing the first version of a product, all test cases are part of this suite.
- **Full Regression Suite**—Includes all test cases used to test previous product versions.
- **Beta Regression Suite**—Contains a subset of test cases from the Full Regression Suite. These tests help ensure that application changes do not break existing functionality between each build.
- **Release Regression Suite**—Contains a smaller subset of test cases from the Full Regression Suite. These tests are only performed on the release candidate build.

### Create folders or custom fields to represent test suites

After you identify test suites, you can create folders or custom fields in TestTrack TCM for each suite. Review your business rules and testing process to determine how to best structure the test suites.

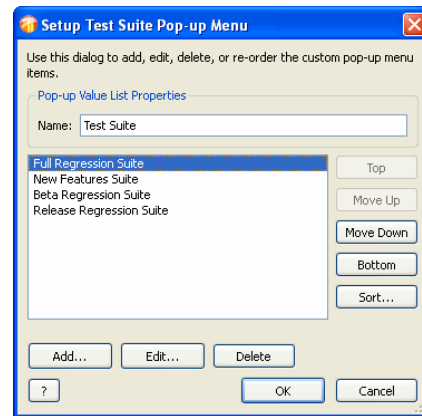
If you decide to use folders, a common strategy is to create a folder for each product version and then create a sub-folder in the product version folder for each application bundled with the product.

Finally, create a folder to represent each test suite you identified. The following screenshot shows the folders used to organize test cases for a product and the folder structure for one version of the product.

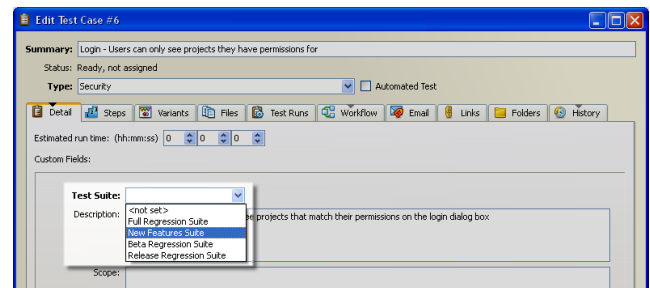


Folders list window

If you decide to use custom fields, a common strategy is to create a custom field that includes each test suite as a value. Custom fields are displayed on the Detail tab in the Edit Test Case and View Test Case windows. Make the field required so testers must add a test case to a suite.



Setup Custom Pop-up Menu dialog box



Edit Test Case window

An advantage of using folders for test suites is that they provide a familiar method to organize test cases. An advantage of using custom fields is that, if they are required, users must select a test

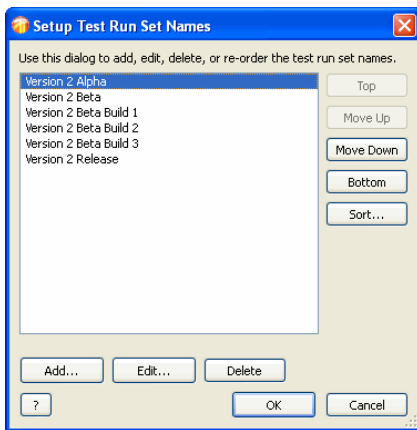
suite when adding or editing a test case. If you use folders, test cases must be manually added to them. You can create filters and generate reports based on both folders and custom fields.

### Configure test run sets to group related test runs

Test run sets are designed to group test runs generated for a testing phase. Test runs can be added to test run sets when they are generated.

While test run sets are similar to folders, the advantage is that the test run set is displayed in the main area of the Edit Test Run and View Test Run windows. Users must click the Folders tab to view the folders a test run is stored in.

When deciding on the test run sets to create, it is important to consider how you want to view information in test run reports to monitor progress. Think about the information you will need at the end of each phase to determine if you are ready to move on to the next phase. Refer to the schedule in the test plan and create a test run set for each milestone. Then create any additional sets that are needed. The following screenshot shows the Setup Test Run Set Names dialog box, which is used to create test run sets.

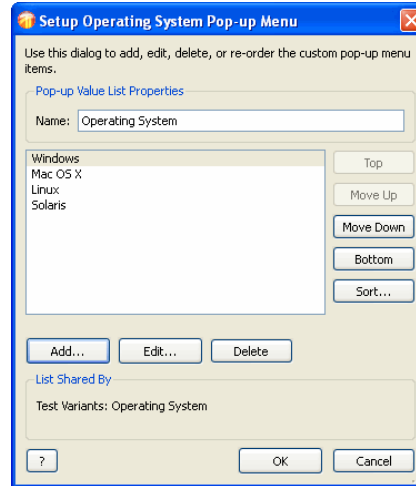


Setup Test Run Set Names dialog box

### Identify test variants

Test variants are attributes of the tested application used to generate test runs. They allow you to create multiple test runs from a single test case to support multiple configurations. For example, if you are testing a cross-platform application, you can create a test variant named Operating System and include each platform as a test variant value. A test run is created for each unique combination of test variant values that are selected when test runs are generated.

Companies often find that it takes too long and is too expensive to test every combination of every variable in an application. Consider



Setup Test Variant Pop-up Menu dialog box

using all-pairs testing to achieve an acceptable level of quality without testing every possible combination. For more information about all-pairs testing, visit [www.seapine.com/allpairs.html](http://www.seapine.com/allpairs.html).

### Develop test cases for new features in parallel with product development

Write new feature test cases at the same time as the developers are implementing the features. This helps prepare the quality assurance team to begin testing as soon as the development group provides the first Alpha build.

After a test case is written, it should be reviewed by a quality assurance team member, revised if changes are required, and then marked as ready so test runs can be generated for it. Also, make sure new test cases are added to the appropriate test suite.

### Generating and organizing test runs

#### Generate test runs when a new product build is available

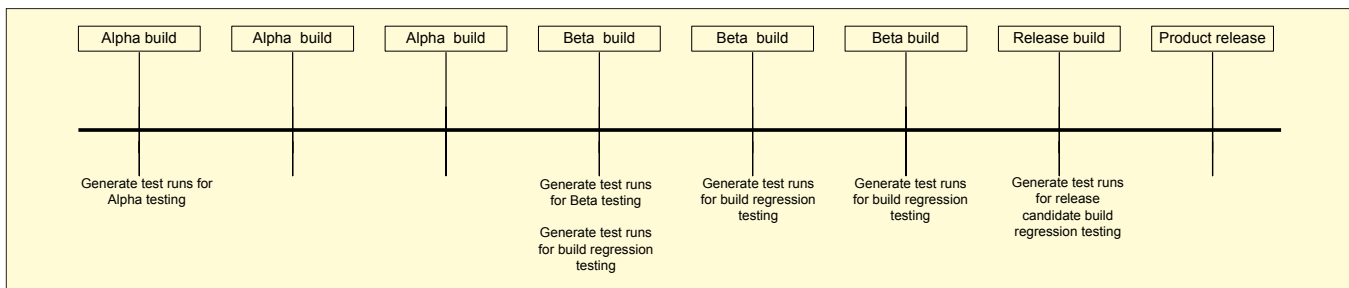
Generate test runs when test cases are complete and the product is ready for testing. This typically occurs when the development group provides a new build.

It is best to generate test runs when each testing phase begins. This helps you more easily track the test cases that test runs have not been created for and evaluate how many test runs still need to be performed so you can determine the progress of the overall testing cycle.

You can assign all the test runs when they are generated or assign them periodically throughout the testing cycle. Assigning test runs when each testing phase begins can help testers better manage their workload because they have a clear indication of the amount of testing they need to complete. It also allows the person assigning

work to compare the number of test runs assigned to each tester and make adjustments as needed. Assigning test runs periodically throughout the testing cycle provides the flexibility to assign tasks as resources become available if you do not know how long it will take to perform tests or how many testers are available during each testing phase.

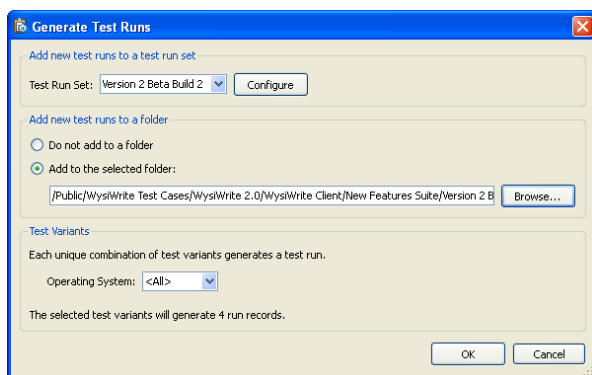
The following diagram shows an example of when to generate test runs during the testing cycle.



### Add test runs to test run sets at generation time

Add test runs to a test run set when they are generated. This ensures that test runs are properly categorized so they are included in filters and reports with the related test runs. The test run sets should generally be configured before test runs are generated. However, if an additional build requires a new test run set, you can configure it when you generate test runs.

In addition to using test run sets, optionally add test runs to folders to organize them in the test suite with related test cases. The following screenshot shows the Generate Test Runs dialog box, which is used to generate and organize new test runs.



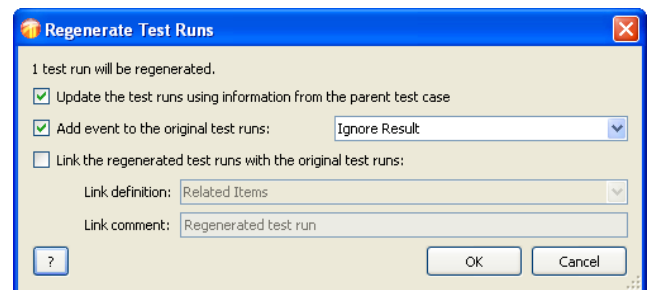
Generate Test Runs dialog box

### Rerunning failed tests

When a test run fails, it moves to the Failed workflow state. Assuming your company's business rules state that releasing a product with failed test runs is unacceptable, you need to rerun a

failed test after the development team fixes the defect that caused the test to fail.

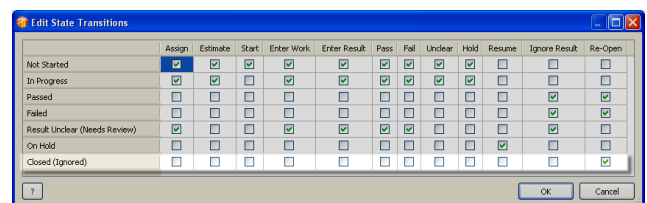
In this situation, regenerate the failed test run instead of reopening it and modifying the results. It is important to keep the failed test run for historical purposes to show that the test initially failed. The new test run will provide verification that the test eventually passed before the product was released. The following screenshot shows the Regenerate Test Runs dialog box, which is used to set options for regenerated test runs.



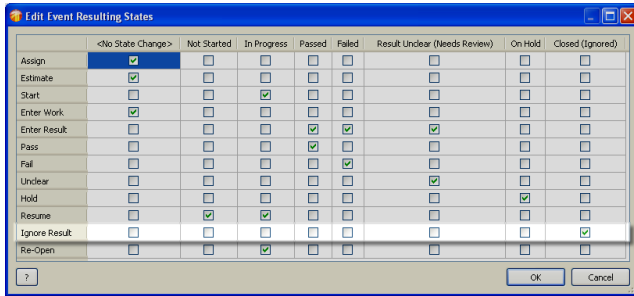
Regenerate Test Runs dialog box

Make sure you move the original, failed test runs to a state that differentiates them from test runs that have not been regenerated. This allows you to maintain an accurate history of test run failures and easily locate failed tests that have not been retested. The default test runs workflow includes the 'Ignore Result' event and 'Closed (Ignored)' state, which are designed for this purpose. You may need to add the state and event if your workflow does not include them.

For the 'Closed (Ignored)' state, select 'Closed' as the attribute and 'Failed' as the test run result. Make sure the 'Re-Open' event can be entered for the new state and the resulting state for the new event is 'Closed (Ignored)'. The following screenshots show the workflow transitions and resulting states.



Edit State Transitions dialog box



Edit Event Resulting States dialog box

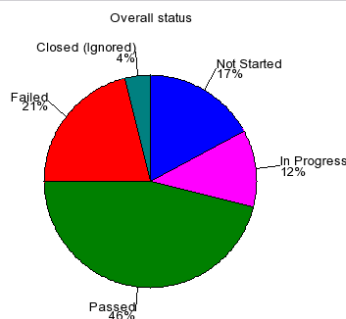
## Evaluating testing progress

### Generate reports based on test run sets

Measuring the progress of your testing effort is critical for determining if testing is on schedule and how much effort is required to complete a testing phase. Use distribution reports based on test run sets to view a snapshot of test runs at a specific point in the testing cycle. The following screenshot shows a sample distribution report that includes the status of test runs in each test run set. This report can help you determine how many test runs are waiting, in progress, on hold, or completed for each test run set. Completed test runs are displayed according to the test run result.

### Test run status by test run set

Report Period: from first test run found through last test run found					
	Version 2 Alpha	Version 2 Beta	Version 2 Beta Build 1	Version 2 Beta Build 2	Totals
Not Started	0	0	0	13	13
In Progress	0	0	6	3	9
Passed	16	13	6	0	35
Failed	4	8	4	0	16
Closed (Ignored)	0	3	0	0	3
<b>Totals</b>	<b>20</b>	<b>24</b>	<b>16</b>	<b>16</b>	



Distribution report—test run status by test run set

The following screenshot shows another sample distribution report that includes the test run results for each test run set. This report can help you determine how many test runs in each test run set are Passed, Failed, or Underdetermined. It includes the percentage of passed test runs and defects created for test runs in each test run set.

### Test run result status by test run set

Report Period: from first test run found through last test run found								
	Total Runs	Passed	Failed	Undetermined	%Passed	Total Defects	Open Defects	Totals
Version 2 Alpha	24	20	4	0	83.33%	0	0	24
Version 2 Beta	24	8	1	0	33.33%	0	0	24
Version 2 Beta Build 1	12	0	4	0	0%	1	1	12
Version 2 Beta Build 2	12	0	0	0	0%	0	0	12
Version 2 Beta Build 3	12	0	0	0	0%	0	0	12
Version 2 Release	12	0	0	0	0%	0	0	12
<b>Totals</b>	<b>96</b>	<b>28</b>	<b>9</b>	<b>0</b>	<b>29.17%</b>	<b>1</b>	<b>1</b>	

Distribution report—test run result status by test run set

Over time, the number of test runs sets will increase. Use filters when you generate reports to limit the test run sets included based on the information you want to evaluate.

### When to generate reports

Although you will periodically generate and evaluate reports to determine the progress of your testing effort, it is important to assess how much testing remains as you near the end of a testing phase. For example, as you approach the end of the Alpha testing phase, generate a distribution report of test run status based on the Alpha test run set to see how much testing remains. When you are halfway through the Beta testing phase, generate the report to determine how many test runs in the Beta test run set are complete. This can indicate if testing is behind, on, or ahead of schedule. Before the product release, review failed test runs and inform development of any roadblocks that could delay the product release.

### Exclude 'Closed (Ignored)' test runs from reports

If you want to report on the latest test run results in a test run set, create a filter that excludes test runs in the 'Closed (Ignored)' workflow state. If you do not exclude this state, the report may not display the desired information because the same test case will be included multiple times if it was run more than once.

### Preparing for the next release

#### Move new feature test cases to the regression suite

After a product is released and before you start testing the next version, move the new feature test cases to the regression suite. This helps you build a comprehensive regression suite of tests to use for testing future product releases.

#### Analyze reports to determine how testing went

After the release, generate and analyze test run reports to see how well testing went. Use this information to improve future testing efforts.

Compare the test run Estimated Run Time to the Actual Run Time. This can help you determine if the estimates were accurate and help with planning.

Evaluate the number of test runs in the 'Closed (Ignored)' state. This can help you determine the number of tests that passed the first time they were run. A high number of regenerated test runs may indicate that the development team should review their unit testing and code review processes to ensure higher quality on the first pass.

### Testing cycle example

The following example shows when to generate test runs, how to group them in test run sets, and when the test runs in each test run set should be performed within a typical test cycle. The example uses the following test suites:

- **New Features Suite**—Includes all new test cases written for new features introduced in the product version. These tests are performed once during the entire Alpha testing phase and again during the entire Beta testing phase.
- **Full Regression Suite**—Includes all test cases used to test previous product versions. These tests are performed once during the entire Alpha testing phase and again during the entire Beta testing phase.
- **Beta Regression Suite**—Contains a subset of test cases from the Full Regression Suite. These tests are performed on each build during the Beta testing phase.
- **Release Regression Suite**—Contains a smaller subset of test cases from the Full Regression Suite. These tests are only performed on the release candidate build.

### Alpha 2 build

When the Alpha 2 build is ready for testing, do not generate new test runs. Testers should continue to perform test runs in the *Version 2 Alpha* test run set.

### Alpha 3 build

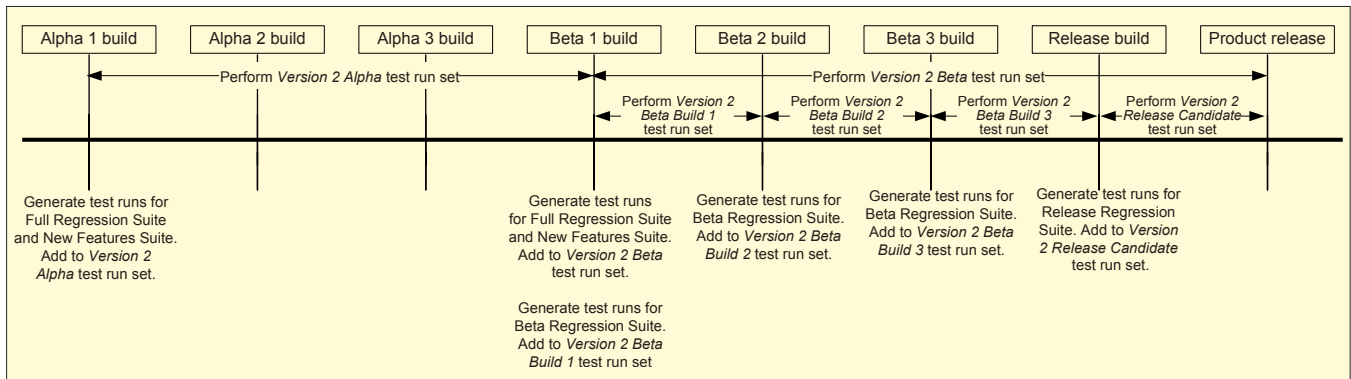
When the Alpha 3 build is ready for testing, do not generate new test runs. Testers should continue to perform test runs in the *Version 2 Alpha* test run set.

To determine how close you are to completing tests before continuing on to the Beta testing phase, generate and evaluate a distribution report based on the status of the *Version 2 Alpha* test run set.

### Beta 1 build

When the Beta 1 build is ready for testing, generate test runs for the test cases in the Full Regression Suite and New Features Suite. Add these test runs to a test run set named *Version 2 Beta*. Testers should perform these tests throughout the entire Beta testing phase, regardless of the Beta build they are performed on.

Also generate test runs for the test cases in the Beta Regression Suite. Add these test runs to a test run set named *Version 2 Beta Build 1*. These test runs should be completed before the Beta 1 build is released for customer testing. To determine when the build is ready to be released, generate and evaluate reports based on the *Version 2 Beta Build 1* test run set.



Testing cycle example

### Alpha 1 build

When the Alpha 1 build is ready for testing, generate test runs for the test cases in the Full Regression Suite and New Features Suite. Add these test runs to a test run set named *Version 2 Alpha*. Testers should perform these test runs throughout the entire Alpha testing phase, regardless of the Alpha build they are performed on.

### Beta 2 build

When the Beta 2 build is ready for testing, generate test runs again for the test cases in the Beta Regression Suite. Add the test runs to a test run set named *Version 2 Beta Build 2*. These test runs should be completed before the Beta 2 build is released for customer testing. Testers should also continue to perform test runs in the *Version 2 Beta* test run set.

Use reports to evaluate progress on the regression and new features tests in the *Version 2 Beta* test run set. Determine if the number of test runs completed is on track with the duration passed in the schedule.

To determine when the build is ready to be released, generate and evaluate reports based on the *Version 2 Beta Build 2* test run set.

### **Beta 3 build**

When the Beta 3 build is ready for testing, generate test runs again for the test cases in the Beta Regression Suite. Add the test runs to a test run set named *Version 2 Beta Build 3*. These test runs should be completed before the Beta 3 build is released for customer testing. Testers should also continue to perform test runs in the *Version 2 Beta* test run set.

To determine when the build is ready to be released, generate and evaluate reports based on the *Version 2 Beta Build 3* test run set.

### **Release build**

When the release build is ready for testing, generate the test runs for the test cases in the Release Regression Suite. Add the test runs to a test run set named *Version 2 Release Candidate*. Testers will perform these test runs on the Release build only. These test runs should be complete before the product is released. Testers should also complete the test runs in the *Version 2 Beta* test run set.

To determine when testing is complete and the build is ready to be released, generate and evaluate reports based on the *Version 2 Release Candidate and Version 2 Beta* test run sets.

### **Conclusion**

These best practices form the basis for effective test case management using TestTrack TCM. Implementing them can help you avoid common mistakes and improve your test case management and testing process.