Working with Test Environments

What is a Test Environment

Generically speaking, a test environment is an environment that contains all necessary elements, including the SUT, so you can perform testing on it.

Depending on your context, a test environment may represent:

- a testing stage (e.g. "development", "staging", "preproduction", "production")
- a device model or device operating system (e.g. "Android", "iOS")
- an operating system (e.g. "Windows", "macOS", "Linux")
- browser (e.g. "Edge", "Chrome", "Firefox")

Thus, semantics of what a "Test Environment" represents depends on your specific context.

Please note

In Xray, Test Environments are focused on the execution aspect, providing the means to schedule tests and analyze their results in different environments.

Thus, they're explicitly associated with Test Execution issues.

Benefits of using Test Environments

- avoid duplication of Tests, whenever you have to run the same test on different environments
- ability to track the latest status of tests on different environments
- ability to track coverage on each environment
- ability to track overall coverage, considering the coverage/results on each environment
- ability to perform reporting, including traceability, per each environment or globally (i.e. considering all results on all different environments)

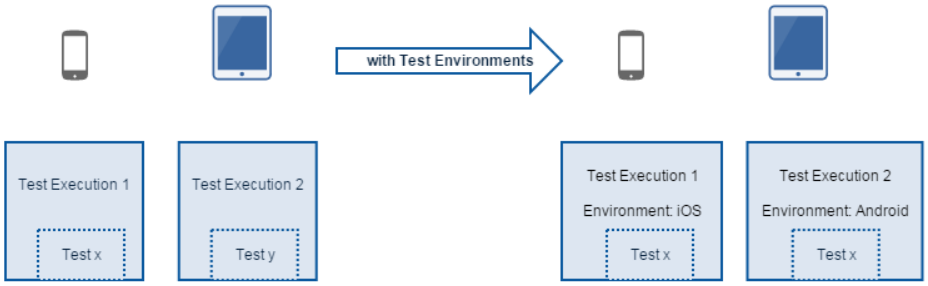
Test Environments in Xray at a glance

The current status of a Test is generally calculated by looking at the Test's last execution . However, this does not work well if you execute the same Test in two different test environments (e.g., devices) and you want the two results to be consolidated.

Within a Test Execution, you may specify the **Test Environment**(s) where the tests will be executed in the respective attribute. A Test Environment is similar to a label, but Xray has special logic to deal with it.

If you use Test Environments, you may reuse the same Test for multiple test environments and create Test Executions for each one. If you don't use Test Environments and you want to track tests for multiple environments (e.g., devices), then the only way to do it is to create multiple tests, one per each test environment.

Let's say that you have executions for two Test Environments: "Android" and "iOS". The test is considered as PASS only if the **latest** executions for Android and iOS are **both** PASS; otherwise, it will be FAIL.



Note: If you don't use Test Environments, then only the latest result matters for assessing the current status of the Test.

How it works

Let's say we have a Test being executed two times. We will start by creating Test Executions TE1 and TE2.

TE2 will be executed after TE1, so TE1 is considered older.

Below are example scenarios and shows how the overall, **consolidated status** of the Test is calculated in each case.

If you have a mix of Test Executions (i.e., with and without Test Environment as in scenario D), it will be treated the same way as scenarios A, B, and C.

Scenario	Test Environment (s) of TE 1	Test Environment (s) of TE 2	Test run status in TE1	Test run status in TE2	The calculated value for the overall, consolidated status of the Test (i.e. for the "All Environments")	Other
A	Android	iOS	PASS	PASS	**PASS**	The test will be considered to be PASS in both Android and iOS environments.
B	iOS	iOS	PASS	FAIL	**FAIL**	The test will be considered to be FAIL in iOS.
C	iOS	iOS	FAIL	PASS	**PASS**	The test will be considered to be PASS in iOS.
D	iOS	-	FAIL	PASS	**FAIL**	The test will be considered to be FAIL in iOS and PASS for the empty environment.
E	-	-	PASS	FAIL	**FAIL**	The test will be considered to be FAIL for the empty environment.
F	-	-	FAIL	PASS	**PASS**	The test will be considered to be PASS for the empty environment.

Please note

The *empty* Test Environment is treated similarly to any other environment having a well-defined name.

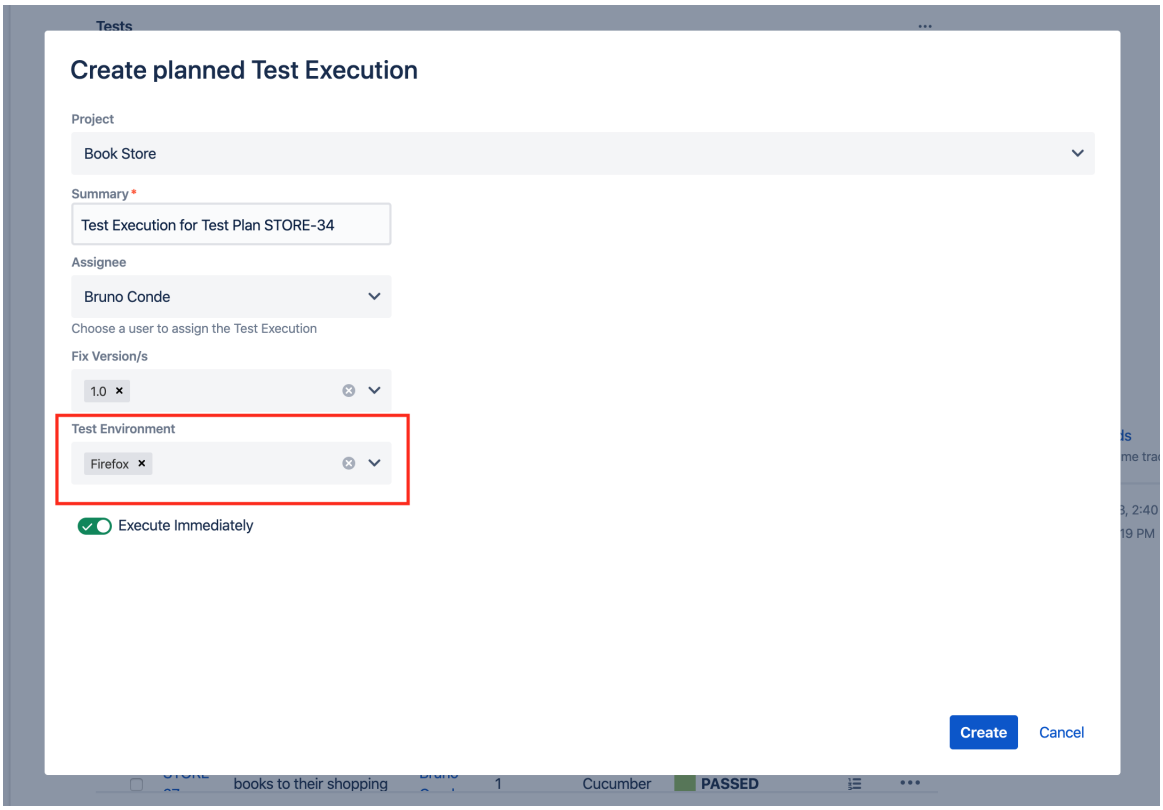
How to use

Whenever creating a Test Execution, you must set the Test Environment in which the execution will be executed. You can use this field as a simple label: just add the environment or reuse a previously created one.

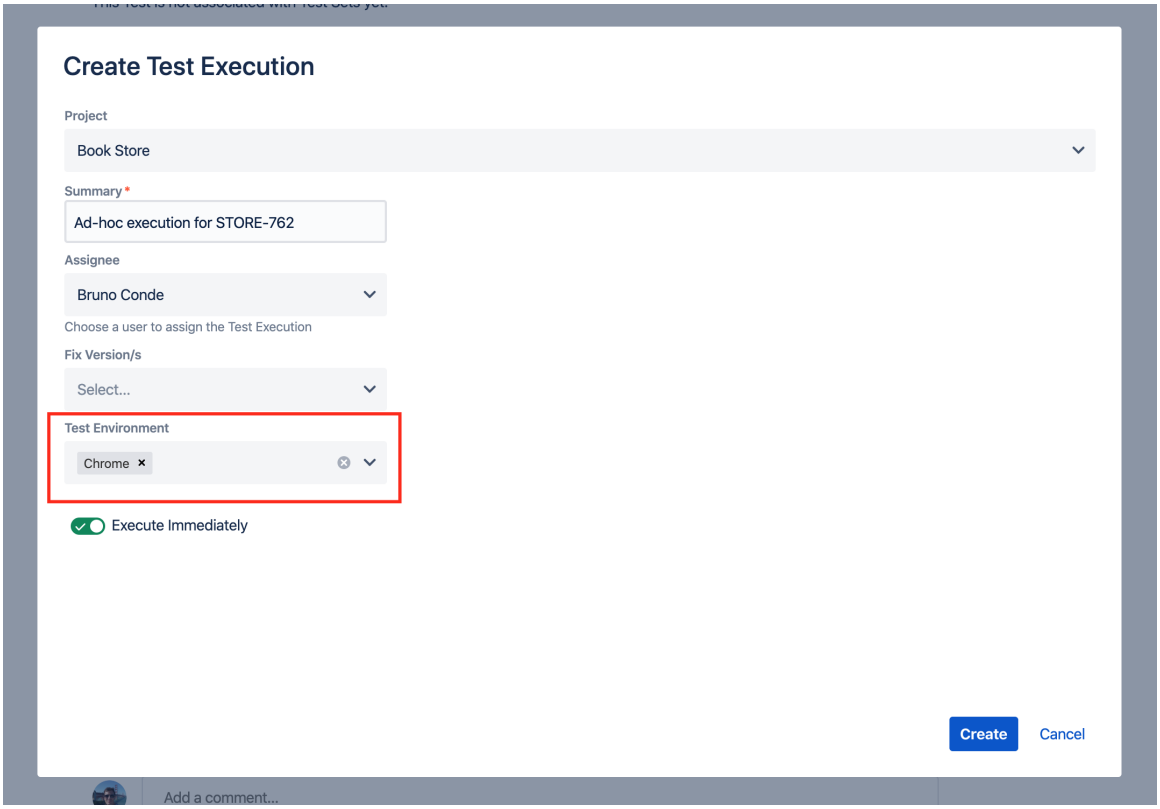
Please see some important Tips and Recommendations ahead.

Creating a Test Execution

You can create a Test Execution issue from the Test Plan and the Test issue using actions provided by Xray. In this context, the Test Environment field will be provided so that users can immediately specify a Test Environment.



Creating a Test Execution from a Test Plan for "Firefox" environment

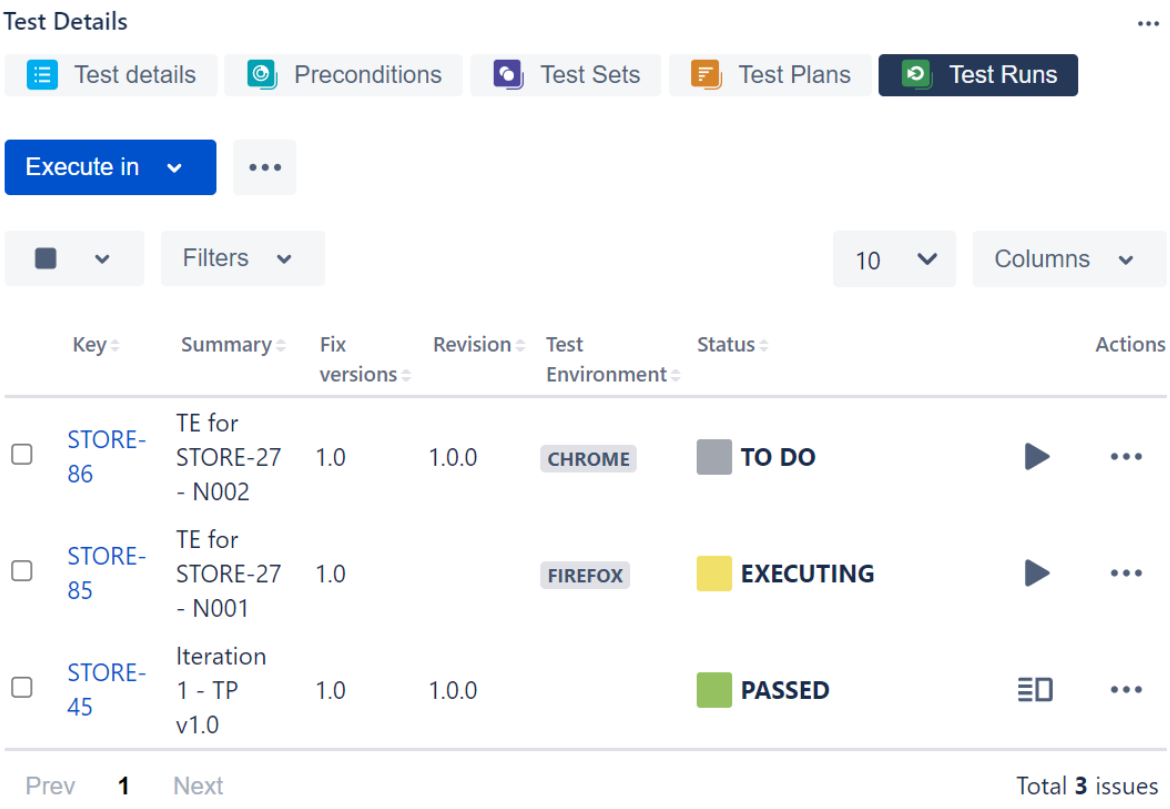


Creating a Test Execution from the Test issue for "Chrome" environment

Tracking the results on different environments

The Test Environments column is shown in your Test Runs table so you can distinguish each execution of the Test between the different environments.

This information can be seen in the Test issue screen (see next screenshot) or in other places that show a list of Test Runs (e.g. Test Plan issue screen).



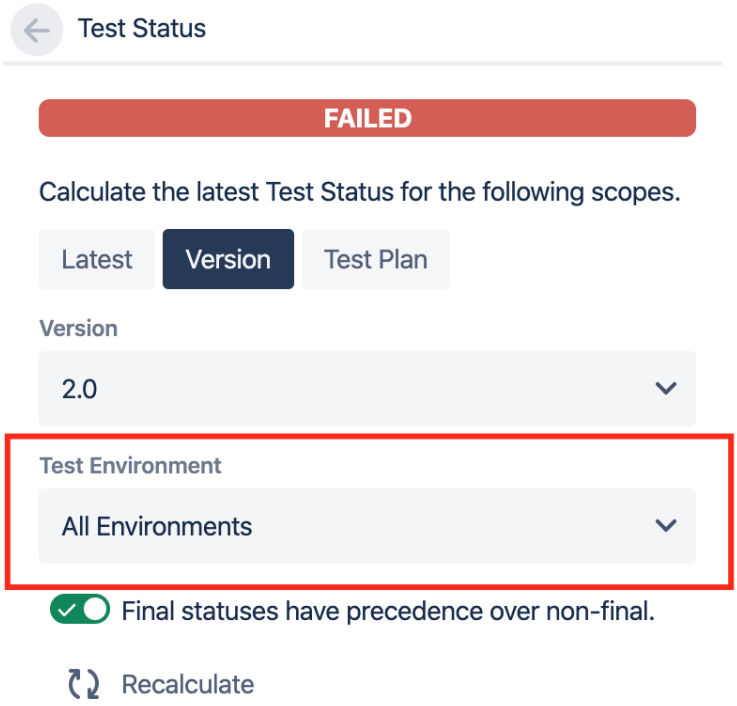
The same test has been executed in both Test Environments (a Test Execution per Test Environment).

Analyzing the impact of the results on different environments

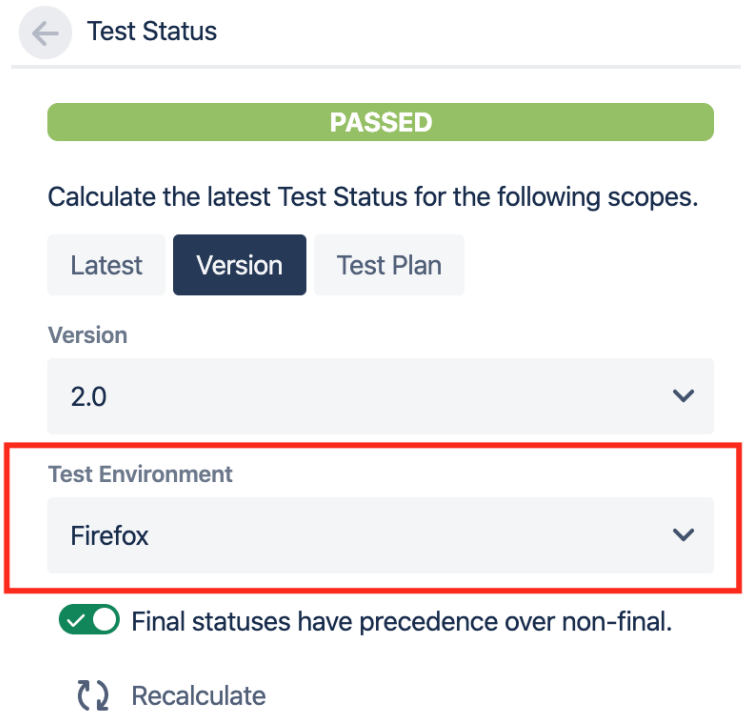
Results obtained for Test Environments will impact coverage.

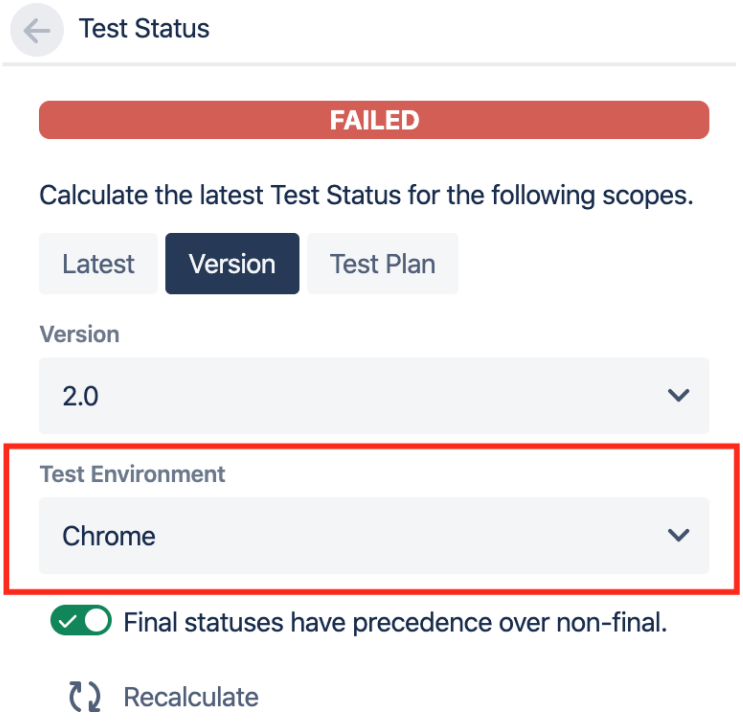
Considering the previous screenshot, the "Test Status" for the Test issue will show **FAILED** because the Test has failed for one of the environments. On the test issue page, a web panel named is provided by Xray that allows users to calculate the status of the Test considering different scopes.

If the "All Environments" option is selected for the Test Environment scope field, Xray will consider the latest Test Execution for each environment which, in this case, will result in **FAILED** .



If you want to analyze the test status and show the latest results on a specific environment, just use the picker on the "Test Status" section. As seen ahead, this will produce different results because different results were obtained in different environments.



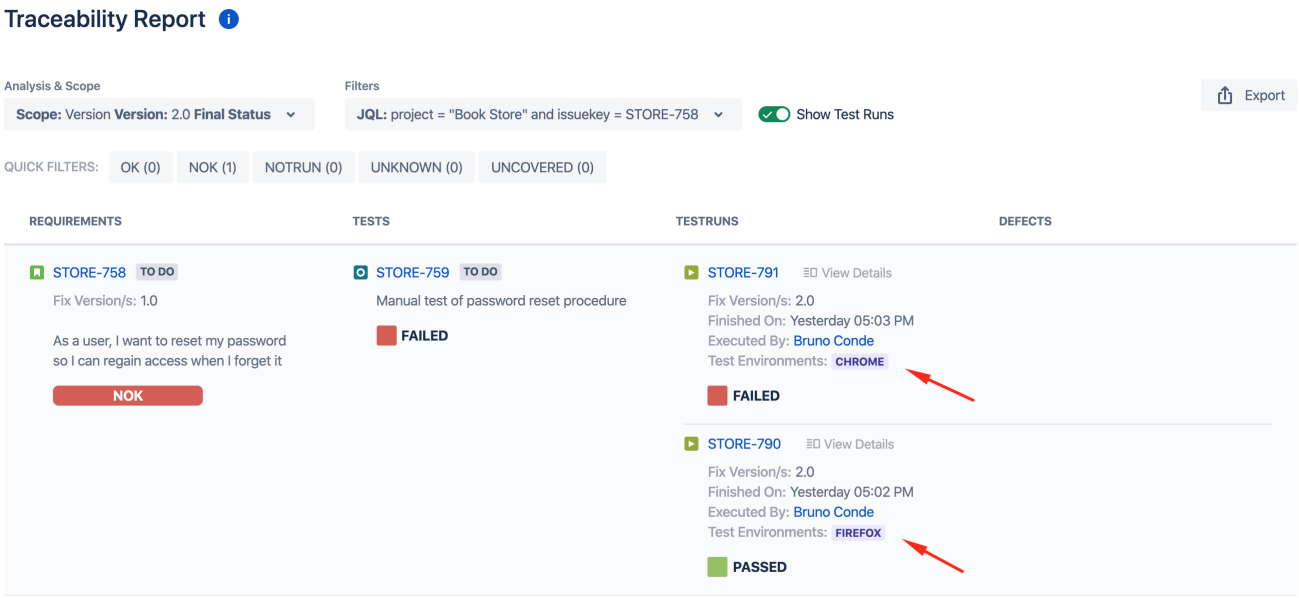


Please check Coverage Analysis to learn more about coverage analysis possibilities.

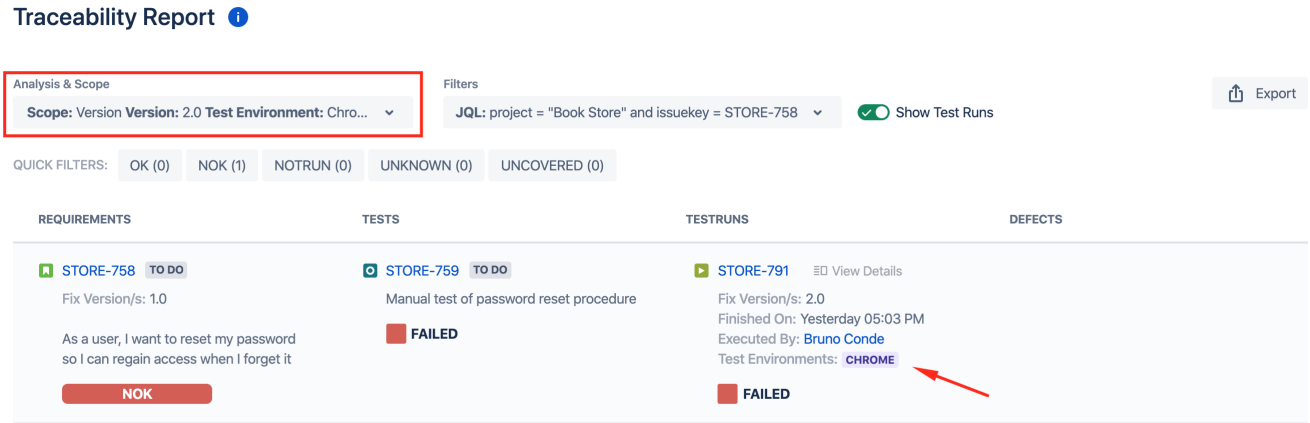
Just like we can calculate the status for different Test Environment scopes on the Test issue, we can perform the same analysis on Requirement issues. If you go into a Requirement issue covered by Test cases, a web panel named "Test Coverage" will be visible on the issue page allowing users to analyze the coverage status. This analysis also includes the Test Environment scope field. Hence, you can also see the status of the Requirement issues based on the Test Environments in which the Tests were executed. The following screenshot displays this status of a Requirement issue for "All Environments".

It is also possible to analyze testing thoroughly considering Test Environments; t his analysis can be done using the Traceability Report or the Overall Coverage Report, among others.

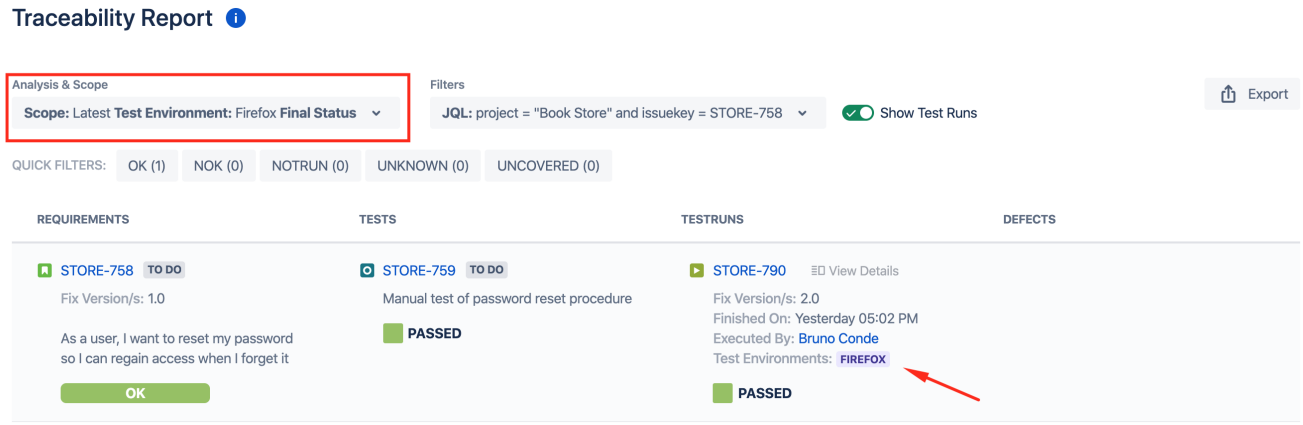
The exact behavior upon choosing a specific Test Environment depends on the report itself but, either explicitly or implicitly, Test Runs will be filtered by the selected Test Environment and reports will reflect it.



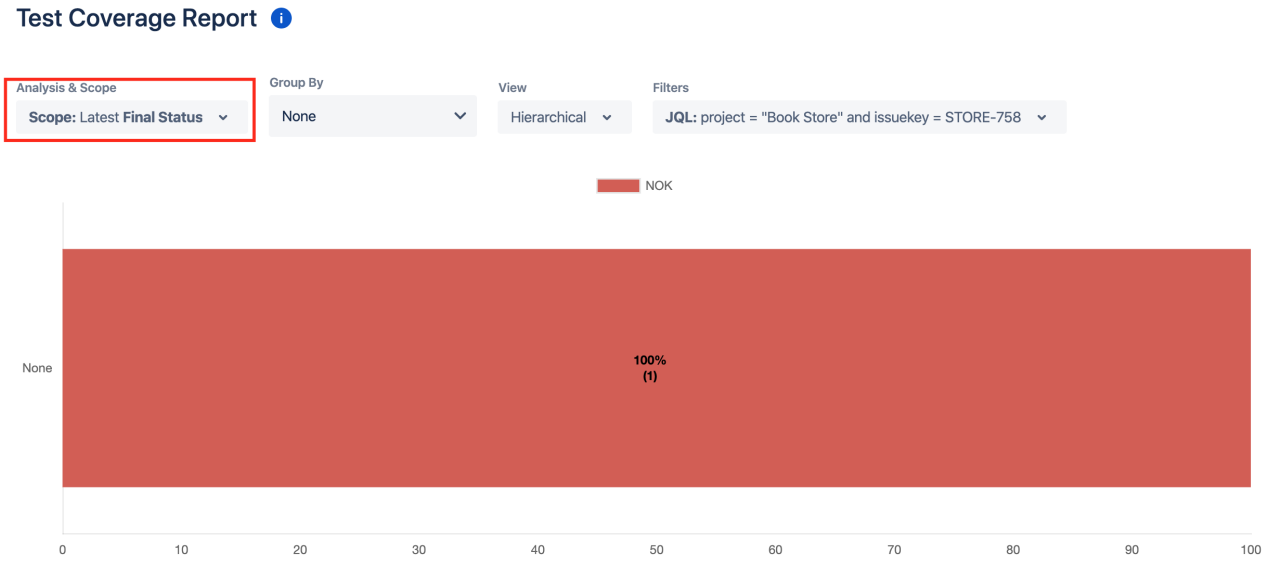
Traceability Report being used to analyze the results of a Story for "All Environments".



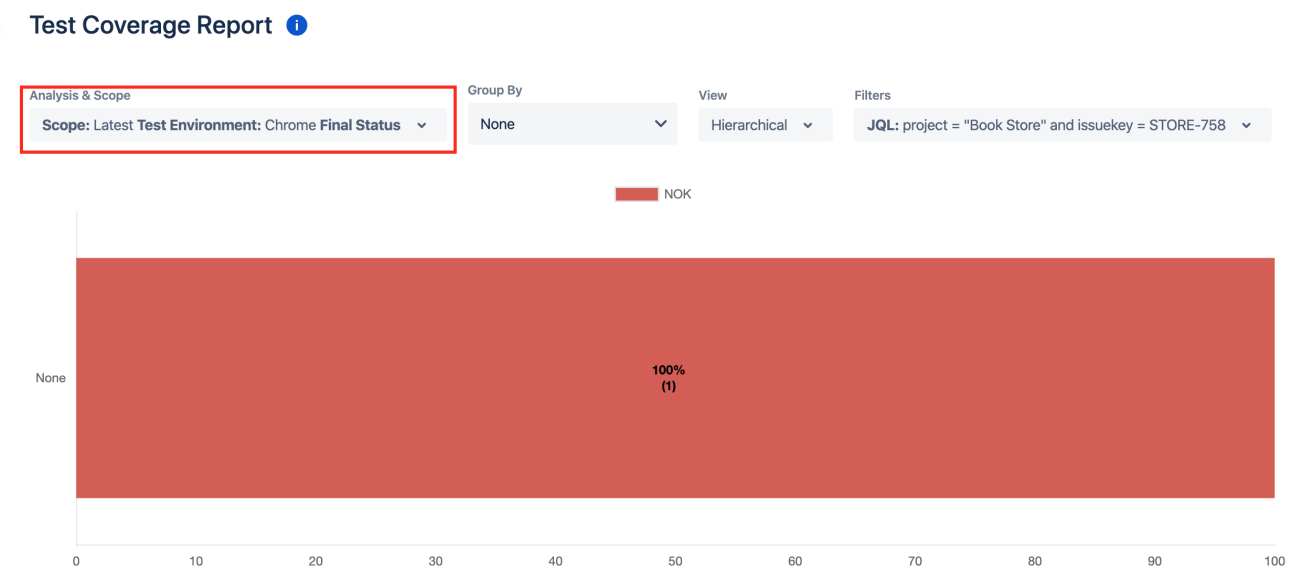
Traceability Report being used to analyze the results of a Story for the "Chrome" Test Environment.



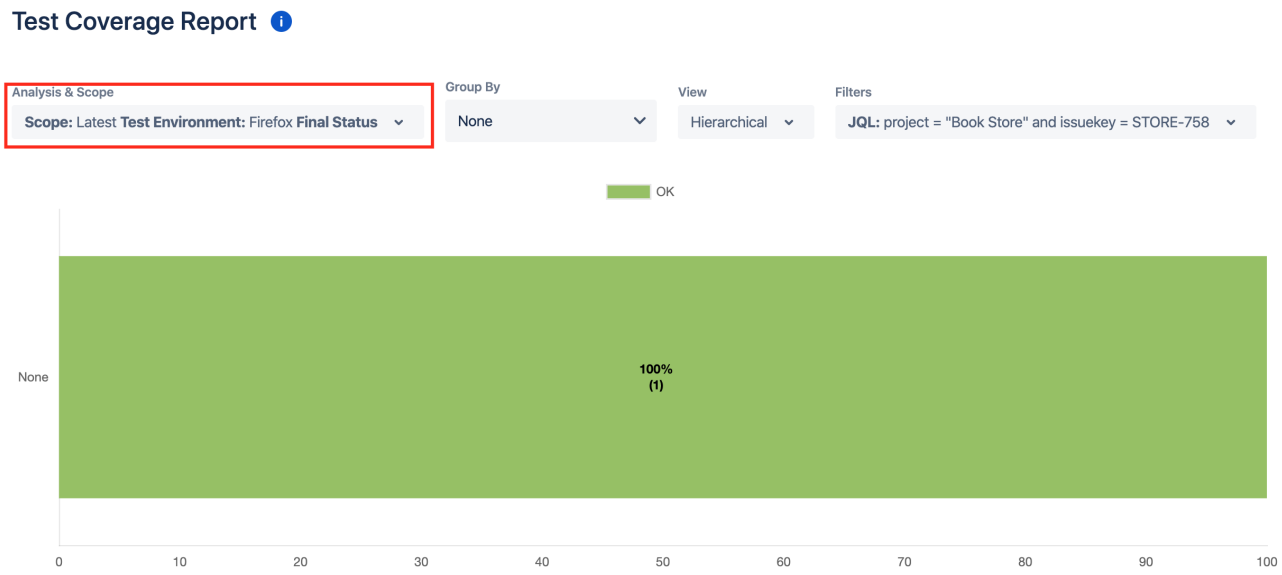
Traceability Report being used to analyze the results of a Story for the "Firefox" Test Environment.



Test Coverage Report being used to analyze the results of a Story for "All Environments".



Test Coverage Report being used to analyze the results of a Story for the "Chrome" Test Environment.



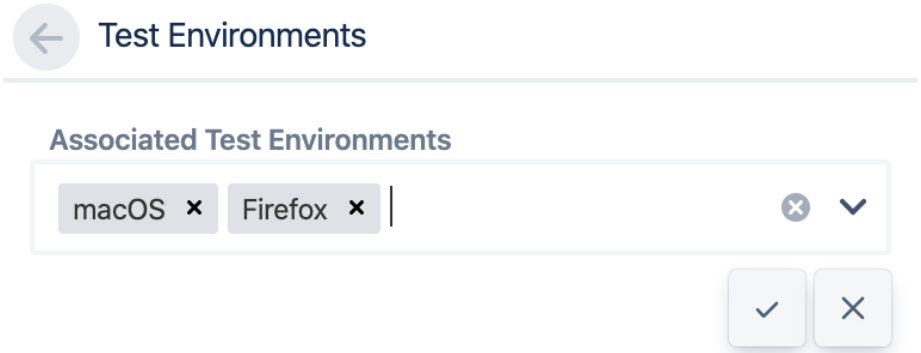
Test Coverage Report being used to analyze the results of a Story for the "Firefox" Test Environment.

Using multiple environments at the same time

Sometimes, you may have multiple categorizations for a given environment; in theory, you can think as it being something multidimensional.

Consider a very basic example: whenever performing web/UI based testing you will be using a browser and an operating system and you may want to analyze the results per a browser perspective or per an operating system perspective.

The recommended way to deal with environments having multiple dimensions is to treat each dimension (e.g. browser name, operating system name) individually. In other words, add the values of each dimension to the "Test Environments" field separately.



Whenever you assign "macOS" and "Firefox" to the Test Environments of a given Test Execution, it's equivalent to saying that your Test Run is scheduled for/was run in the "macOS" and also in the "Firefox" environment.

This approach will limit the number of environments to the total number of possible values for each dimension, as opposed to having *<number_of_values_dimension_1>*<number_of_values_dimension_2>*...* environments.

The drawback of this solution is that you won't be able to analyze the results for an environment tagged as "macOS" and "Firefox" at the same time, for example; you can just analyze results from a specific dimension.

To have in mind

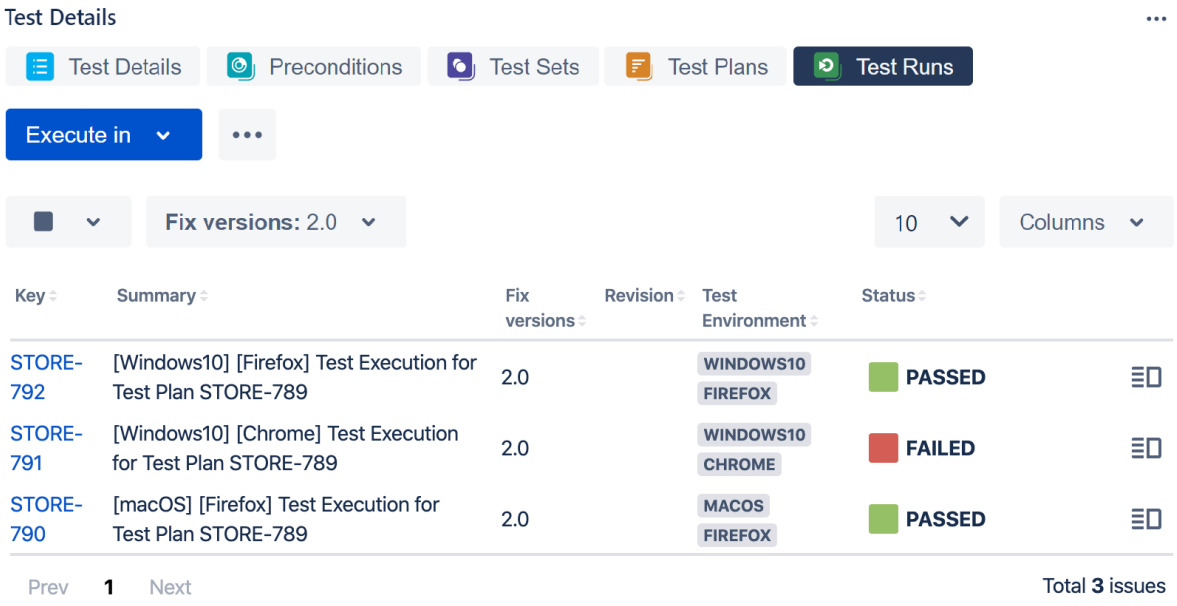
One way to deal with these kinds of environments would be to flatten them and treat them as usual, i.e. you could name the environment such as “Windows10_Chrome” or “macOS_Firefox” but…

- You could have a ton of composed environments which wouldn’t be manageable at all
- You couldn’t analyze coverage just from the perspective of one of those variables (e.g. “macOS” or “Firefox”); you would be restricted to analyze it from the perspective of the composed environment

Example

Test executed in the context of Test Execution assigned to several environments at the same time

1. Windows10, Chrome (failed)
2. Windows10, Firefox (passed)
3. macOS, Firefox (passed)



The calculated status of the test, per Test environment, will be the following.

	Status	Why?
Windows10	**PASSED**	due to the last result obtained in "Windows10" environment on STORE-792
macOS	**PASSED**	due to the last result obtained in the "macOS" environment on STORE-790
Firefox	**PASSED**	due to the last result obtained in "Firefox" environment on STORE-792
Chrome	**FAILED**	due to the last result obtained in "Chrome" environment on STORE-791
"All Environments" **(if analyzing the status of the test without identifying a specific environment)**	**FAILED**	as the last result for one of the environments ("Chrome") was FAIL (i.e. on STORE-791)

Tips and Recommendations

Do's

- Use Test Environments only if you want to run the **same** Test case in different environments and track results per each environment.
- Evaluate if you really need to assign multiple environments at the same time; using just one is preferable if you can afford that simplicity

Don'ts

- Don't create dozens or hundreds of Test Environments as it will harden their usage and add some performance overhead
- Don’t make composed environment names, such as ”<os>_<browser>_<stage>” as it will pollute the environment namespace and harden management
- Don't try to do data-driven testing using Test Environments; they're not tailored for that