Using Generic Tests for Automation

Overview

Generic Tests may be used in several ways: as exploratory or non-organized manual tests; they can also be used to have visibility of your automated tests in Jira.

A "Generic" Test can be used as an abstraction of an automated Test, so that it can be managed in Jira, linked to requirements, and we can report on its related results.

Basically, you manage Generic Tests the same way as you would do for other Test Types.

A Generic Test is uniquely identified by the issue key. We can also use the **Generic Test Definition** field to somehow identify the Test (e.g., by setting it with class and method that implements the test code, or the automated script file name).

Xray does not enforce any constraint on the **Generic Test Definition** field, so it is optional and you may use it as an additional field for quickly identifying what this test is all about.

- Overview
- How to use Generic Tests
  - Automated testing
    - Automatic provisioning
    - Manual provisioning
  - Exploratory testing

How to use Generic Tests

Automated testing

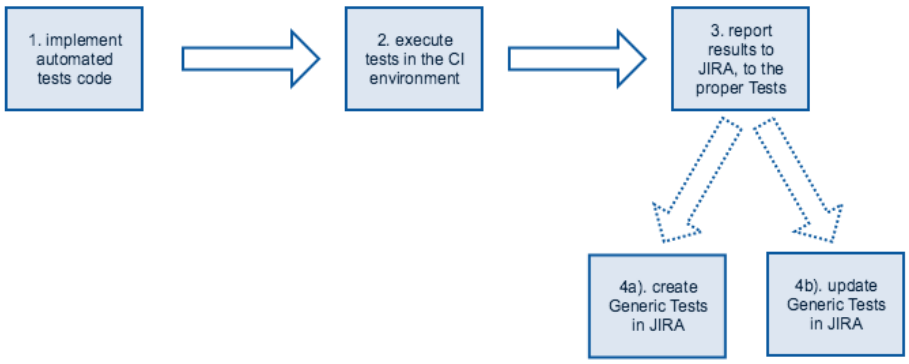
Automatic provisioning

Generic Tests may be used as an abstraction of automated tests, regardless of the testing framework and the technologies/platforms used.

In this scenario, common if you're using JUnit, TestNG, NUnit, Robot framework, developers/testers write the automated tests as code accordingly with the automation framework being used.

After that, they just need to submit the results to Xray and it will automaticaly create Test entities for each of them (or update existing ones if they already exist), so you have visibility of the automation results in Xray.

As soon as those Tests are linked to requirements, then you can evaluate the requirement coverage based also on the results of these automated tests.



1. Implement the automated test code, store it in the source control system, and put the reference to the Test in Jira (i.e., the issue key).
    a. This depends on the testing framework: it can be as a "tag" or as the test name, if the framework supports that.
2. Execute tests in the CI environment.
3. Report execution results using the format specific to the automation framework (either using the REST API directly or through Bamboo/Jenkins add-ons).
4. Create our update existing Tests in Xray

a. In the first iteration, Tests will be automaticaly created in Xray and the **Generic Test Definition** field will act as the test identifier (e.g. classname plus classmethod corresponding to automated test). Xray is able to identify uniquely from the report file
b. In second and onward iterations, existing Tests will be updated; Xray will use the **Generic Test Definition** field as the test identifier in order to find the existing Test in JIRA
c. Depending on the test automation framework being used, links can also be automaticaly created between the Tests and the requirements

After Tests are created in Jira, they can be managed in Jira. Thus, users can add additional information to them, associate them with other entities (e.g. requirements, Test Sets, etc)..

Learn more

You can see many examples in the page Integrating with Testing Frameworks, including:

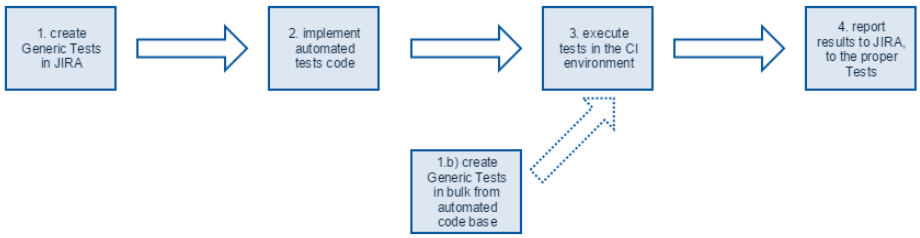
- Testing using Selenium and JUnit in Java
- Testing using TestNG in Java
- Testing using NUnit in C#
- Testing using Robot Framework integration in Python or Java

Manual provisioning

You may also create yourself Generic Tests as an abstraction of some test, so you can track its results in Jira. Generic Tests are great if you don't need to have a structured Test composed of steps (otherwise you would have to use Manual Tests as abstractions).

These Tests can be implemented as code in our own custom framework and you may report results back to them using the REST API and Xray's JSON format.

In general, you would follow this workflow.



1. Create a Generic Test in Jira.
    a. The test will be uniquely identified by the issue key; however, the **Generic Test Definition** custom field may be used as a more friendly way to identify the test (e.g., with the name of the test class/method, name of the script or executable implementing the test).
    b. The test can be automatically created in Jira when importing test results; as mentioned in the previous point, the **Generic Test Definition** field acts as the test identifier.
2. Implement the automated test code, store it in the source control system, and put the reference to the Test in Jira (i.e., the issue key).
    a. This depends on the testing framework: it can be as a "tag" or as the test name, if the framework supports that.
3. Execute tests in the CI environment.
4. Report execution results using Xray JSON format (and optionally, the REST API).
    a. In order to build the JSON file containing the test results, you will need to reference the Tests in Jira by their issue key.

If you have a project already being validated by automated tests which are not in Jira, you may create the Generic Tests in bulk either by building a CSV and import it or by using Jira's REST API (see example here).

Learn more

See Automated Tests (Import/Export) for more information on Xray's JSON format and Import Execution Results - REST to know how to import those test results using the REST API.

Exploratory testing

Exploratory testing allows you to better understand your SUT, using your existing knowledge of the system and your expertize to find even more about it. That learning will be used as positive feedback/reinforcement.

You can use Generic Tests as means to abstract some exploratory session. You can use the **Generic Test Definition** field to identify the session charter (e.g. goals of that session/mission).

During the execution of your session, you can attach evidence, report if all went fine and if not you may create defects. You may also decide to create a Manual Test, with the reproducible steps, for validating that the problem does not arise in the future.

Please note

If you have multiple testers involved in your exploratory session, then you can create different Test Executions each one assigned to a different tester; you may use the Test Execution planning dates to define the fixed timeframe allocated to your session. The Test Execution description can be used to specify the tester charter, which complements and focuses the session charter for the tester(s) involved in that Test Execution. Please note that a Test Execution can only contain the same Test once.