Testing in BDD with Gherkin based frameworks (e.g. Cucumber)

- Overview
- Supported features
  - Main features
  - Supported Gherkin keywords
- Workflows
  - Jira based Cucumber workflow
  - Pure VCS based workflow
  - Legacy Cucumber migration workflow
- Common problems
  - Submission of Cucumber results says "No tests found in execution result"
  - Is it possible to create of Cucumber tests automatically when submitting Cucumber results?
  - Where is the master of information for Cucumber Tests?

Overview

Xray supports Gherkin based frameworks since its first version.

Gherkin is mainly used in BDD (behaviour-driven development) context. However, please note that Gherkin and BDD are two different things: you may use Gherkin to describe your test scenarios and still not follow BDD.

BDD starts by having a clear understanding of the business value, or the ultimate benefit of a user story, which is done by describing it properly (e.g. "*As a ..., I want to ..., So that **[benefit]***").

Having this understanding in mind will help to depict "behaviours" that the system must address; these are the actual acceptance criteria, which are described in scenarios using a “ubiquitous language”, as Dan North states. Gherkin can be used to provide the foundations of this language.

Key to BDD is having a shared/common understanding of the purpose of the feature: what is it supposed to do and why. This can be done collaboratively by the so-called "three amigos" (i.e. people from the business, development/coding and testing roles), which in the end need to agree on what composes a "feature" (i.e. the story itself along with the respective acceptance criteria as scenarios).

One of the frameworks used in BDD context is Cucumber, which Xray has prime support for. In this context, you may want to have a look at a tutorial for Java.

Similarly to what happens with other frameworks, with Cucumber "specification" and "implementation" are two distinct things, even though related: the test specification is done in natural language using Gherkin, while the implementation of each specification's phrase (i.e., step) is done in code (e.g., Java, Ruby or some other language).

Xray provides the tooling to create and manage the specification inside Jira and to import/export it. However, how the implementation is managed is outside of Xray's scope.

Please have a look at possible workflows below, to see how you can integrate the specification managed by Xray with the corresponding code stored in your version control system.

Besides Cucumber, Xray also supports other frameworks, including Behave and SpecFlow.

Supported features

Xray has comprehensive support for Gherkin. First, it supports the creation of Scenarios and Scenario Outlines as Test cases, using one of the possible values for the "Test Type" field (e.g. Gherkin, Cucumber).

Besides this, Xray also supports Backgrounds created as Pre-Conditions, using similar values for the "Pre-Condition Type" field.

The next sections detail in more depth all the features supported by Xray.

Main features

- creation/edition of Scenario, Scenario Outline and Background entities
- validation with Cucumber (in different languages), Behave and SpecFlow
- centralized place to manage and do refactoring on Gherkin steps using the Automated Steps Library
- automatic provisioning of Gherkin based steps on the Automated Steps Library
- bulk import/update based on .feature files using the REST API and/or one of the available CI plugins
- export of .feature files containing Scenario, Scenario Outline and Background based on the respective Tests and Pre-Conditions through the UI, REST API and/or one of the available CI plugins

Supported Gherkin keywords

From the primary Gherkin keywords, Xray supports:

- Given
- When
- Then
- And
- But
- Feature
- Scenario
- Scenario Outline
- Examples
- Background

Besides this, Xray also supports:

- """ (Doc Strings) - partial support (supported on creation/edition but not yet as reusable steps on the Automated Steps Library)
- | (Data Tables) - partial support (supported on creation/edition but not yet as reusable steps on the Automated Steps Library)
- @ (Tags), during the .feature export process based on the labels assigned to Test issues (e.g. Scenarios/Scenario Outlines) or during the import process
- # (Comments)

Workflows

One key thing that you must decide is: where do you want to manage the Cucumber scenarios? Preferably you will only have a single source of truth. Do you want to use Jira as the master of information? Or do you want to use Git, for example, as the master for your features and corresponding scenarios?

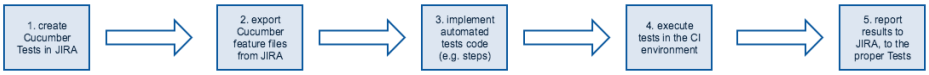
Please note

Having both Jira and the VCS (e.g. Git, SVN) as master of information for Cucumber Scenarios/Scenario Outlines can lead to synchronisation issues because the scenarios edited in JIRA won't be committed automatically to the VCS to the original feature files used.

Thus, you can use a mix of the Jira and VCS workflows below but have in mind that the scenarios can be out of synch between what you have in Jira and what you have in the VCS.

Jira based Cucumber workflow

In this workflow, Tests are created and managed in Jira, thus Jira will be the master for the Cucumber scenarios.



1. Specify Cucumber tests using natural language, in Jira.
2. Export Cucumber features from Jira to the CI environment, using the REST API.
3. Implement tests in code and commit them to the source code versioning system.
4. Execute tests in the CI environment.
5. Report results to Xray, using the REST API.

Note that Cucumber steps and related code may, in fact, be implemented before the first step, or may not even be needed to be implemented for a given Test if all the sentences are already implemented as code.

Learn more

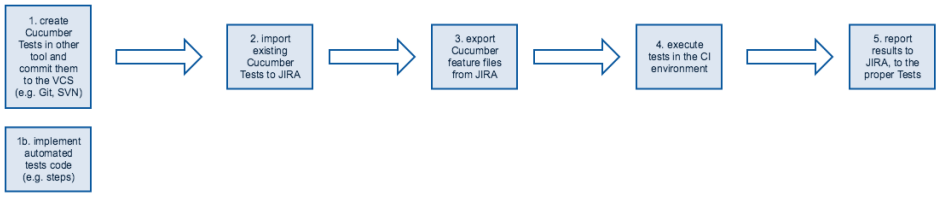
See Automated Tests (Import/Export) for more information on how Cucumber feature files are generated and how to import back results to Xray.

Pure VCS based workflow

Please note

The support for this workflow has been added in v3.1, in experimental mode.

In this workflow, the features are being "managed" in the source code VCS (versioning control system). This means that users will be editing features elsewhere, other than Jira, and they will want to synchronize the scenarios contained within those features to Jira. Therefore the master of information will be Git, SVN or whatever VCS is being used.



1. Create Cucumber tests (e.g. Scenario/Scenario Outlines) and their respective step code in some other tool.
    a. **add an unique tag to each Scenario/Scenario outline, in the form of "@id:xxx", in which xxx is a number (e.g. 1, 2, 3, ...)**
2. Import existing Cucumber tests using the REST API (e.g., from an existing code base which may have tests, and their corresponding step implementation)
3. Export Cucumber features from Jira to the CI environment, using the REST API or the Jenkins/Bamboo plugins. It is crucial to use these exported features instead of the ones from the VCS, since the features exported from JIRA will contain additional tags that will allow several things, including the ability to correctly report the results to back to the correct entities in Jira, afterwards.
4. Execute tests in the CI environment.
5. Report results to Xray, using the REST API or the Jenkins/Bamboo plugins.
6. Repeat from step 1 onward.

Please note

This flow will work fine as long as:

- An unique tag "@id:xxx" is added to each Scenario/Scenario Outline within the file. The id just needs to be unique (e.g. sequential) within the feature file
- The path of the feature file is not changed

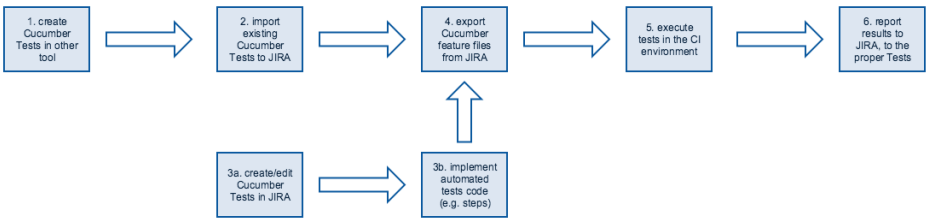
Is any of the previous conditions is not satisfied then new and duplicated Tests, and Pre-Conditions, may be created.

See an example in Importing Cucumber Tests - REST.

Legacy Cucumber migration workflow

The workflow below applies to legacy/existing projects that already have Cucumber tests in some version control system (e.g., GIT, SVN), possibly alongside the software's implementation.

Existing Cucumber tests are initially imported to Jira. Editing the tests can then proceed on the Jira side. While test editing can also occur simultaneously in Jira and in some other tool, it is highly recommended to keep it only on one side, preferably Jira.



1. Create Cucumber tests (e.g., Scenario/Scenario Outlines) and their respective step code in some other tool.
2. Import existing Cucumber tests using the REST API (e.g., from an existing code base which may have tests, and their corresponding step implementation)
3. Optionally, and in parallel,
    a. Specify additional or edit existing Cucumber tests in natural language, in Jira.
    b. Implement remaining tests in code and commit them to the source code versioning system.
4. Export Cucumber features from Jira to the CI environment, using the REST API or the Jenkins/Bamboo plugins.
5. Execute tests in the CI environment.
6. Report results to Xray, using the REST API or the Jenkins/Bamboo plugins.
7. Repeat from step 3 onwards.

Common problems

Submission of Cucumber results says "No tests found in execution result"

Most probably you're trying to submit a Cucumber JSON results file without having previously created the Cucumber Tests (e.g., Scenario/Scenario Outline) in Jira.

You need to first have these Test issues created in Jira, with the correct specification. Then, you need to export those Tests to one or more Cucumber .feature files, so you're able to run them in your CI environment.

Following this workflow is important because whenever you export the features from Jira, along with the corresponding Cucumber Scenarios/Scenario Outlines, the Scenarios will be tagged with labels that will allow Xray to map the results back to the proper entities in Jira.

Is it possible to create of Cucumber tests automatically when submitting Cucumber results?

No. It's not possible to properly recreate the Cucumber Scenario/Scenario Outline/Background from a Cucumber result file in JSON format.

Where is the master of information for Cucumber Tests?

Our recommendation is to manage it in Jira and always edit the Cucumber Scenarios/Scenario Outlines/Backgrounds there. You can use your VCS (e.g. Git, SVN) as the master but don't try to use it and at the same time also use Jira as master.