Importing Cucumber Tests - REST

Importing Cucumber Tests

The following endpoint is provided to import a Cucumber .feature file or a zip file containing multiple .feature files. The files in the zip file may be in folders /subfolders.

In addition to the .feature or a zip file the endpoint also accepts a JSON file which can specify additional information that will be added to the Test or Precondition at the moment of their creation. The format of JSON where additional information for the Test and Precondition can be specified is similar to the one Jira uses to create/update issues, for more information about the format, check the Jira documentation here.

Cucumber ".feature" file	**/api/v1/import/feature**

The endpoint parameters are:

- projectKey - Jira project Key
- projectId - Jira project ID
- source - a name designating the source of the features being imported (e.g. the source project name)

Each .feature file will be processed and will try to find or create a Test/Pre-Condition inside the project given in the *projectKey* parameter. We use the following rules:

Tests:

1. If Test key is present in the file:
    a. exists in Jira, system tries to find the Test by key and update it; else...
    b. does not exists in Jira, system returns an error message.
2. If Test key is not present in the file, system tries to find the Test having:
    a. the same source and original relative path of .feature (e.g. "[project_lambda]core/sample_addition.feature) and
        i. same ID (extracted from a scenario label named id:xxx) or
        ii. same summary; else...
    b. the same summary in the target project; else...
    c. create Test in that project. The tags used in the scenario/scenario outline are also added as labels.

Pre-Conditions:

1. If Pre-Condition is present in the file:
    a. exists in Jira, system tries to find the Pre-Condition by key and update it; else...
    b. does not exist in Jira, system returns an error message.
2. If Pre-Condition key is not present in the file, system tries to find the Pre-Condition having:
    a. the same source and original relative path of .feature (e.g. "[project_lambda]core/sample_addition.feature); else...
    b. the same summary in the target project; else...
    c. create Pre-Condition in that project. The tags used in the background are also added as labels.

The mapping from the Scenario/Scenario Outline present in the .feature files to the Test issues in Jira would be as follows:

Scenario/Scenario Outline	Test in JIRA
name of the Scenario/Scenario Outline	"Summary" field
steps	"Scenario" field
tags of the Scenario/Scenario Outline	labels

The "Feature" section is not imported since the feature itself should exist previously as a Jira requirement issue (e.g., story).

The exception is the tags before the "Feature: " section; if a requirement issue is found for the specified key, then a "Tests" link is created between the Test and the requirement issue.

If the Cucumber feature has a background, a Pre-Condition issue will be created, if the issue key in the tag does not exist in JIRA, or updated, containing the information provided in that background.

If the background has no name, then the Summary of the Pre-Condition is going to be a string containing the keys of the Tests of that Cucumber feature (e.g. "Background for: CALC-1, CALC-2").

Below is an example of a .feature file containing a Scenario Outline and two Pre Conditions:

```
@REQ_CALC-889
Feature: As a user, I can calculate the sum of 2 numbers

        Background:
                #@PRECOND_TX-114
                Given that the calculator is turned on
                And the mode is to advanced
                #@PRECOND_TX-155
                Given that the calculator has been reset

        @UI @core
        Scenario Outline: Cucumber Test As a user, I can calculate the sum of 2 numbers
                Given I have entered <input_1> into the calculator
                And I have entered <input_2> into the calculator
                When I press <button>
                Then the result should be <output> on the screen

                  Examples:
                    | input_1 | input_2 | button | output |
                    | 20      | 30      | add    | 50     |
                    | 2       | 5       | add    | 7      |
                    | 0       | 40      | add    | 40     |
                    | 4       | 50      | add    | 54     |
```

In this other hypotetical example, a feature file is shown containing one Background and two tests: one Scenario Outline and a Scenario. Each Scenario /Scenario Outline is identified by an internal "id:xxx", in order to uniquely identify the scenario within the feature file.

```
@REQ_CALC-1910
Feature: As a user, I can calculate the sum of two numbers

Background:
  Given I have a calculator
  And I have some fingers


@id:1 @fast
Scenario Outline: Cucumber Test As a user, I can calculate the sum of two positive numbers
  Given I have entered <input_1> into the calculator
  And I have entered <input_2> into the calculator
  When I press <button>
  Then the result should be <output> on the screen

Examples:
| input_1         | input_2         | button| output|
| 20                  | 30                 | add        | 50         |
| 2                  | 5                 | add         | 7          |
| 0                  | 40                | add         | 40         |
| 4                  | 50                | add         | 54         |
| 5                  | 50                | add         | 55         |


@id:2
Scenario: Cucumber Test As a user, I can calculate the sum of two negative numbers
  Given I have entered -1 into the calculator
  And I have entered -3 into the calculator
  When I press add
  Then the result should be -4 on the screen
```

Whenever this feature is imported for the first time, assuming it was imported from file named "features/addition.feature" in a zip file,

- a Pre-Condition with the summary "Background for: <issue_key_of_first_test>,<issue_key_of_second_test>" will be created;

- a Cucumber Test of type "Scenario Outline" will be created, having the summary "Cucumber Test As a user, I can calculate the sum of two positive numbers". The Test will have the label "fast";
- a Cucumber Test of type "Scenario" will be created, having the summary "Cucumber Test As a user, I can calculate the sum of two negative numbers";
- All previous Tests will be linked to the requirement CALC-1910

Whenever this same feature is imported for the second and following times, assuming it was imported from file named "features/addition.feature" in a zip file,

- a Pre-Condition with the summary "Background for: <issue_key_of_first_test>,<issue_key_of_second_test>" and the label "features/addition.feature" will be updated;
- a Cucumber Test having the labels "features/addition.feature" and "id:1" will be updated with the specification of the Scenario Outline; The Test will have the label "fast" (added if needed);
- a Cucumber Test having the labels "features/addition.feature" and "id:1" will be updated with the specification of the Scenario;
- All previous Tests will be linked to the requirement CALC-1910

Request

QUERY PARAMETERS

parameter	type	description
projectKey	*String*	key of the project where the tests and pre-conditions are going to be created.
projectId	String	id of the project where the tests and pre-conditions are going to be created.
source	String	a name designating the source of the features being imported (e.g. the source project name)

multipart/form-data:

"file" *: a* ***MultipartFormParam*** *containing a* ***".feature" file*** *or a* ***ZIP file*** *to import.*

"testInfo" *: a* ***MultipartFormParam*** *containing a* ***JSON file*** *with extra Test information. (****optional****)*

"precondInfo" *: a* ***MultipartFormParam*** *containing a* ***JSON file*** *with extra Precondition information. (****optional****)*

Example Request

curl -H "Content-Type: multipart/form-data" -X POST -H "Authorization: Bearer $token" -F "file=@1.feature" https://xray.cloud.getxray.app/api/v1/import/feature?projectKey=DEMO

curl -H "Content-Type: multipart/form-data" -X POST -H "Authorization: Bearer $token" -F "file=@features.zip" https://xray.cloud.getxray.app/api/v1/import/feature?projectId=1000

curl -H "Content-Type: multipart/form-data" -X POST -H "Authorization: Bearer $token" -F "file=@features.zip" https://xray.cloud.getxray.app/api/v1/import/feature?projectId=1000&source= project_lambda

curl -H "Content-Type: multipart/form-data" -X POST -H "Authorization: Bearer $token" -F "file=@1.feature" -F "testInfo=testInfo.json" -F "precondInfo=precondInfo.json" https://xray.cloud.getxray.app/api/v1/import/feature?projectKey=DEMO

Responses

200 OK : **application/octet-stream** : Successful. The cucumber features where successfully imported to Jira.

Example Output

```
{
    "errors": [
                "Error in file calculator.feature: Precondition with key DEMO-54321 was not found!"
        ],
    "updatedOrCreatedTests": [
        {
            "id": "32495",
            "key": "DEMO-15119",
            "self": "https://devxray3.atlassian.net/rest/api/2/issue/32495"
        },
        {
            "id": "32493",
            "key": "DEMO-15117",
```

```
            "self": "https://devxray3.atlassian.net/rest/api/2/issue/32493"
        }
    ],
    "updatedOrCreatedPreconditions": [
                {
            "id": "12345",
            "key": "DEMO-12345",
            "self": "https://devxray3.atlassian.net/rest/api/2/issue/12345"
        }
        ]
}
```


400 BAD_REQUEST : **text/plain :** Returns the error.

401 UNAUTHORIZED : **text/plain** : The Xray license is not valid.

500 INTERNAL SERVER ERROR : **text/plain** : An internal error occurred when generating the *feature* file(s).