Importing Tests using Test Case Importer

You can use the Xray's Test Case Importer to import new or existent tests, preconditions and test sets issues from a CSV or JSON source file.

The Test Case Importer follows a wizard like interface, guiding you through the process which envolves:

- preparing data (CSV/JSON) beforehand
- submit the data
- choose the destination project along with some global settings
- for CSV, define the mapping of CSV columns <=> fields in Jira

Supported issue types

The Test Case Importer is only able to create or update **tests, preconditions and test set** issues. If you wish to create or update other types of issues, you should use Jira's CSV importer instead. Note that Jira's CSV importer is not able to handle any Xray related semantics though.

- Accessing the Test Case Importer
  - From the top Apps menu
  - From within Testing Board in a project
- Importing data
  - CSV Importing
    - Downloadable examples
    - Preparing the source file
    - Open Test Case Importer
    - File Import step
    - Setup step
    - Map fields step
  - JSON importing
    - Preparing the source file
    - Open Test Case Importer
    - File Import step
    - Setup step
  - Results of the import process
  - Configuration file
- Examples

See and try some examples by yourself

Please see a tutorial with working Examples using Test Case Importer, showcasing different scenarios, which you can download and try by yourself.

Accessing the Test Case Importer

Xray provides the ability to import multiple Tests, Preconditions or Test Sets at once either as a standard user or as a Jira administrator; the user can create or update all those issues types with a maximum number limit of 1000 issues per import.

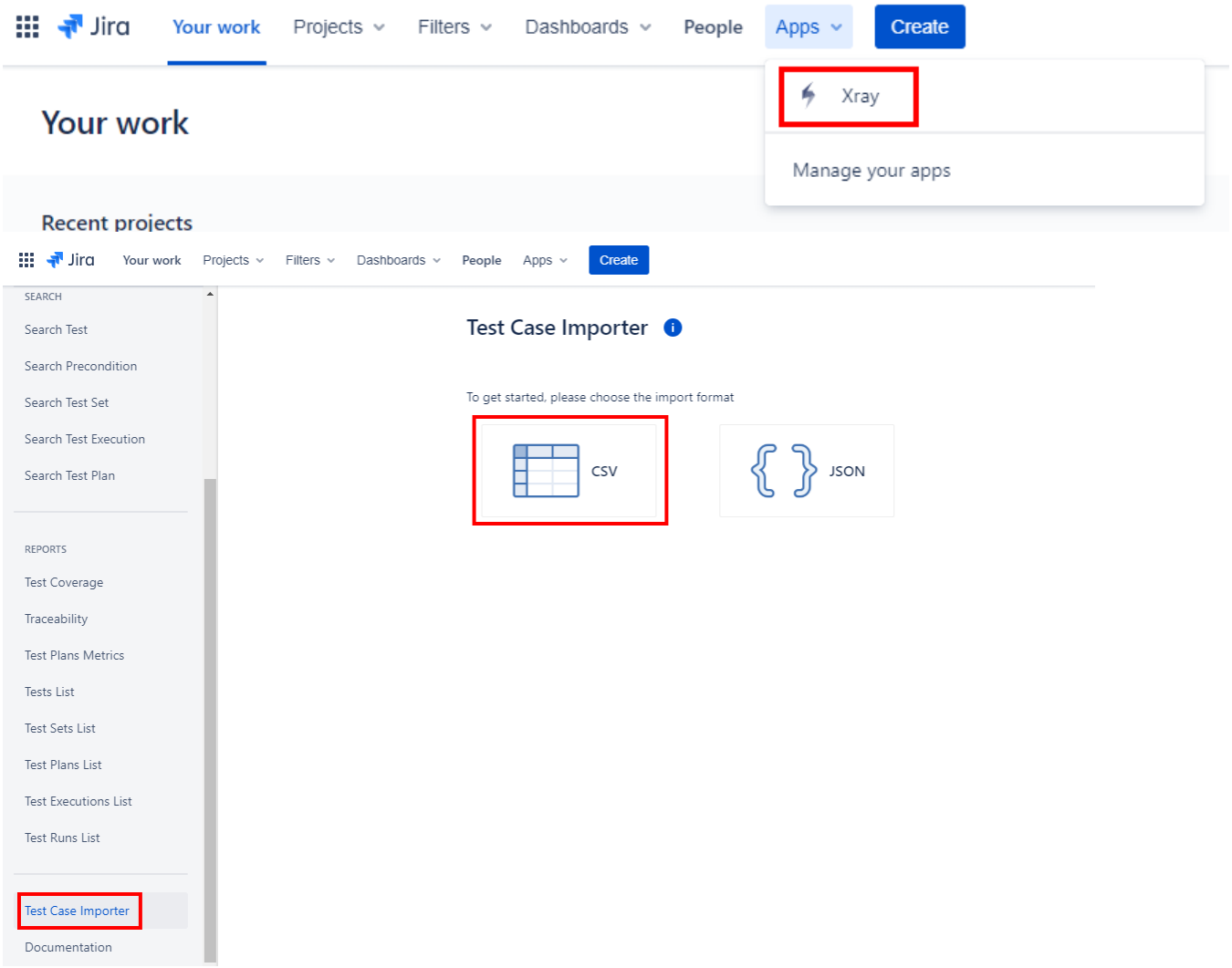
Permissions

Only users that have the **Make Bulk Changes** permission will have this option available. The configuration can be found in the *Jira Administration / System*.

Additionally, the user will only be able to import data into projects where he has the **Create Issues** permission. This configuration can be found in the *Jira Projects / Project Settings / Permissions.*

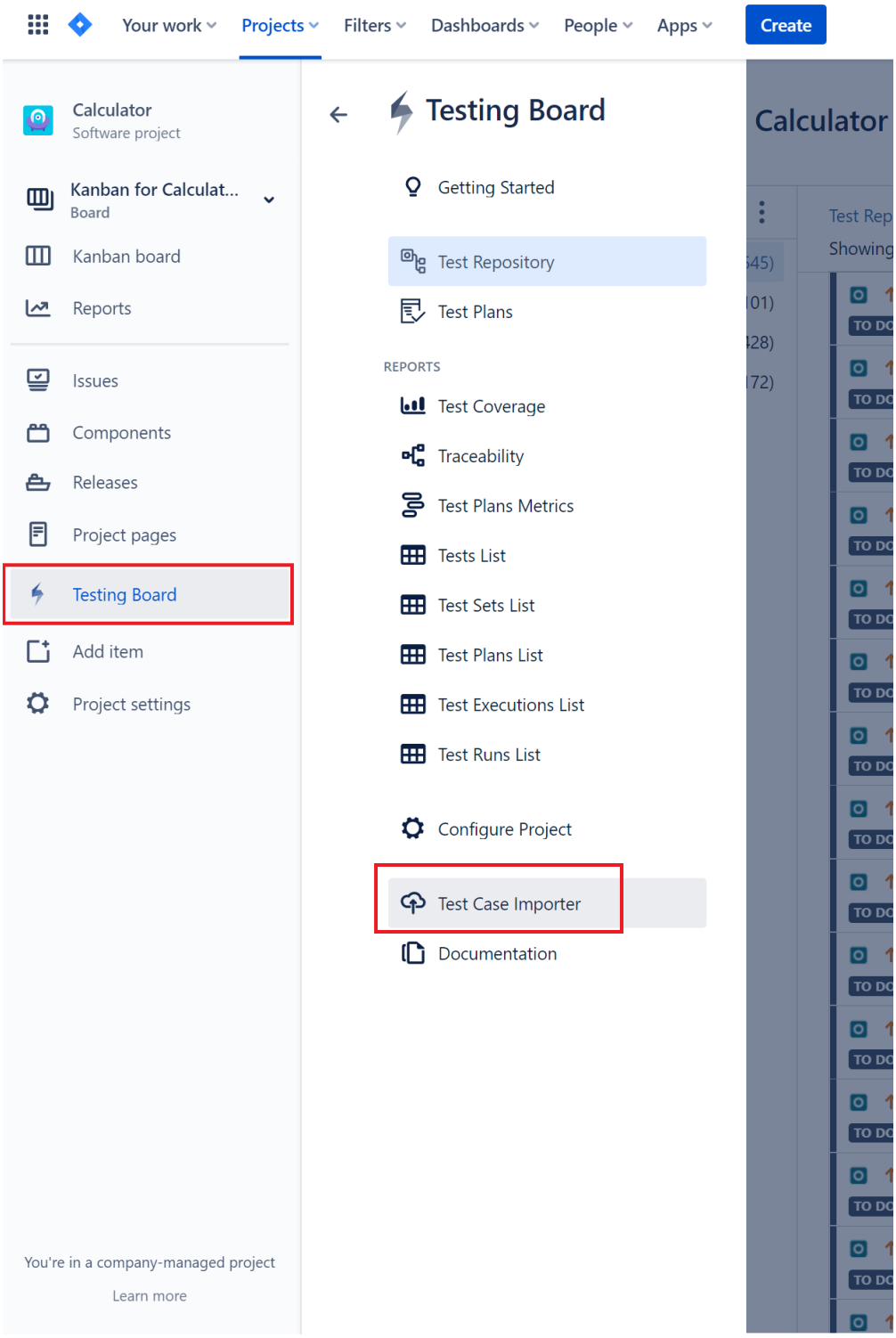
From the top Apps menu

Go to **Apps > Xray**. In the "Xray" side menu, select "Test Case Importer" and then select the format you want to use (**e.g. CSV**).

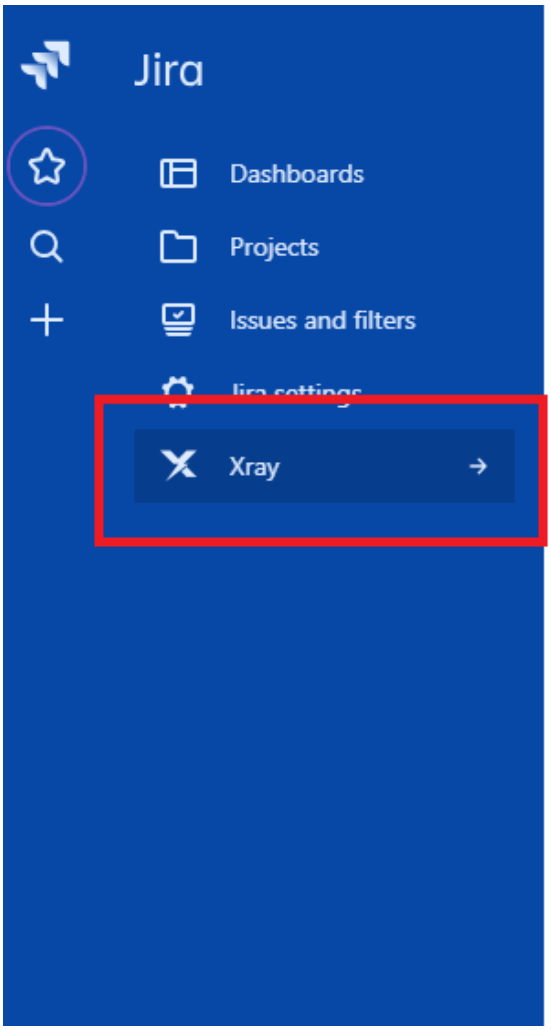


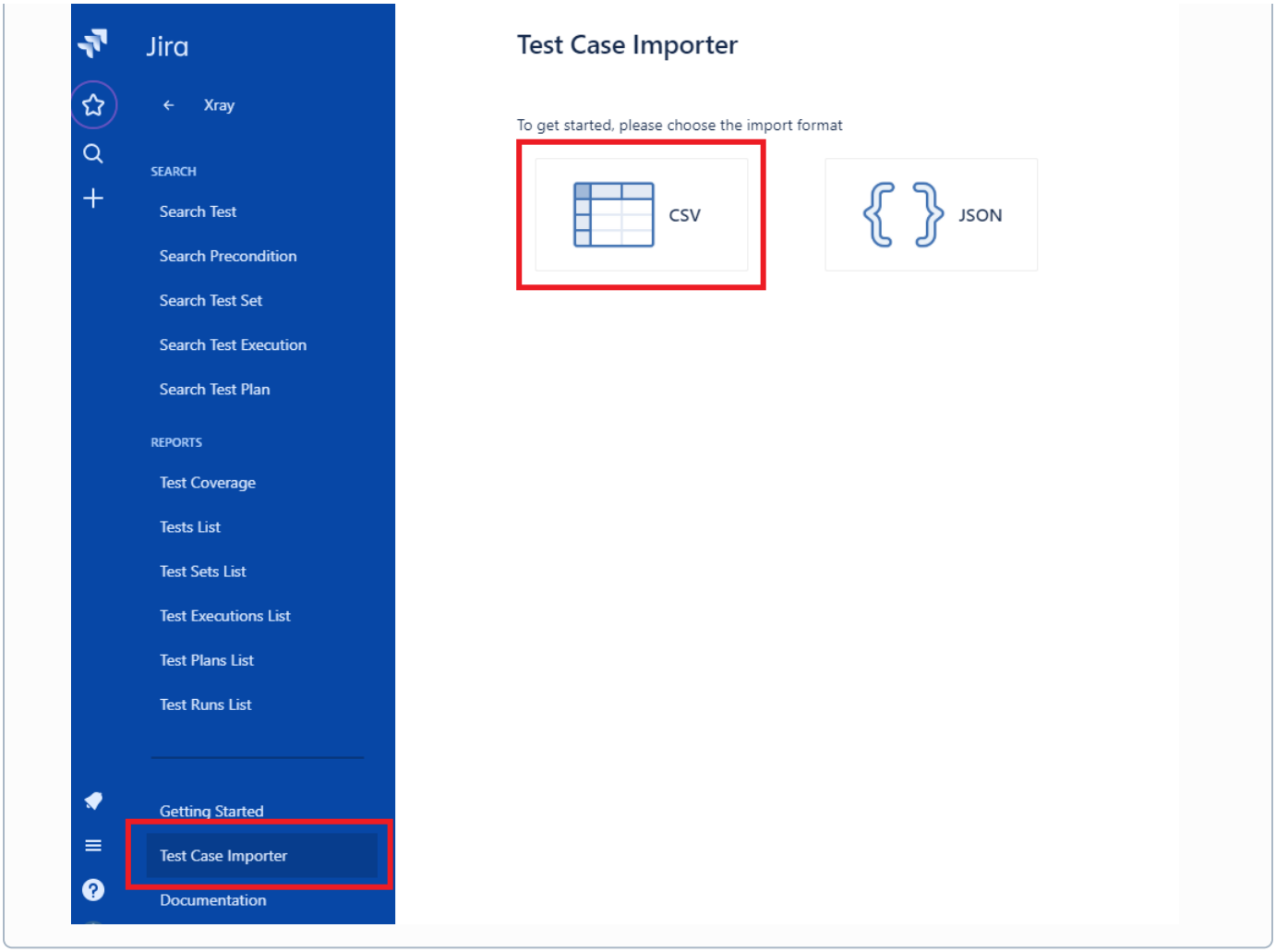
From within Testing Board in a project

The Test Case Importer is also accessible from within the Testing Board of a Xray-enabled project.



If you have the old Jira UI


Jira
Dashboards
Projects
Issues and filters
Xray



Importing data

CSV Importing

Downloadable examples

Please check multiple examples in the tutorial Examples using Test Case Importer (as well as code in GitHub).

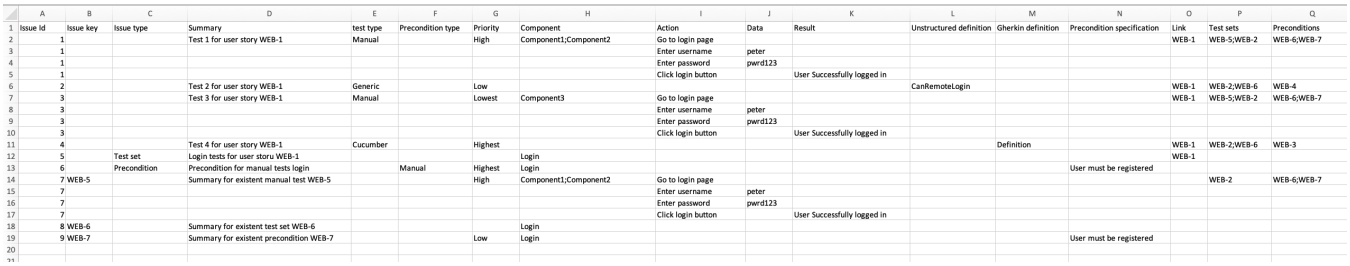
Preparing the source file

The CSV source file must follow some simple rules.

You can name the fields as you wish since they are going to be mapped during the importation process, but there are three mandatory fields that must be mapped:

1. **Issue Id** – a unique identifier for the issue, this field is used to group lines that belong to the same test case or to identify a precondition or a test set.
2. **Summary** – this field is mandatory since Jira doesn't allow you to create an issue without a summary.
3. **Test Type** – this field defines the test type of the each test to import. The test type must match one of the types of the project to which the test is being imported.

Here is an example of a source file as seen in a spreadsheet application:



Issue Id	Issue key	Issue type	Summary	test type	Precondition type	Priority	Component	Action	Data	Result	Unstructured definition	Gherkin definition	Precondition specification	Link	Test sets	Preconditions
1			Test 1 for user story WEB-1	Manual		High	Component1;Component2	Go to login page						WEB-1	WEB-5;WEB-2	WEB-6;WEB-7
1								Enter username	peter							
1								Enter password	pwrd123							
1								Click login button		User Successfully logged in						
2			Test 2 for user story WEB-1	Generic		Low					CanRemoteLogin			WEB-1	WEB-2;WEB-6	WEB-4
3			Test 3 for user story WEB-1	Manual		Lowest	Component3	Go to login page						WEB-1	WEB-5;WEB-2	WEB-6;WEB-7
3								Enter username	peter							
3								Enter password	pwrd123							
3								Click login button		User Successfully logged in						
4			Test 4 for user story WEB-1	Cucumber		Highest						Definition		WEB-1	WEB-2;WEB-6	WEB-3
5		Test set	Login tests for user storu WEB-1				Login							WEB-1		
6		Precondition	Precondition for manual tests login		Manual	Highest	Login						User must be registered			
7	WEB-5		Summary for existent manual test WEB-5			High	Component1;Component2	Go to login page							WEB-2	WEB-6;WEB-7
7								Enter username	peter							
7								Enter password	pwrd123							
7								Click login button		User Successfully logged in						
8	WEB-6		Summary for existent test set WEB-6				Login									
9	WEB-7		Summary for existent precondition WEB-7			Low	Login						User must be registered			

Here is the same example as seen in a text editor:

```
Issue Id;Issue key;Issue type;Summary;test type;Precondition type;Priority;Component;Action;Data;Result;Unstructured definition;Gherkin definition;Precondition specification;Link;Test sets;Preconditions
1;;;Test 1 for user story WEB-1;Manual;;High;"Component1;Component2";Go to login page;;;;;;WEB-1;"WEB-5;WEB-2";"WEB-6;WEB-7"
1;;;;;;;;;Enter username;peter;;;;;;;
1;;;;;;;;;Enter password;pwrd123;;;;;;;
1;;;;;;;;;Click login button;;User Successfully logged in;;;;;;
2;;;Test 2 for user story WEB-1;Generic;;Low;;;;;CanRemoteLogin;;;WEB-1;"WEB-2;WEB-6";WEB-4
3;;;Test 3 for user story WEB-1;Manual;;Lowest;Component3;Go to login page;;;;;;WEB-1;"WEB-5;WEB-2";"WEB-6;WEB-7"
3;;;;;;;;;Enter username;peter;;;;;;;
3;;;;;;;;;Enter password;pwrd123;;;;;;;
3;;;;;;;;;Click login button;;User Successfully logged in;;;;;;
4;;;Test 4 for user story WEB-1;Cucumber;;Highest;;;;;;Definition;;WEB-1;"WEB-2;WEB-6";WEB-3
5;;Test set;Login tests for user story WEB-1;;;;Login;;;;;;;WEB-1;;
6;;Precondition;Precondition for manual tests login;;Manual;Highest;Login;;;;;;User must be registered;;;
7;WEB-5;;Summary for existent manual test WEB-5;;;High;"Component1;Component2";Go to login page;;;;;;;WEB-2;"WEB-6;WEB-7"
7;;;;;;;;;Enter username;peter;;;;;;;
7;;;;;;;;;Enter password;pwrd123;;;;;;;
7;;;;;;;;;Click login button;;User Successfully logged in;;;;;;
8;WEB-6;;Summary for existent test set WEB-6;;;;Login;;;;;;;;;;
9;WEB-7;;Summary for existent precondition WEB-7;;;Low;Login;;;;;;;User must be registered;;;
```

Importing into multiple projects

The issues can all be imported to the same project or into multiple projects, one column in the source file can be used to define the destination project (either by having its key or id).

CSV column delimiter

If you choose the "," (comma) as the CSV column delimiter, then you must use quotation marks around any field that contains commas. The same is true if you use ";" (semicolon) as a delimiter and you want to use them in a field, for example as a list value delimiter (see the example above).

Fields with line breaks

If you need to use new lines within a field, for example, in the issue description, then you need to quote the field.

<<!clear!>> special marker

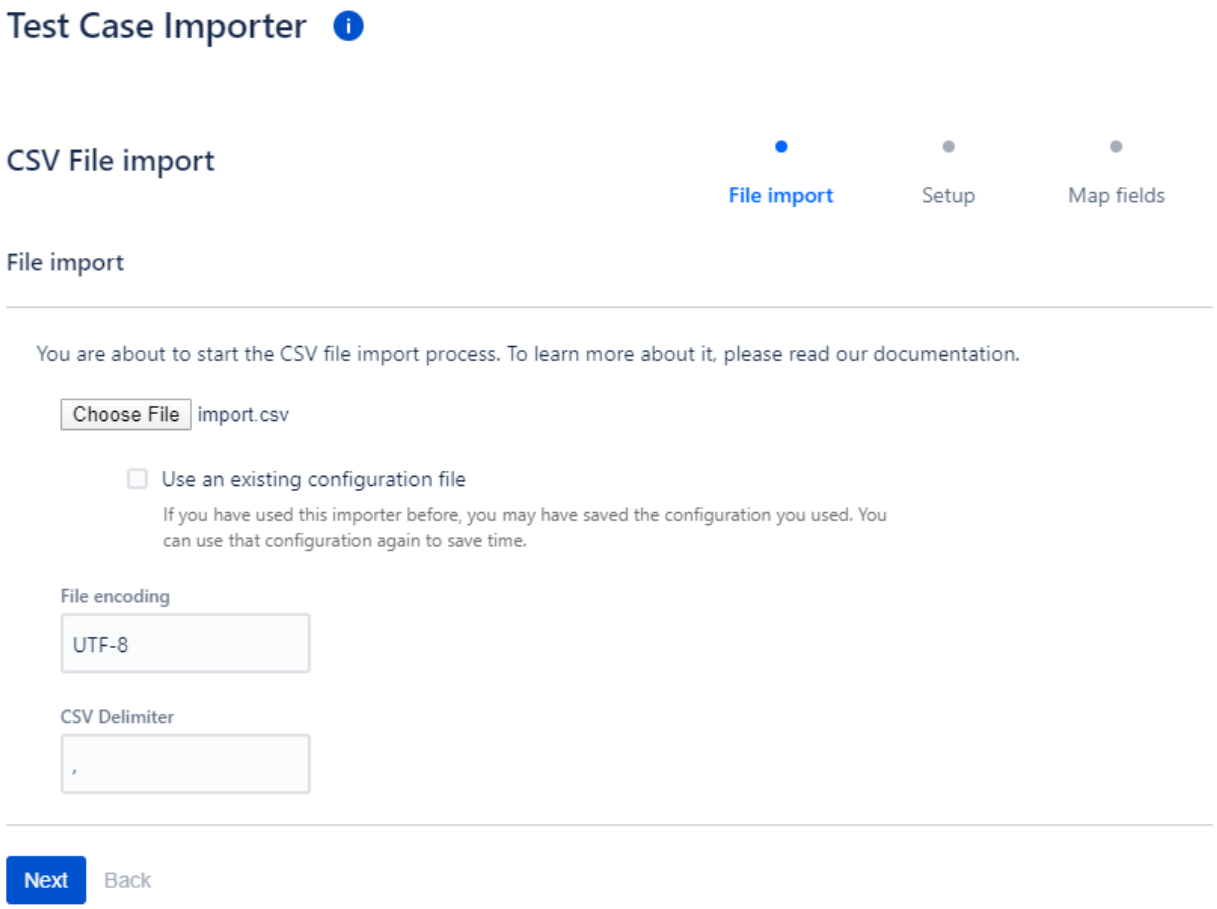
The <<!clear!>> special marker removes the values of the field (with the exception of test step fields) where is it specified (only applicable as a valid input of a csv file).

Open Test Case Importer

Open Test Case Importer and select CSV format.

File Import step

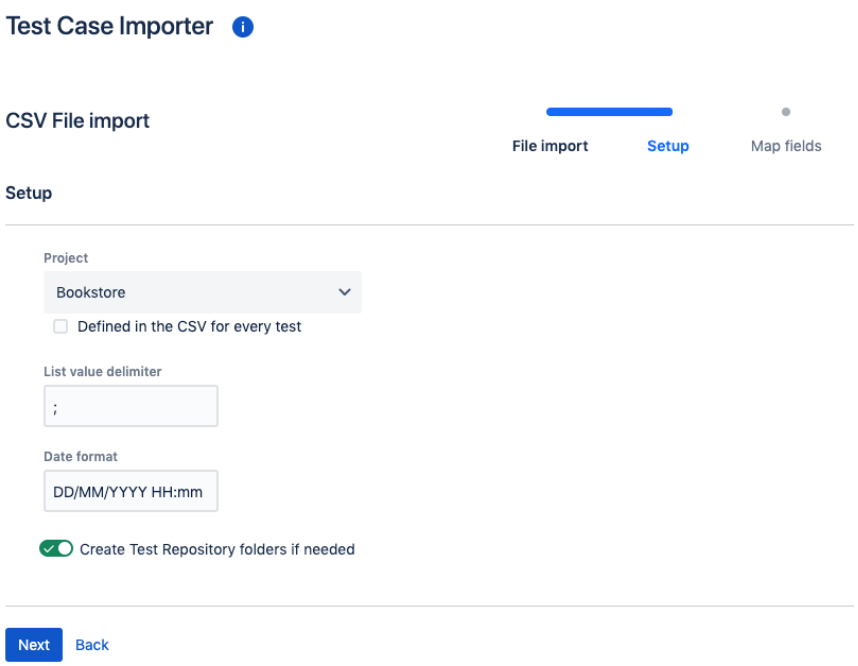
Provide the source file and settings regarding the file to import.



- An existing configuration file saved from the last import made with this file or a similar one.
- The file encoding used in the CSV source file; this is especially important when the file contains non-ASCII characters. The supported encodings can be seen here.
- The CSV delimiter is the column delimiter used.

Setup step

Additional setup information, for choosing the destination project along some settings related with the source data.



- The default project to which the tests will be imported into. Tests without project information associated will be imported into the project defined by this field.
- The list value delimiter is the delimiter used for fields that are lists of values.
- Date format used to parse the fields that are representing dates, refer to this page for help on how to define a valid format.
- Flag indicating if Test Repository folders should be created automatically if they do not exist.

Map fields step

During this step, you'll define the mapping of columns to Jira/Xray fields.

There are three mandatory fields that must be mapped: Test ID, Summary and Test Type.

Besides these ones, if you have defined additional fields as mandatory for Test issues, then you will also need to specify their mapping.

Test Case Importer

CSV File import

File import | Setup | Map fields

Map fields

Select the CSV fields to import and how you would like these converted to fields in JIRA

CSV Field		Jira Field
Issue Id (e.g. 1)	→	Issue ID
Issue key (e.g. First row doesn't have a value)	→	Issue Key
Issue type (e.g. First row doesn't have a value)	→	Issue Type
Summary (e.g. Test 1 for user story WEB-1)	→	Summary
test type (e.g. Manual)	→	Test Type
Precondition type (e.g. First row doesn't have a value)	→	Precondition Type
Priority (e.g. High)	→	Priority Name
Component (e.g. Component1;Component2)	→	Component Names
Action (e.g. Go to login page)	→	Action*
Data (e.g. First row doesn't have a value)	→	Data
Result (e.g. First row doesn't have a value)	→	Expected Result
Unstructured definition (e.g. First row doesn't have a value)	→	Unstructured Definition
Gherkin definition (e.g. First row doesn't have a value)	→	Gherkin Definition
Precondition specification (e.g. First row doesn't have a value)	→	Precondition Specification
Link (e.g. WEB-1)	→	Link "is blocked by" (inward)
Test sets (e.g. WEB-5;WEB-2)	→	Test Sets
Preconditions (e.g. WEB-6;WEB-7)	→	Preconditions

Begin Import | Back

Project field

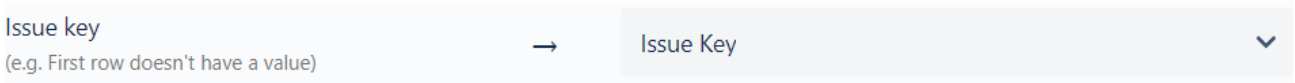
The project to which the test will be imported can be defined for each issue in a field. The fields "Project Key" or "Project Id" can be used for this.

If no project field is mapped or when the value in this field is empty, the project selected in the Setup step will be the one where these issues will be imported into.



Issue Key field

To update existing issues, your CSV file needs to contain a column that maps to Issue Key. If an issue exists for a given key, it will be updated.

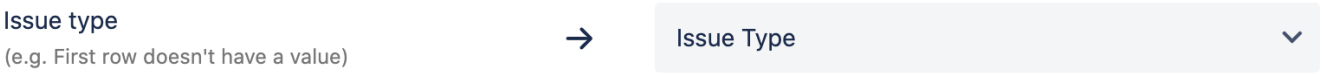


Issue Type field

The type of the issue being imported. The supported issue types are:

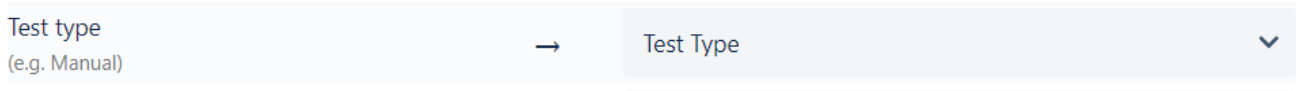
- Test ("test" case insensitive)
- Precondition ("precondition" case insensitive)
- Test set ("testset", "test_set" or "test set" case insensitive)

If no issue type is defined for an issue being imported, it will default to test. When updating issues, the field can be left empty on the csv file.



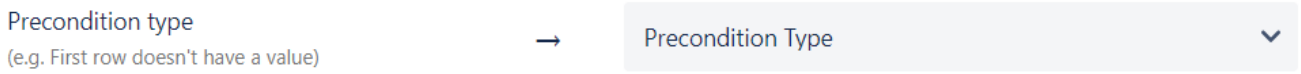
Test Type field

The type of the Test being imported. Must be filled for a Test in your CSV file, when creating one, and can be left empty when updating an existing Test.



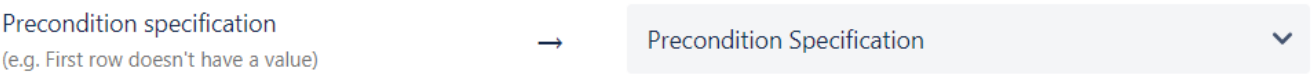
Precondition Type field

The type of the Precondition being imported. Must be filled for a Precondition in your CSV file, when creating one, and can be left empty when updating an existing Precondition.



Precondition Specification field

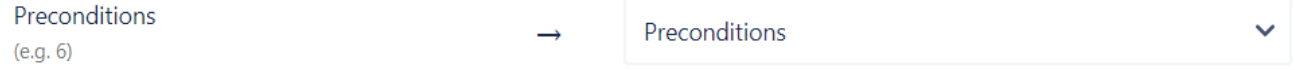
The "Steps", "Definition" or "Background" of a manual, generic or cucumber Precondition type, respectively.



Preconditions field

Only for test issues, a list of preconditions to which a test will be added.

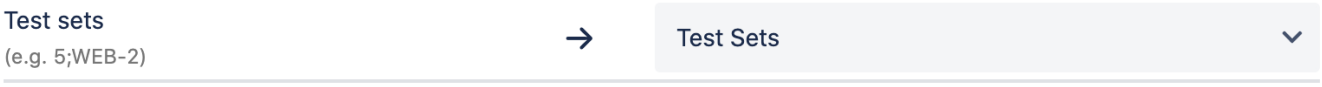
Each value should be the issue key of an existing precondition or the id of a precondition also being imported ("Test ID" field).



Test Sets field

Only for test issues, a list of test sets to which a test will be added.

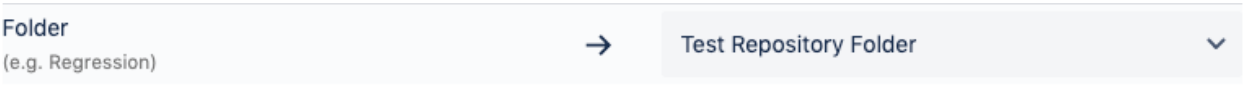
Each value should be the issue key of an existing test set or the id of a test set also being imported ("Test ID" field).



Test Repository Folder field

The Test Repository folder path to the folder in which the Tests will be associated.

The folder path consists of folder names separated by the "/" character, for example: *Main Tests/Sub Tests 1/Inner Tests 2*



List fields

List fields can hold multiple values separated by the list delimiter defined in the Setup step.



Link fields

Links can be created between imported tests and other existent issues using the "Link ..." Jira fields. Only applicable when creating an test.

The field is a list field and the value of the respective field should hold a list the issue keys of the issues to link to the test.

The direction (inward/outward) can also be specified.



Issue links limit

Due to technical restrictions there is a limit to the total amount of links the import file can have. The limit is 2000 issue links apart from the first one in every test.
In other words, every test in the import file can have an issue link defined, every additional issue link after that one, added up can be no more than 2000.
An error will be shown if this limit is exceeded.

Datetime fields

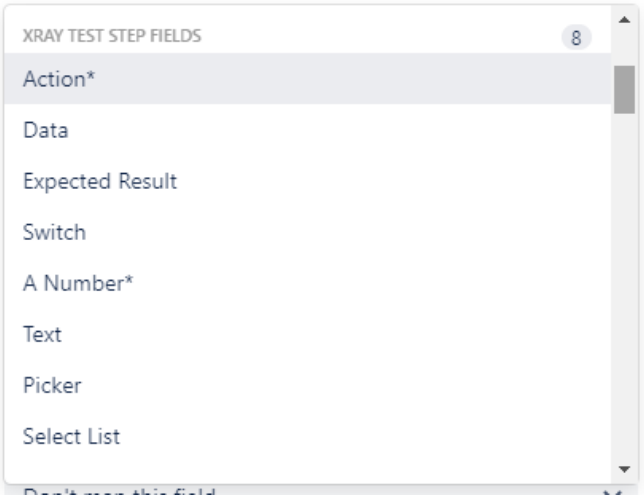
Datetime fields must have values that can be parsed using the Datetime format in the Setup step.

This page explains possible date formats and respective values.



Test step fields

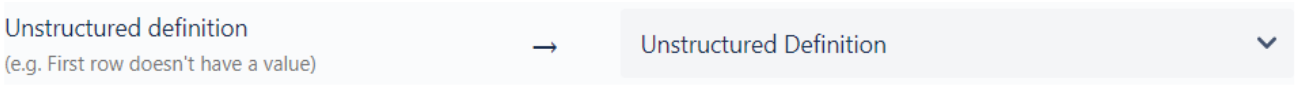
You can also map columns to test step fields for manual tests.
The fields are in the group "Xray Test Step Fields".



The configured test step fields for every project in the instance is shown here and the ones that are required are shown with a "*".

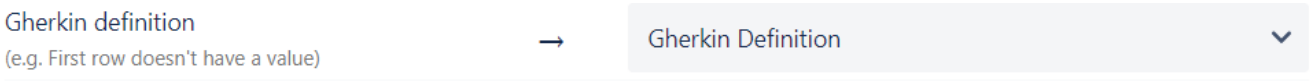
Unstructured Definition field

The "Definiton" of a generic Test type.



Gherkin Definition field

The "Scenario/Scenario Outline" of a cucumber Test type.



After this step the import process will start.

JSON importing

Preparing the source file

The JSON source file must have the same structure as the one used in the import through the REST API.

Example of one test being create and one existent test being updated in the JSON import format:

```
[
    {
        "fields": {
            "summary": "Test 1 for user story WEB-1",
            "priority": {"name": "High"},
            "components": [ {"name": "Component1"}, {"name": "Component2"}]
        },
        "xray_testtype": "Manual",
        "steps": [
            {
                "action": "Go to login page",
                "data": "",
                "result": ""
            },
            {
                "action": "Enter username",
                "data": "peter",
                "result": ""
            },
            {
                "action": "Enter password",
                "data": "pwrd123",
                "result": ""
            },
            {
                "action": "Click login button",
                "data": "",
                "result": "User successfully logged in"
            }
        ],
        "update": {
            "issuelinks": [
                {
                    "add": {
                        "type": {"id": "10201"},
                        "outwardIssue": {"key": "WEB-1"}
                    }
                }
            ]
        },
        "xray_test_sets": ["5","WEB-2"],
        "xray_preconditions": ["6"]
    }
]
```

```
[
    {
        "fields": {
            "summary": "Summary for existent manual test WEB-5",
            "components": [ {"name": "Component1"}]
        },
        "key": "WEB-5",
        "steps": [
            {
                "action": "Go to login page",
                "data": "",
                "result": ""
            },
            {
                "action": "Enter username",
                "data": "peter",
                "result": ""
            },
            {
                "action": "Enter password",
                "data": "pwrd123",
                "result": ""
            },
            {
                "action": "Click login button",
                "data": "",
                "result": "User successfully logged in"
            }
        ],
        "update": {
            "issuelinks": [
                {
                    "add": {
                        "type": {"id": "10201"},
                        "outwardIssue": {"key": "WEB-1"}
                    }
                }
            ]
        },
        "xray_test_sets": ["5","WEB-2"],
        "xray_preconditions": ["6"]
    }
]
```

Example of one Test Set being created and one existent Test Set being updated in the JSON import format:

```
[
    {
        "xray_issue_type": "testset",
        "xray_id": "5",
        "fields": {
            "summary": "Login tests for user story WEB-1",
            "project": {
                "key": "WEB"
            }
        }

    },
    {
        "key": "WEB-6",
        "fields": {
            "summary": "Summary of existing Test Set WEB-6",
            "project": {
                "key": "WEB"
            }
        },

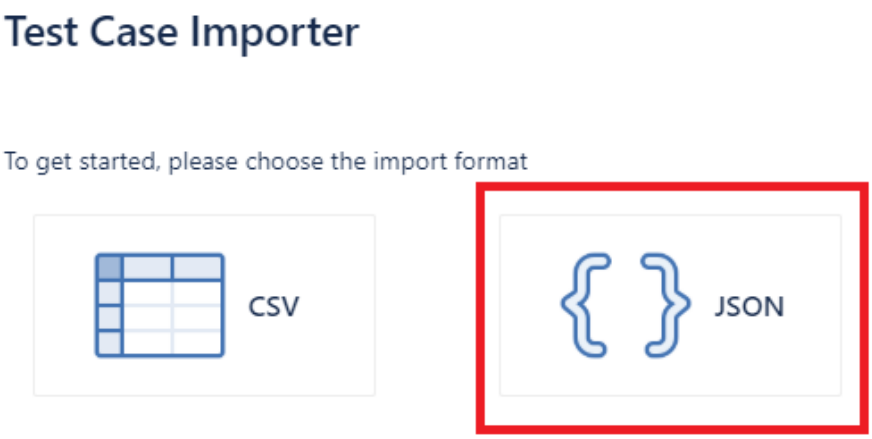
    }
]
```

Example of one Precondition being created and one existent Precondition being updated in the JSON import format:

```
[
    {
        "xray_issue_type": "precondition",
        "xray_id": "6",
        "fields": {
            "summary": "Precondition for manual tests login",
            "project": {
                "key": "WEB"
            }
        },
        "xray_precondition_type": "Manual",
        "xray_precondition_specification": "specification"
    },
    {
        "key": "WEB-7",
        "fields": {
            "summary": "Summary of existent Test Set WEB-7",
            "project": {
                "key": "WEB"
            }
        },
        "xray_precondition_specification": "specification"
    }
]
```

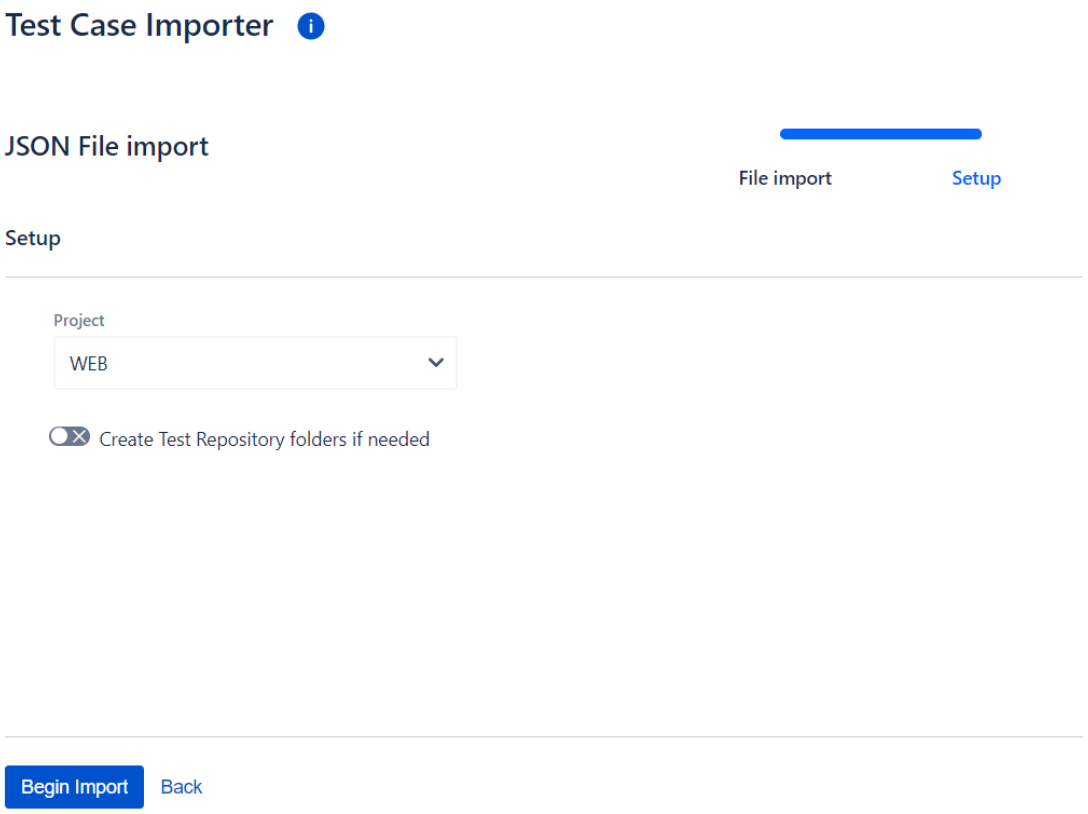
Open Test Case Importer

Open Test Case Importer and choose the JSON file format.



File Import step

Choose the file having the source data to import from.

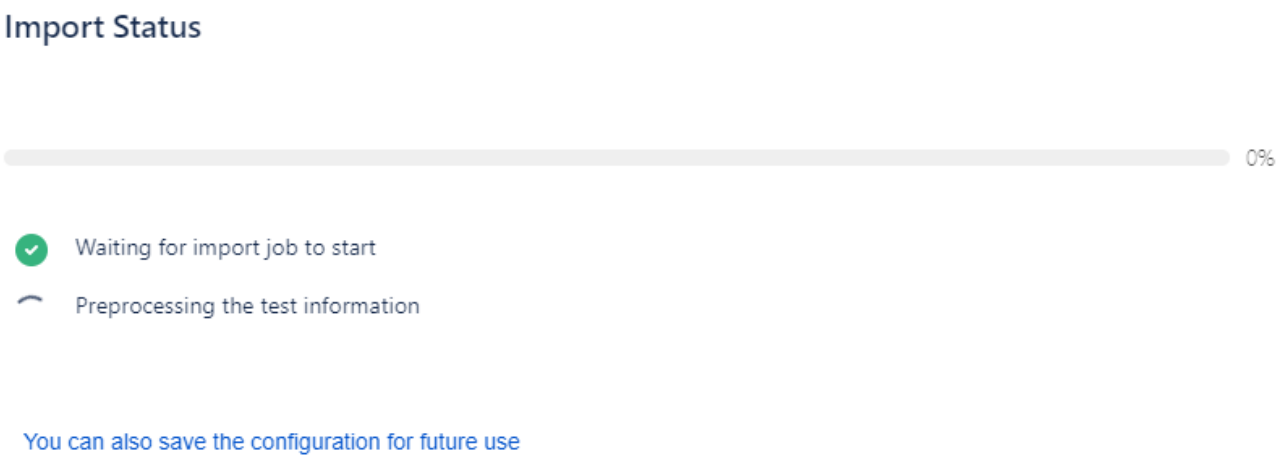


Setup step

Choosing the default project to import into. If each test to import already refers to a project, this field is not necessary to be filled, otherwise you should choose a project.

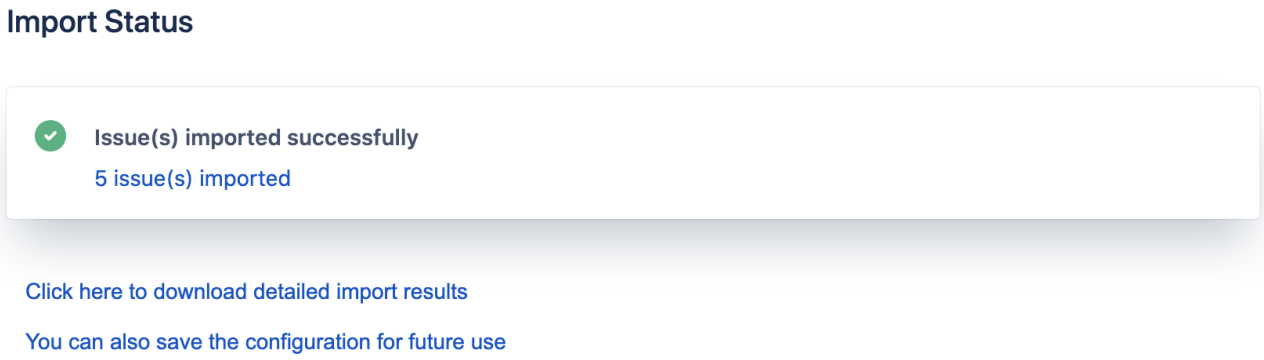
Results of the import process

After the import process has began its status will be periodically updated in the Test Case Importer page. No new import process can be started until the current one finishes.



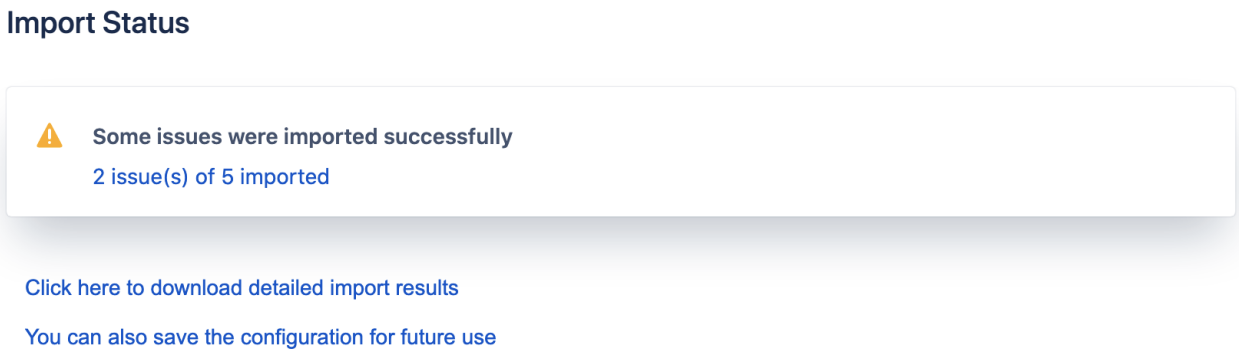
After the process finishes the result will be visible for about a day and then it will be removed from the page.

Example of 5 tests imported successfully:



You may also download the detailed import results, this is useful to get the issue keys of the imported issues and also to see which issues were not imported and why.

Example of 2 issues imported out of 5:



Example of detailed import results:

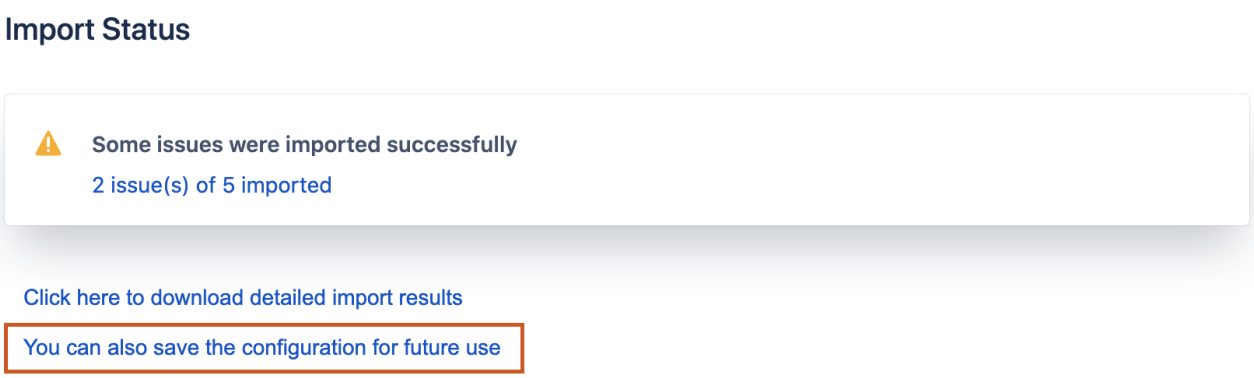

```
{
    "errors": [
        {
            "elementNumber": 0,
            "errors": {
                "components": "Component name 'Component5' is not valid"
            }
        },
        {
            "elementNumber": 1,
            "errors": {
                "priority": "Specify the Priority (name) in the string format"
            }
        }
    ],
    "issues": [
        {
            "elementNumber": 2,
            "id": "455085",
            "key": "WEB-39",
            "self": "https://xray-staging.atlassian.net/rest/api/2/issue/455085"
        }
    ],
    "warnings": []
}
```


In the detailed import results, the `elementNumber` field represents to which issue the information corresponds to (elementNumber 0 being the first issue).

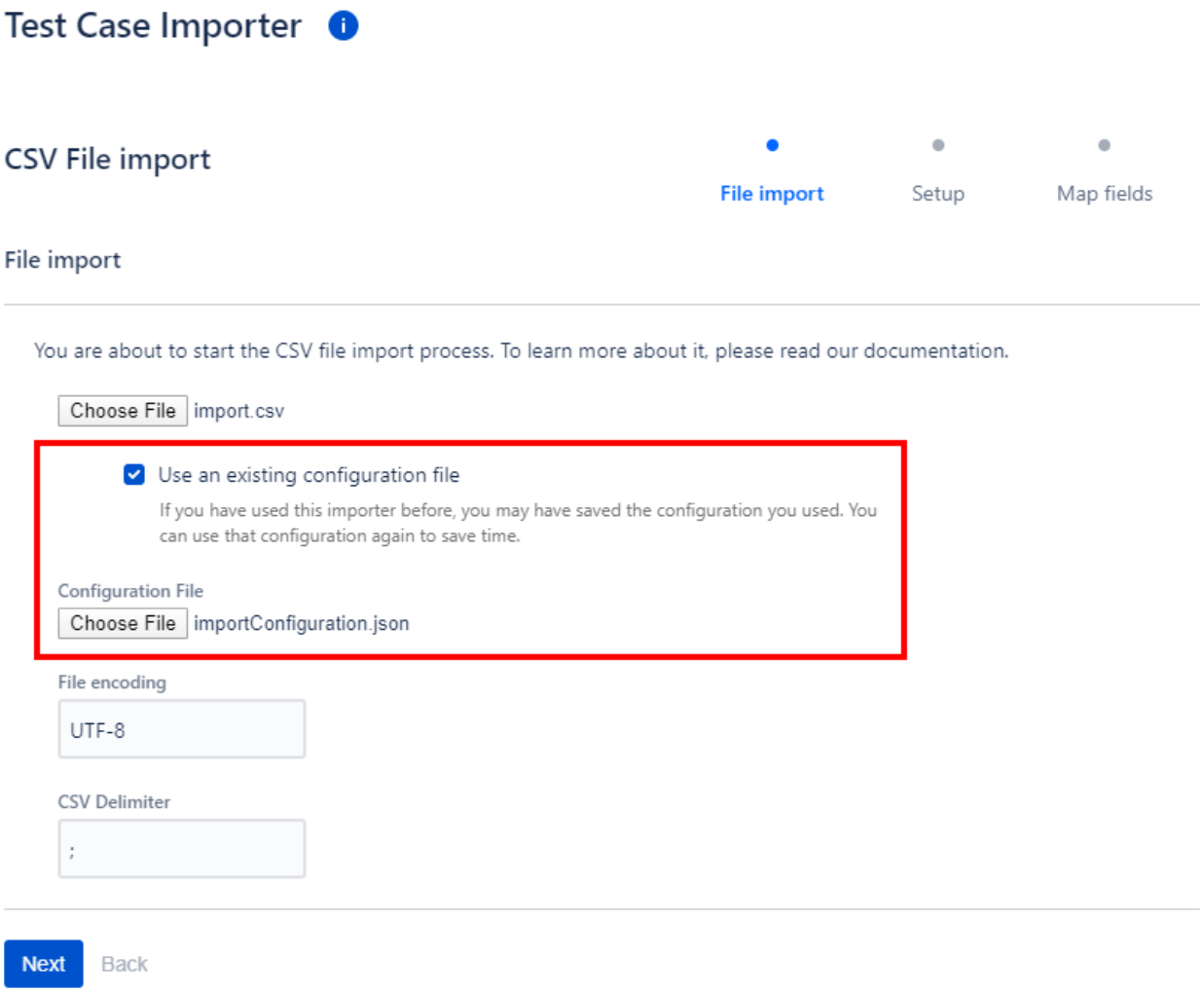
When importing tests into Test Repository folders, if for any reason the process is unable to create a folder or move a test to the respective folder, the respective tests will be still be migrated and a warning detailing the situation will be written to the import result.

Configuration file

When importing a file, when the import status is being shown, you can download the configuration file that will have all the settings you've set:



And then you can use it when you import the next file (if the configurations are supposed to be the same or similar):



Examples

Please see a tutorial with working Examples using Test Case Importer, showcasing different scenarios, which you can download and try by yourself.