Parameterized Tests

- Overview
  - Parameterized tests
  - Datasets
    - Parameter types
    - Combinatorial parameters
    - Dataset scopes
    - Dataset limits
  - Parameterized preconditions
  - Execution
  - Parameterized tests vs test environments
- Defining a parameterized test
  - Manual Steps
  - Generic or Unstructured definition
  - Gherkin Scenarios
- Creating a dataset
- Importing a dataset from a CSV file
- Exporting a dataset to a CSV file
- Overriding a dataset in a test plan or test execution

Overview

Test parameterization is a powerful practice that allows the same test to be executed multiple times with different parameters. Parameters are similar to input values (variables) that can change with each execution.

Without the ability to define parameterized tests, all values must be hard-coded into the specification, which makes the test static and difficult to modify. Static tests lead to a lot of redundancy since you need to define and clone the same test for the different combinations of values needed to cover different scenarios or variations of the same test.

By extracting the variability of the test specification into a table of test inputs and verifiable outputs, we are effectively doing **data-driven testing**. Ideally, these tests are automated. However, this might not be always possible or viable and manual tests can take advantage of this methodology as well.

Parameterized tests

Let's start with an example:

Suppose we need to validate the login on a website with a set of valid and invalid user names and passwords:

Username	Password	Valid
admin	123123	valid
john.doe	######	invalid
jane.doe	jane123	valid

The following test cases need to be executed:

#	Action	Data	Expected Result
1	Open the website		The main page is displayed and the user is able to enter login credentials
2	Enter the following login and password and press the Login button	Login: admin Password: 123123	The login is valid.

#	Action	Data	Expected Result
1	Open the website		The main page is displayed and the user is able to enter login credentials
2	Enter the following login and password and press the Login button	Login: john.doe Password: ######	The login is invalid.

#	Action	Data	Expected Result
1	Open the website		The main page is displayed and the user is able to enter login credentials
2	Enter the following login and password and press the Login button	Login: jane.doe Password: jane123	The login is valid.

Instead of creating separate tests, test designers can instead create a single test with the following parameters: *Username*, *Password,* and *Valid.*

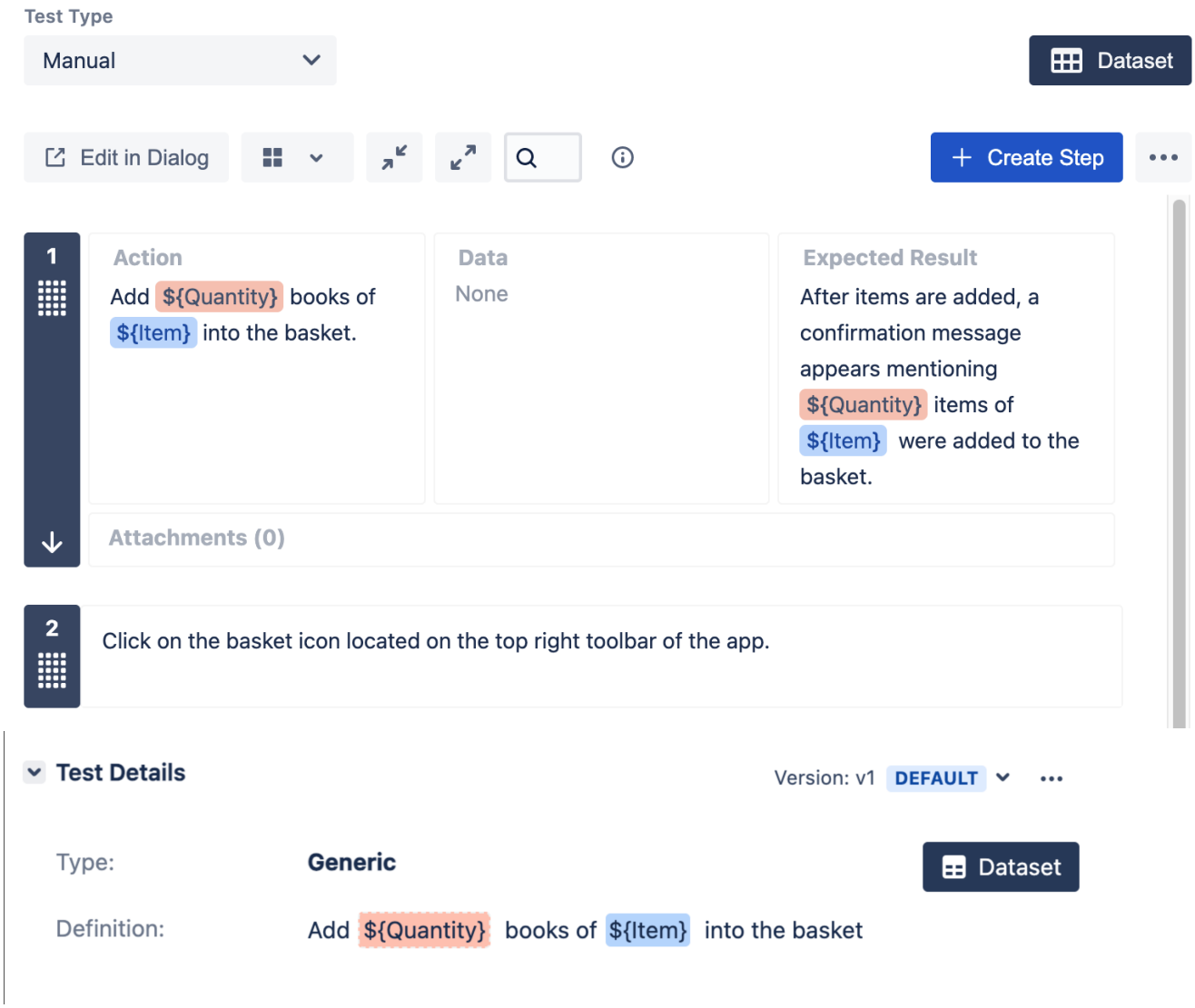
#	Action	Data	Expected Result
1	Open the website		The main page is displayed and the user is able to enter login credentials
2	Enter the following login and password and press the Login button	Login: **${Username}** Password: **${Password}**	The login is **${Valid}**.

Parameterized tests in Xray are defined just like any other test with the addition of some parameter names within the specification.

Parameters are embedded within the test specifications using the following notation: **${PARAMETER_NAME}** for Steps definitions and Unstructured tests, and **<PARAMETER_NAME>** for Gherkin.

This notation is used to reference parameters within the **test definition**. You can reference parameters in any **text-based** test step **custom field, Unstructured** definition, or **Gherkin** Scenarios.

Parameters are defined within datasets. However, it is possible to reference a parameter that is not yet defined (meaning it does not have a corresponding name within the dataset). In this case, the parameter will be highlighted in red.



Datasets

The parameters, along with their values, are defined within a **dataset**. A dataset is a collection of data represented with a tabular view where every column of the table represents a particular variable (or **parameter**), and each row corresponds to a given record (or **iteration**) of the dataset.

The number of rows in the dataset determines the number of iterations to execute. If the dataset contains a single row, there will be a single execution parameterized with the values defined on the dataset row.

Datasets can be defined in different entities and scopes. A dataset can be defined, edited, or simply viewed using the "**Dataset**" button located in each Xray entity or scope.

Parameter types

Parameters can have the following types:

- **Text** - where the parameter value will be set using an open text field.
- **List** - where the parameter value can be selected from a predefined list of options.

List parameters can be created using either **ad hoc** or **predefined** lists.

Ad hoc lists are defined locally for each parameter, while predefined lists are created by administrators at different levels:

- Global - managed by Jira administrators
- Project - managed by project administrators

Global lists can not be used directly. These must be included in the project by the project administrator before they can be accessed within a dataset of that project.

Having predefined lists is useful if the list parameters are commonly used in multiple datasets. If multiple projects make use of the same list, you can also create a global list so that it can be used by different projects. This way you have a central place to manage common parameter lists.

Examples of predefined lists might be:

- Profiles,
- Users,
- Roles,
- Colors,
- Credit card types,
- Addresses,
- ...

When creating a new list parameter using a predefined list, you can choose a list that is available within the current project. The current project is determined by the parent issue where the dataset is defined.

Combinatorial parameters

Combinatorial parameters are special parameters that will be combined with the remaining parameters (combinatorial or seeding parameters) to generate all possible combinations automatically. This prevents users from typing all the combinations when creating a dataset.

S*eeding* parameters are those parameters that describe fixed test cases. The seeding parameters will not be combined among each other, only with combinatorial parameters.

Let's consider the following example:

We created a test to check if we can add books to a shopping cart in our online book store. We have parameters like *Item*, *Price*, *Rating*, *In Stock*, *Condition*, and *Format*. There are certain books we want to test (3 in this case). However, we would like to test all combinations of these books with the following parameters: ***Gift*** and ***Quantity***.

In this case, these will be the seeding parameters:

- *Item*,
- *Price*,
- *Rating*,
- *In Stock*,
- *Condition*,
- *Format*

Because we want to test these items with all the combinations of *Gift* and *Quantity* parameters, we can create these as **Combinatorial** parameters:

- *Gift**
- *Quantity**

Combinatorial parameters are denoted with an asterisc (*) suffix.

Xray will generate all possible combinations upon execution automatically.

Iterations 24

- **Iteration 1 -** In Search of Lost Time | $34 | 5 | Marcel Proust | Yes | New | Paperback | Yes | 1
- **Iteration 2 -** One Hundred Years of Solitude | $20 | 4.9 | Gabriel Garcia Marquez | Yes | Used | Paperback | Yes | 1
- **Iteration 3 -** The Great Gatsby | $39 | 4.7 | F. Scott Fitzgerald | No | New | Kindle | Yes | 1
- **Iteration 4 -** In Search of Lost Time | $34 | 5 | Marcel Proust | Yes | New | Paperback | No | 1
- **Iteration 5 -** One Hundred Years of Solitude | $20 | 4.9 | Gabriel Garcia Marquez | Yes | Used | Paperback | No | 1
- **Iteration 6 -** The Great Gatsby | $39 | 4.7 | F. Scott Fitzgerald | No | New | Kindle | No | 1
- **Iteration 7 -** In Search of Lost Time | $34 | 5 | Marcel Proust | Yes | New | Paperback | Yes | 2
- **Iteration 8 -** One Hundred Years of Solitude | $20 | 4.9 | Gabriel Garcia Marquez | Yes | Used | Paperback | Yes | 2
- **Iteration 9 -** The Great Gatsby | $39 | 4.7 | F. Scott Fitzgerald | No | New | Kindle | Yes | 2
- **Iteration 10 -** In Search of Lost Time | $34 | 5 | Marcel Proust | Yes | New | Paperback | No | 2
- **Iteration 11 -** One Hundred Years of Solitude | $20 | 4.9 | Gabriel Garcia Marquez | Yes | Used | Paperback | No | 2
- **Iteration 12 -** The Great Gatsby | $39 | 4.7 | F. Scott Fitzgerald | No | New | Kindle | No | 2
- **Iteration 13 -** In Search of Lost Time | $34 | 5 | Marcel Proust | Yes | New | Paperback | Yes | 5

Dataset scopes

A dataset can be defined in the following entities/scopes:

1. Test (default dataset)
2. Test Plan - Test
3. Test Execution - Test (Test Run)

The closest dataset to the test run will be one used to generate the iterations, effectively overriding any dataset defined in higher levels:

Test Execution - Test (Test Run) > Test Plan - Test > Test (default)

The test dataset is the default. If there is the need to override or change this dataset you can do this at the planning or execution phases.

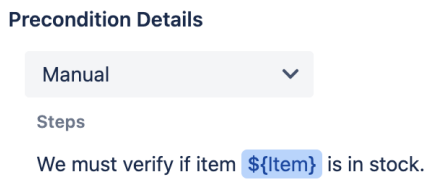
Dataset limits

- Maximum number of #**iterations** per dataset: **1000**
- Maximum Number of #**parameters** per dataset: **20**
- A dataset can not contain duplicate rows.

Parameterized preconditions

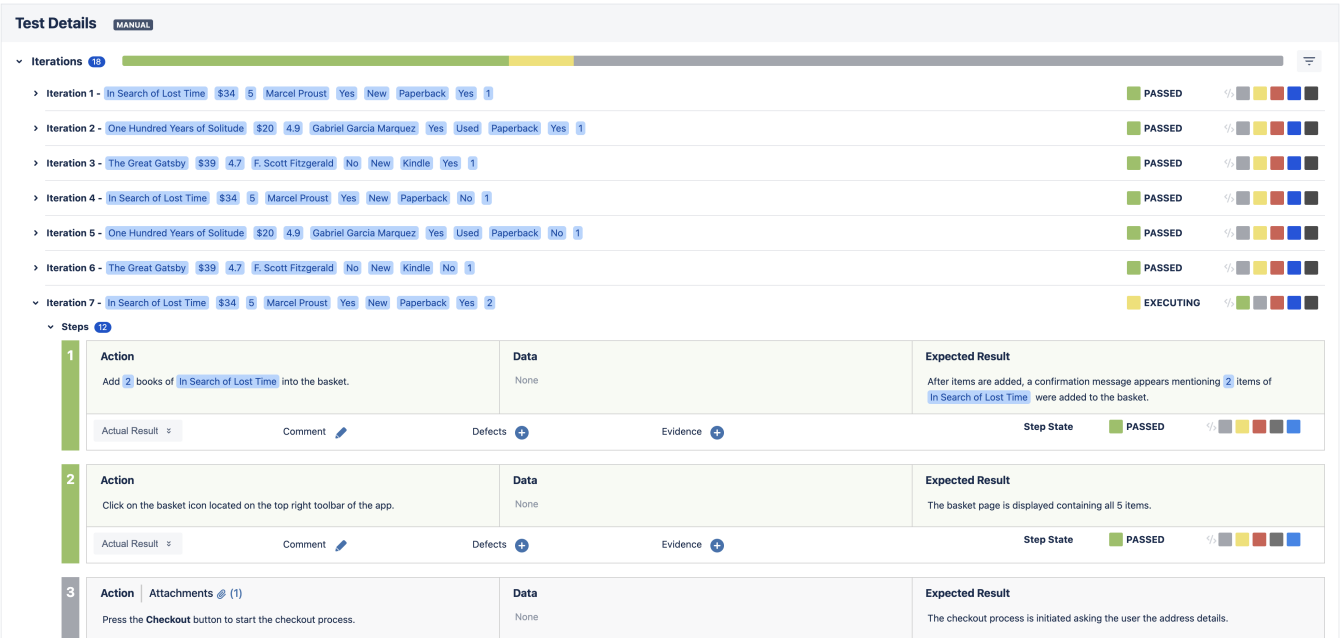
Precondition issues can also be parameterized by including parameter names in the precondition specification.

The parameters will be unfolded on the execution screen, just like Test cases. For this, the dataset must have the same parameters, matched by name.



Execution

All iterations for a given test are executed within the context of the same test run. Each iteration can be expanded and the steps executed individually. The step parameters will be replaced by the corresponding iteration values. The steps affect the iteration status that, in turn, affects the overall test run status.



Please check the Execution page for more details.

Parameterized tests vs test environments

Although test environments can be thought of as a particular case of parameterized tests (given that environments can be considered parameters of test cases), datasets might not be the best choice for environmental variables that are not embedded in the test specification (e.g. manual test steps).

As such, it is not good practice to use datasets with environment variables such as *Browsers*, *OSs*, *Databases, or Devices*. This is because:

- Test engineers often configure a specific setup (or environment) to execute a set of tests. In this case, test engineers might want to execute the tests oriented to the environment and not the test case. This means they execute all the test cases first for one environment, and then for another one, and so on. This way, having all these environments within the same test run is not ideal as users would have to jump from one test run to the other without finishing the execution.
- Usually, test environments are independent variables or dimensions. For instance, let's consider the following test environment: *Browser*, *Databas e,* and *OS*. If a test case fails only in a specific browser, once the bug is fixed, we might not want to re-execute the same test for all databases and OS's as we are confident that the change did not affect these variables. With datasets, if one iteration fails, you need to re-execute all the iterations again (if you use a different test run, of course).
- Xray does not provide reports based on parameters.

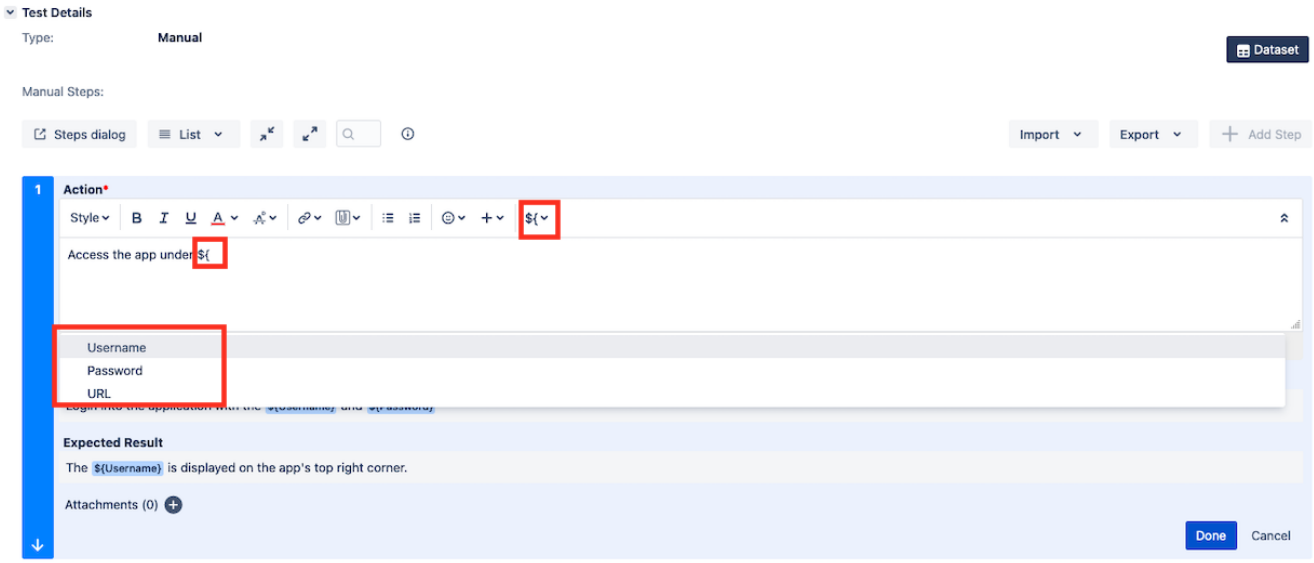
In conclusion, if your parameters are environment variables that do not need to be included in the test specification, don't use datasets. Use test environments instead.

> **Test environments improvements**
>
> We have plans to improve how test environments are managed in Xray. It will be possible to specify or generate combinations of environments for different variables.
>
> Please check our roadmap.

Defining a parameterized test

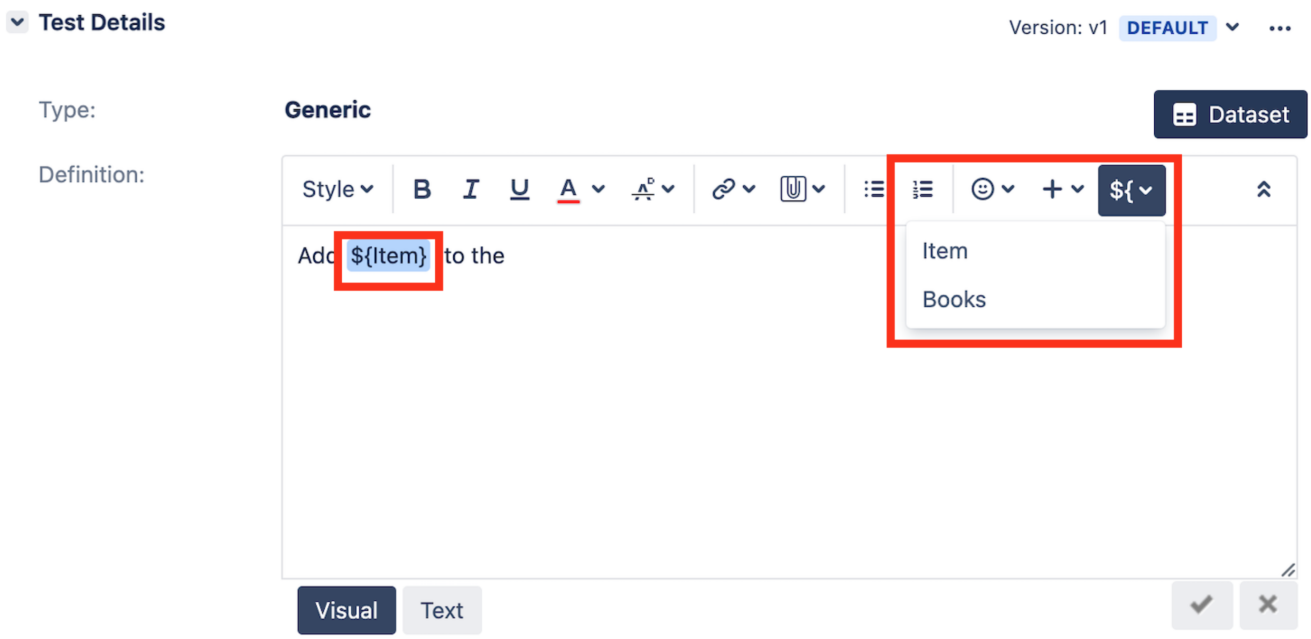
Manual Steps

1. Create or edit a given step using either the inline view within the test issue or the steps dialog.
2. When specifying a step, to reference a parameter you have two options:
    a. Start typing **${**. If there is a default dataset defined on the test, you should see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be placed with the text.
    b. Use the toolbar button **${**. After pressing this button, and if there is a default dataset defined on the test, you should see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be placed on the cursor position.



Generic or Unstructured definition

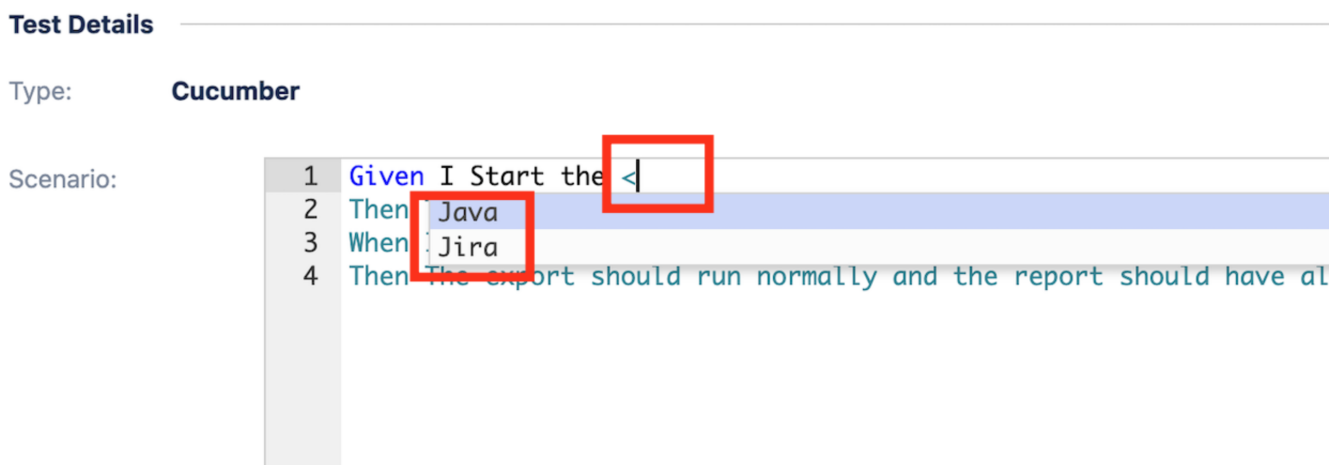
1. Edit the definition using the inline view within the test issue.
2. To reference a parameter, you have two options:
    a. Start typing **${**. If a default dataset is defined on the test, you should see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be placed with the text.
    b. Use the toolbar button **${**. After pressing this button, and if there is a default dataset defined on the test, you should see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be placed on the cursor position.



> To define parameters on unstructured tests, the Jira markup wiki language must be activated in the **Generic Test Definition** custom field.
>
> To enable the Rich Text Editor, you have to set the renderer of the definition custom field to Wiki Style Renderer.

Gherkin Scenarios

1. Create or edit a given step using either the inline view within the test issue or the steps dialog.
2. When specifying a Scenario, to reference a parameter you should start typing **<**. If there is a default dataset defined on the test, you should see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be placed with the text.



Creating a dataset

In order to create or edit the default dataset (within a test):

1. Press the **"Dataset"** button. This will open the dataset dialog. Here, you can define a dataset by creating parameters and adding values to them.

2. Creating parameters:

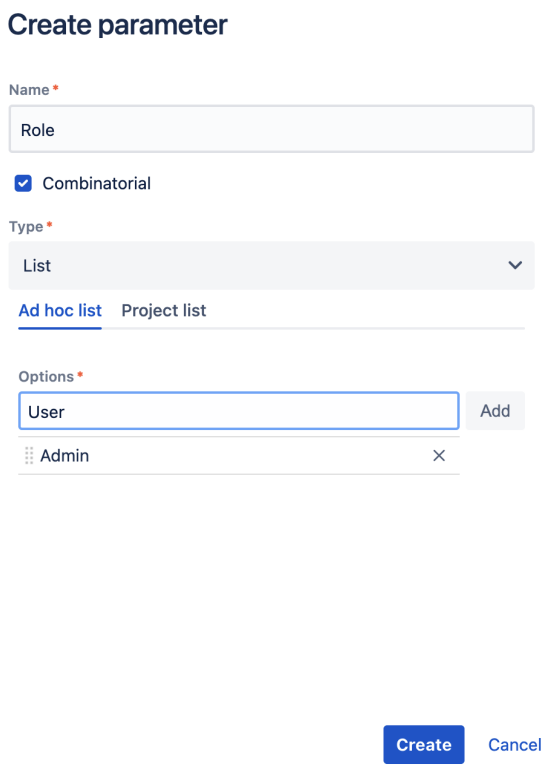
2.1. Press the **"Create parameter"** button located on the dataset main toolbar or on the empty view. This will open a small dialog to specify the parameter attributes.

2.2. Specify the name. Parameter names must start with a letter or underscore and can only contain letters, numbers, a space between words, "_", "-" and a max of 64 characters.

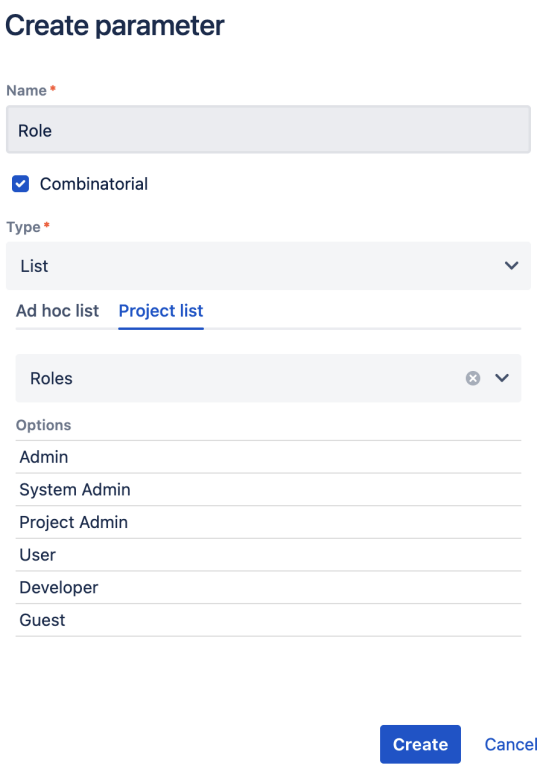
2.3. Check the **Combinatorial** checkbox if you are creating a combinatorial parameter.

2.4. Choose the parameter type: **Text or List**. If the parameter type is a List you can:

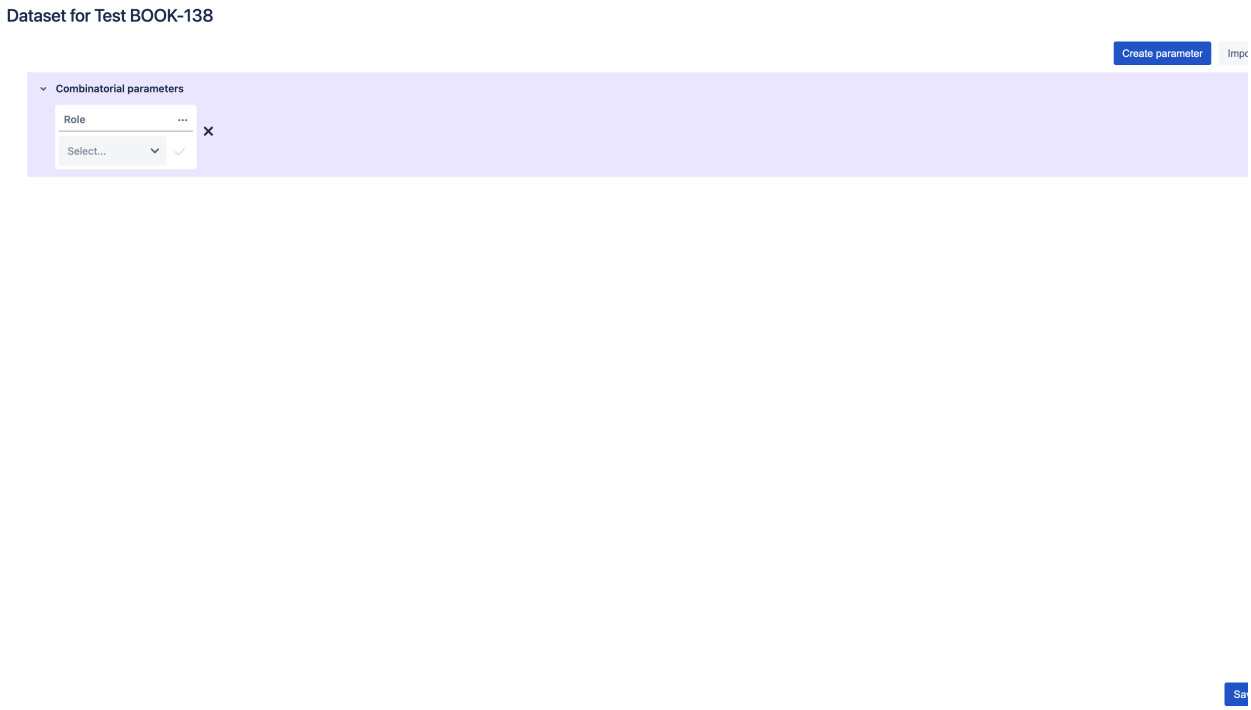
a. create an **ad hoc** list just for this parameter. You need to specify the values for the list.



b. use a project **predefined list**.



2.5. Press **"Save"** to create the parameter. The parameter must be placed on the dataset.

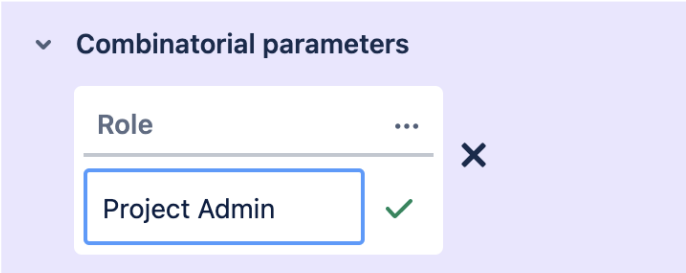


3. Adding combinatorial parameter values:

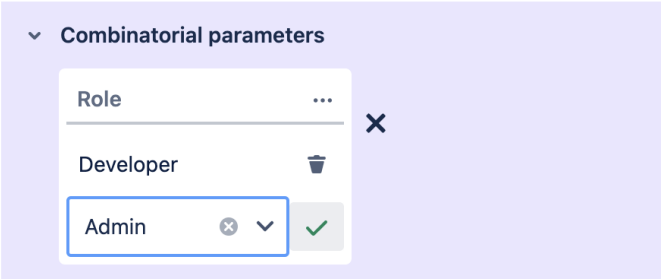
Once you have at least one parameter, you can start filling their values and adding new iterations.

A placeholder is provided within each combinatorial parameter. In order to add new values to combinatorial parameters:

3.1. For **text** parameters, just type the value and press the **check** button next to the field.



3.2. For **list** parameters, select an option and press the **check** button next to the field.



4. Adding rows (filling the parameter values):

4.1. Once you have non-combinatorial parameters, an empty placeholder row will appear so that the parameters can be populated for the default iteration.

Editing parameter values is as simple as editing their corresponding cells. The values will be kept when the cell loses focus.

You can navigate between cells of the same row and also between rows using the keyboard: TAB (forward), SHIFT+TAB (backward).

#	Role	User	Email	Password
1	Admin	admin	admin@getxray.com	123123
2	User	john.doe	john.doe@getxray.com	XXXX

New

4.2. To create new rows, you can press the **"New"** button below the table or navigate using the keyboard from the last row (a new row will be created automatically by navigating forward on the last cell of the last row).

There are a total of 3 iterations to execute.

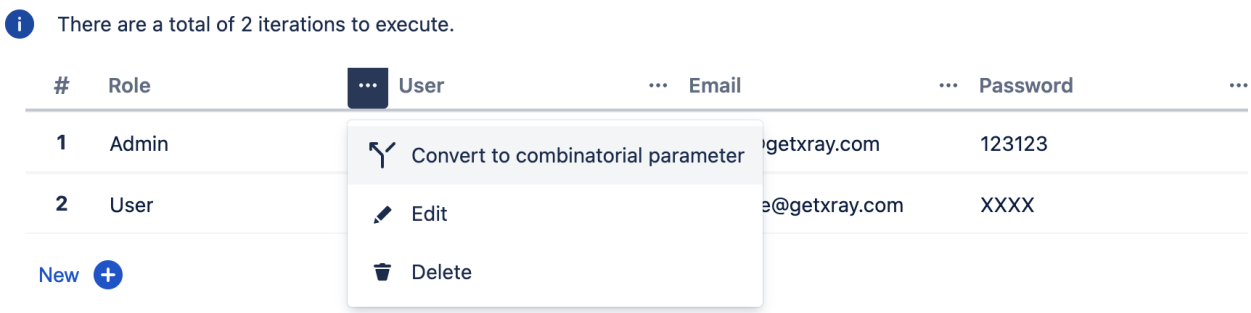
#	Role	User	Email	Password
1	Admin	admin	admin@getxray.com	123123
2	User	john.doe	john.doe@getxray.com	XXXX
3	None	None	None	None

New

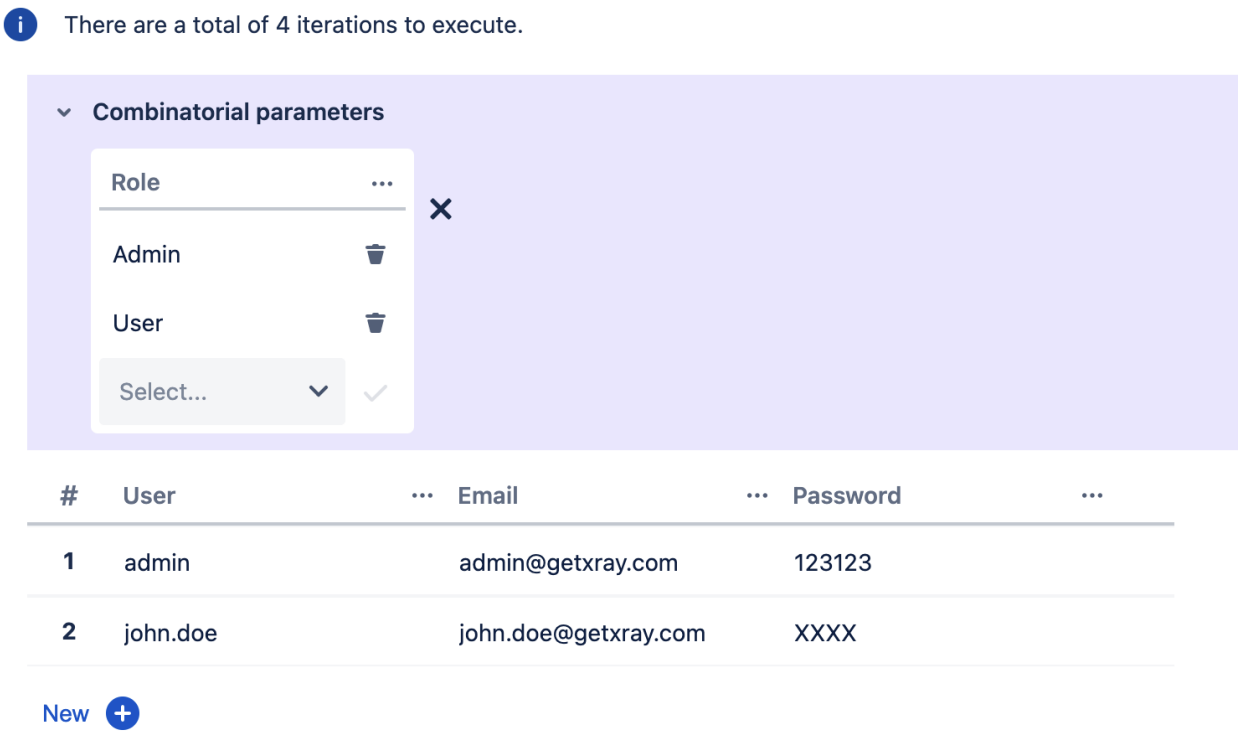
5. Converting a seeding parameter to a combinatorial parameter:

It is possible to convert an existing parameter to a combinatorial parameter. This will remove the parameter column from the seeding parameters table and group all remaining rows automatically. A new combinatorial parameter will be created by grouping all the values.

5.1. Next to each column, there is an options button. Clicking this button will reveal a menu with options for *Editing*, *Deleting,* and *Converting* parameters.



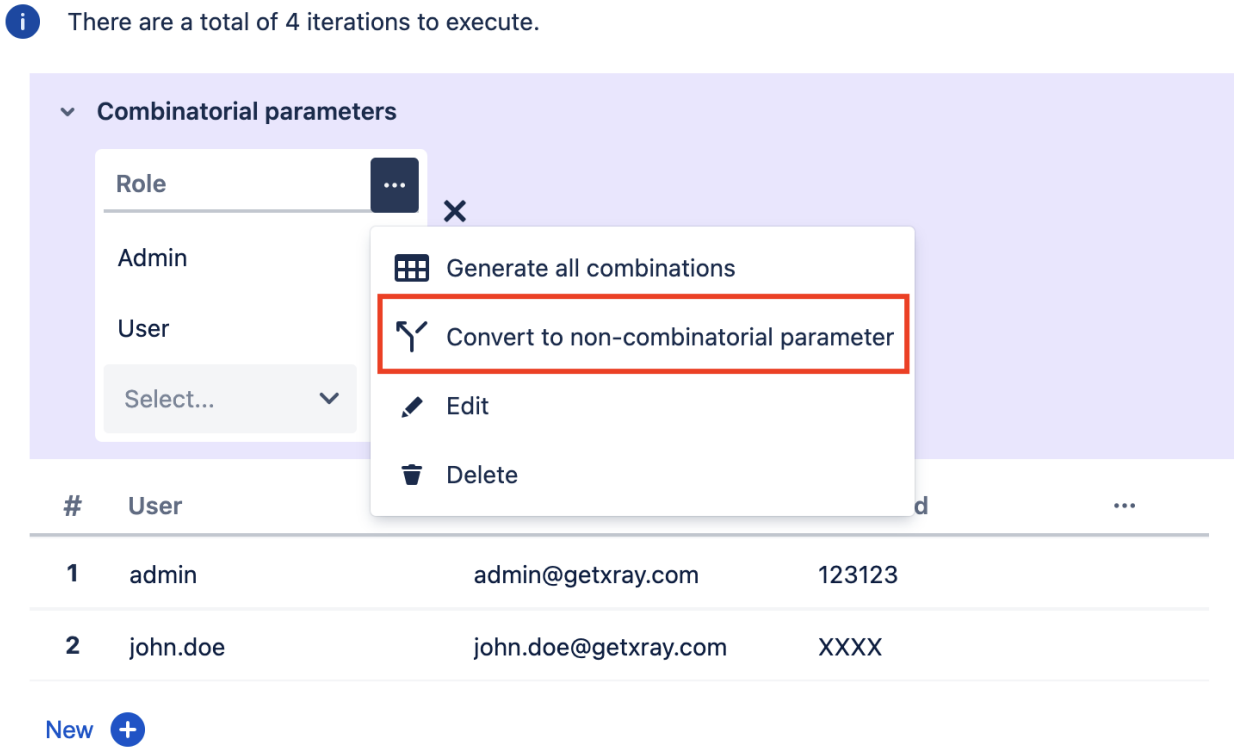
5.2. Click on **"Convert to combinatorial parameter."**



6. Converting a combinatorial parameter to a seeding parameter:

It is also possible to convert a combinatorial parameter back into a seeding parameter.

6.1. Next to each combinatorial column, there is an options button. Clicking this button will reveal a menu with options for *Editing*, *Deleting,* and *Converting* parameters.

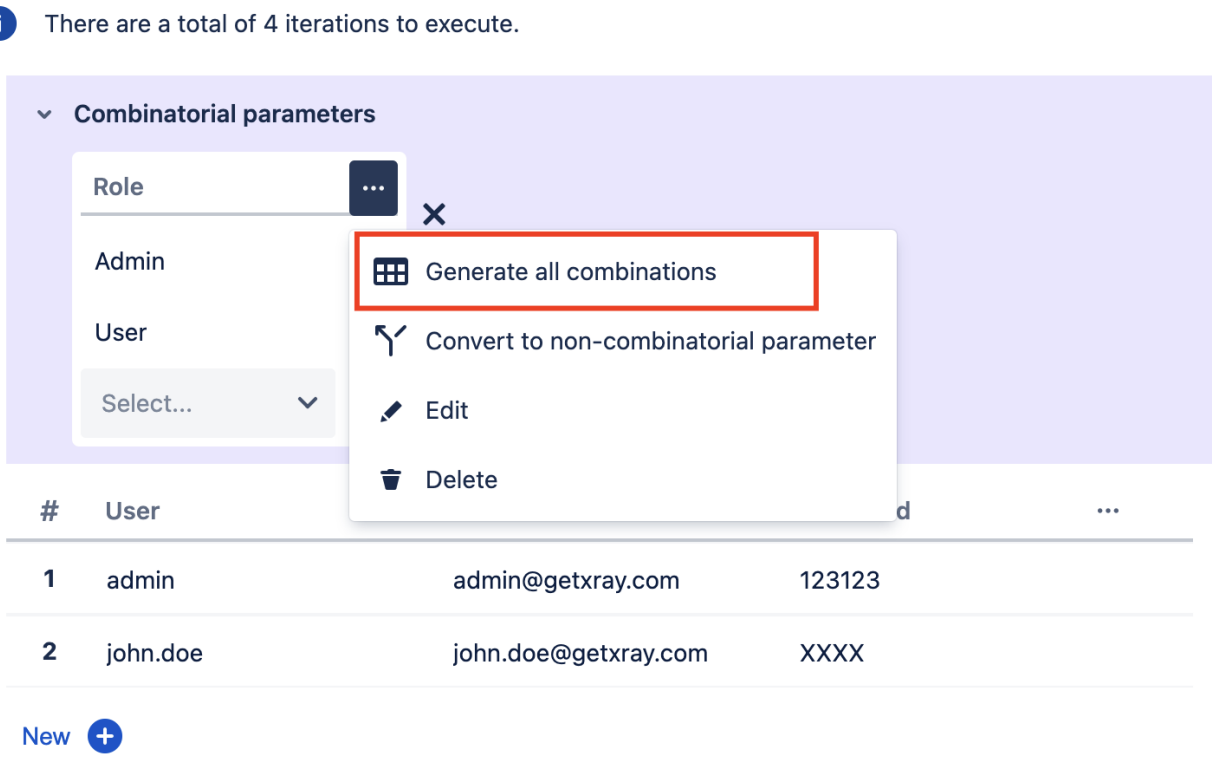


6.2. Click on the **"Convert to non-combinatorial parameter"**.

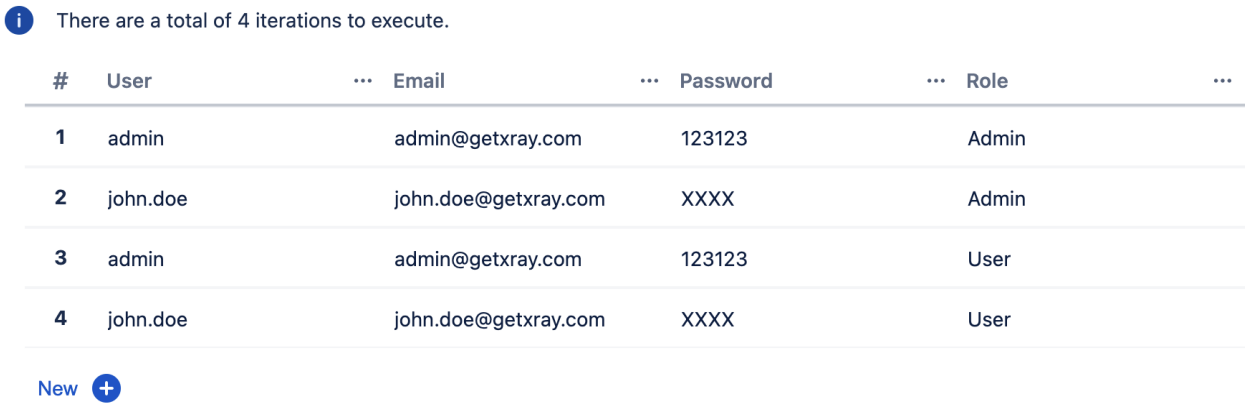
7. Generating all combinations:

You don't need to generate all the combinations for a given dataset in order to execute all iterations. Xray will do this automatically for all of the combinatorial parameters. However, sometimes we don't need all the combinations. In order to do this, we can generate all combinations and remove some iterations afterward.

7.1. Next to each combinatorial column, there is an **options** button. Clicking this button will reveal a menu with options for *Editing*, *Deleting, C onverting* parameters, and *Generating all combinations*.



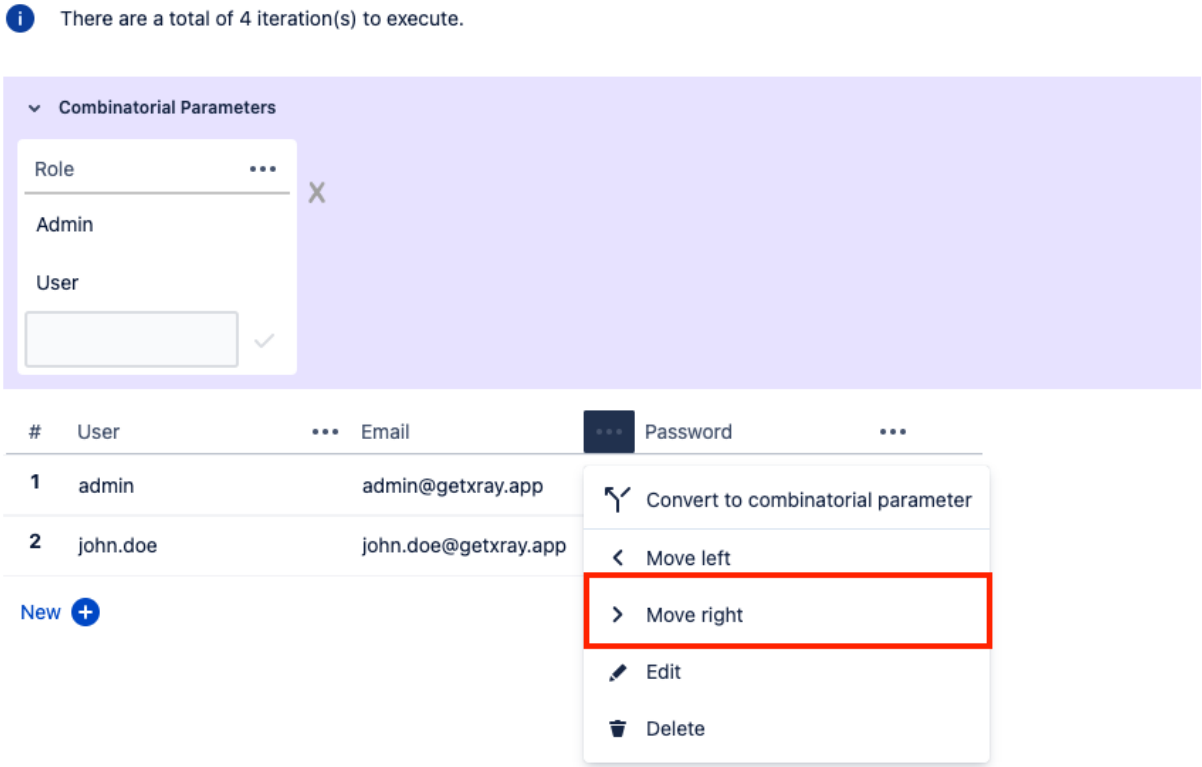
7.2. Click on the **"Generate all combinations."** A confirmation dialog will appear. Confirming the changes will apply the cartesian product between the combinational parameter values and the seeding parameter rows.



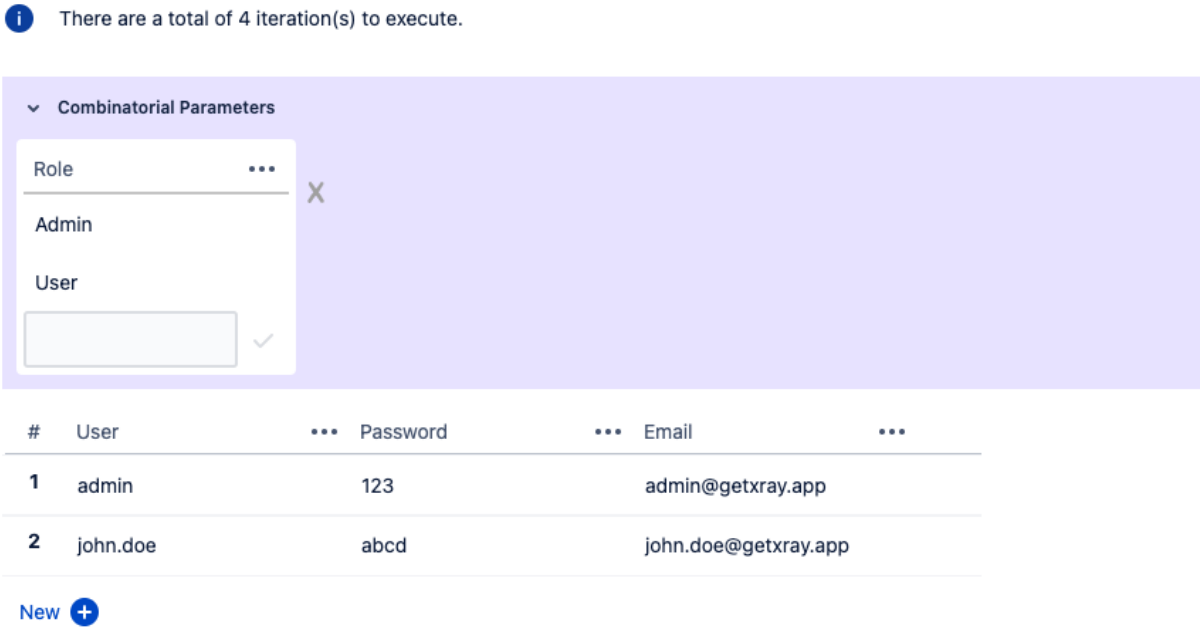
8. Move left/right:

You can re-order any parameters in your dataset, by selecting the "Move left/right" option.

8.1. Open the options menu in either a Combinatorial or non-combinatorial (seeding) parameter, you will have a "Move left/right" option available. Notice that if the parameter is already place on the most outer left (right) position, the "Move left" (right) option is not available.



8.2. Select the move left/right option, This will swap the positions of the parameters.



9. After having performed all changes in the dataset, you need to press the **"Save"** button in order to persist the dataset into the database.

Importing a dataset from a CSV file

Besides defining a dataset by creating parameters and setting their values directly using the Xray UI, it is also possible to import an existing dataset from a CSV file.

In order to import a CSV file:

1. Open the dataset dialog.

2. Choose the **"Import from CSV"** option available on the Import dropdown button (located on the empty state view or on the dataset main button toolbar). A dialog will appear.

3. Select and fill the following options:

   - File - find a local CSV file with the desired dataset

   - CSV delimiter - defaults to "," but you can choose any other character

   - File encoding - defaults to UTF-8

   - Overwrite existing parameter values - if checked, this option will delete any values from existing parameters

   - Create new parameters - If checked, it will create non-existing parameters automatically, based on the CSV column name. Otherwise, it will just append/update the values on the existing parameters.

4. Press **"Import"** to import the external dataset.

> **Combinational parameters**
>
> You can also import combinational parameters. Any parameter with an **asterisk suffix** will be considered a combinational parameter. E.g., Quality*. In this case, the "Create new parameters" option must be enabled.

Exporting a dataset to a CSV file

To export a dataset to a CSV file:

1. Open the Dataset dialog.

2. Click the **options** button next to the import button. Clicking this button will reveal a menu with options for *Export to CSV* and Delete.

3. Click the **Export to CSV**. A file should be downloaded.

> **Export CSV**
>
> The delimiter used to generate the CSV file is a **comma**.
>
> The combinatorial parameters will be exported with an **asterisk** as a suffix.

Overriding a dataset in a test plan or test execution

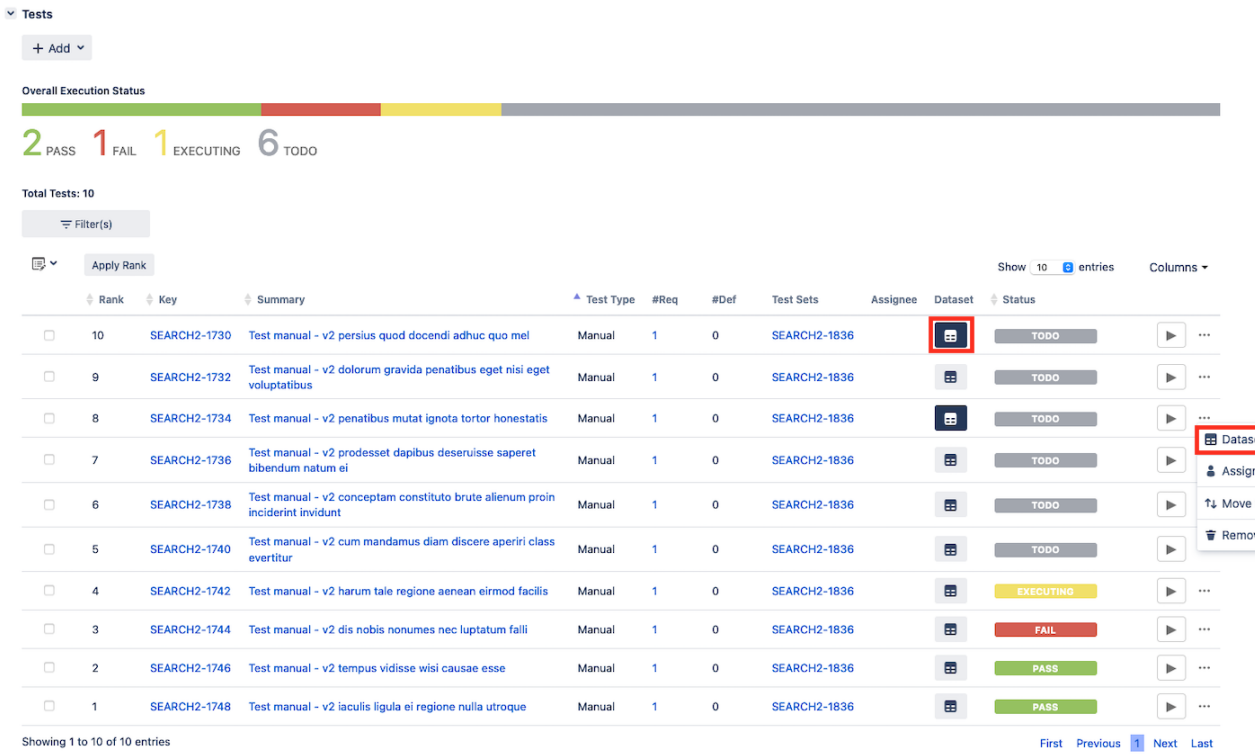
To define or override a dataset at the test plan or test execution level:

1. Navigate to the desired test plan or test execution issue

2. A menu action is provided on each test row. A new **column** named "Dataset" is also available by configuring the column layout for the Test Plan or Test Execution datatable. If there is a dataset already defined at this level, the dataset button will be displayed with a *selected* style.

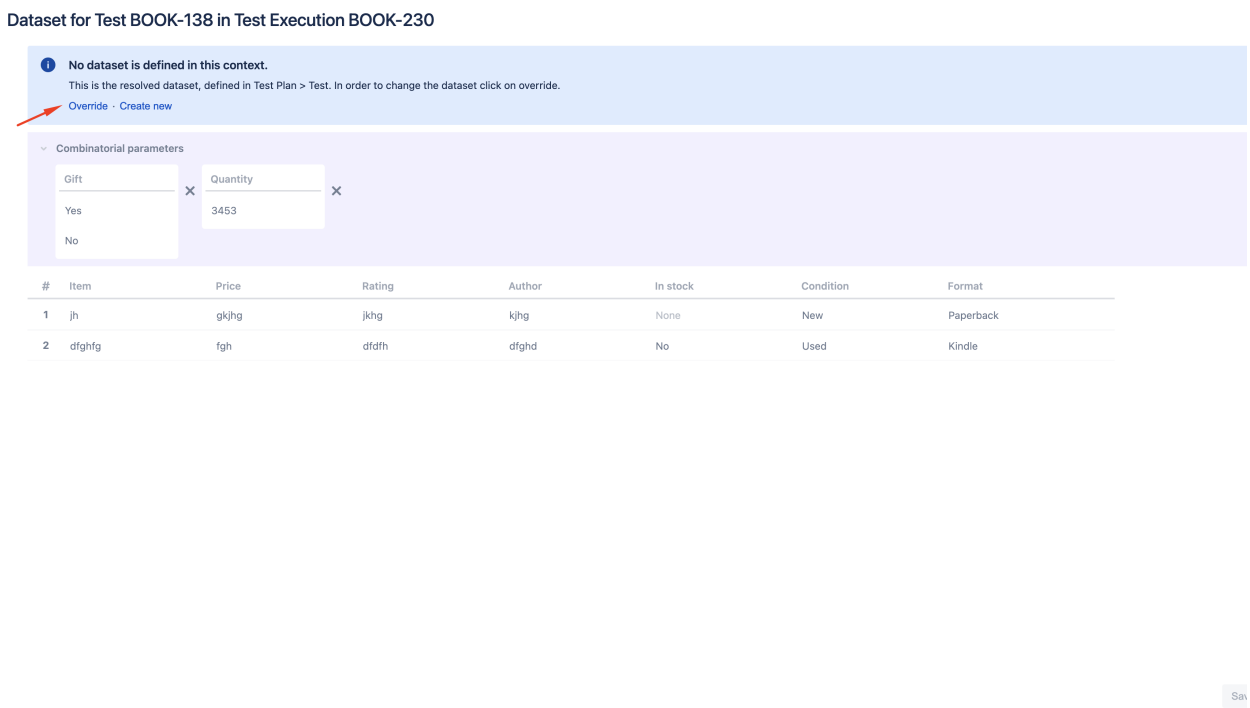
    2.1. Open the dataset dialog by pressing the **"Dataset"** button in the column or

    2.2. Open the dataset dialog by selecting the **"Dataset"** action from the test row options menu.



3. The dataset dialog is opened.

   3.1. If there is a dataset defined on a parent level, it is possible to Override the parent dataset and modify its values.



   3.2. Otherwise, you can start defining a new dataset at this level.

4. Once the dataset is defined (or overridden) you need to press **"Save"** to persist the changes to the database.