

Parameterized Tests

Table of Contents

- [Introduction](#)
 - [Parameterized Tests vs Test Environments](#)
- [Working with Parameterized Tests](#)
 - [Example](#)
 - [Parameterized Tests in Xray](#)
 - [Defining a Parameterized Test](#)
 - [Datasets](#)
 - [Parameter Types](#)
 - [Ad Hoc Lists](#)
 - [Predefined Lists](#)
 - [Combinatorial Parameters](#)
 - [Example](#)
 - [Dataset Scopes](#)
 - [Dataset Limits](#)
 - [Creating a Dataset](#)
 - [More Operations](#)
 - [Importing a Dataset from a CSV File](#)
 - [Exporting a Dataset to a CSV File](#)
 - [Deleting a Dataset](#)
 - [Parameterized Preconditions](#)
 - [Iterations Execution](#)
 - [Overriding a Dataset in a Test Plan or Test Execution](#)

Introduction

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, making the Test static and difficult to modify. Static Tests lead to redundancy since you need to define and clone the same Test for different combinations of values to cover various scenarios or variations of the same Test.

By extracting the variability of the Test specification into a table of test inputs and verifiable outputs, you effectively [data-driven testing](#). Ideally, these Tests are [automated](#). However, this might not always be possible or viable, and manual Tests can also benefit from this methodology.

Parameterized Tests vs Test Environments

Although Test environments can be considered a particular case of parameterized Tests (since environments can be seen as 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](#)).

It is not good practice to use datasets with environment variables such as browsers, operating systems (OSs), databases, or devices. This is because:

1. Test engineers often configure a specific setup (or environment) to execute a set of Tests. In this case, they might want to execute the Tests oriented to the environment rather than the Test case. This means executing all the Test cases first for one environment, then for another, and so on. Having all these environments within the same [Test run](#) is not ideal as users would have to jump from one Test run to another without finishing the execution.
2. Test environments are usually independent variables or dimensions. For instance, consider the following Test environment: browser, database, and OS. If a Test case fails only in a specific browser, once the bug is fixed, you might not want to re-execute the same Test for all databases and OSs, as you 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 (assuming 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. Instead, use test environments.



We [plan](#) to improve how test environments are managed in Xray, making it possible to specify or generate combinations of environments for different variables.

Working with Parameterized Tests

Example

Suppose you need to validate the login on a website with a set of valid and invalid usernames 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	N/A	The main page is displayed and the user can enter login credentials
2	Enter the following login and password, and click the <i>Login</i> button	Login: admin Password: 123123	The login is valid

#	Action	Data	Expected Result
1	Open the website	N/A	The main page is displayed and the user can enter login credentials
2	Enter the following login and password, and click the <i>Login</i> button	Login: john.doe Password: #####	The login is invalid

#	Action	Data	Expected Result
1	Open the website	N/A	The main page is displayed and the user can enter login credentials
2	Enter the following login and password, and click the <i>Login</i> 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	N/A	The main page is displayed and the user can enter login credentials
2	Enter the following login and password, and click the <i>Login</i> button	Login: \${Username} Password: \${Password}	The login is \${Valid} .

Parameterized Tests in Xray

Parameterized Tests in Xray (Figure 1) are defined just like any other Test with the addition of some parameter names within the specification.

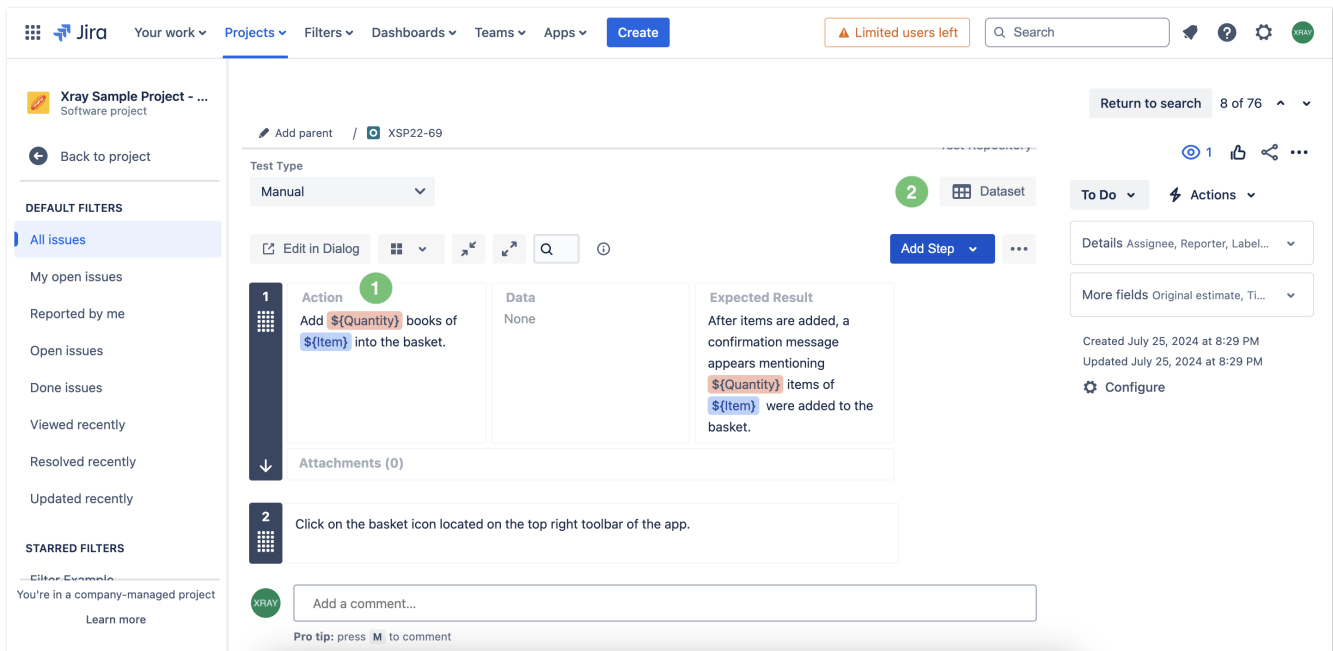


Figure 1 - Parameterized tests

Parameters are embedded within the Test specifications using the following notation: `${PARAMETER_NAME}`.

Parameter names are case-sensitive.

This notation (Figure 1 - 1) is used to reference parameters within the **Test Steps**. You can reference parameters in the **Action**, **Data**, **Expected Result** fields and any text-based custom fields in the Test Steps.

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 (Figure 1 - 1).

Defining a Parameterized Test

To reference parameters within Test steps:

1

Create or edit a **Test Step** using either the inline view within the Test Issue or the Steps modal.

2

When specifying a Test Step, to reference a parameter you have two options:

- Start typing `${` (Figure 2 - 1). If there is a default dataset defined on the Test, you will see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be inserted into the text.
- Use the toolbar button `${` (Figure 2 - 2). After pressing this button, and if there is a default dataset defined on the test, you will see a list of the available parameters. Choose the desired parameter using the cursor keys or mouse. The parameter will be placed at the cursor position.

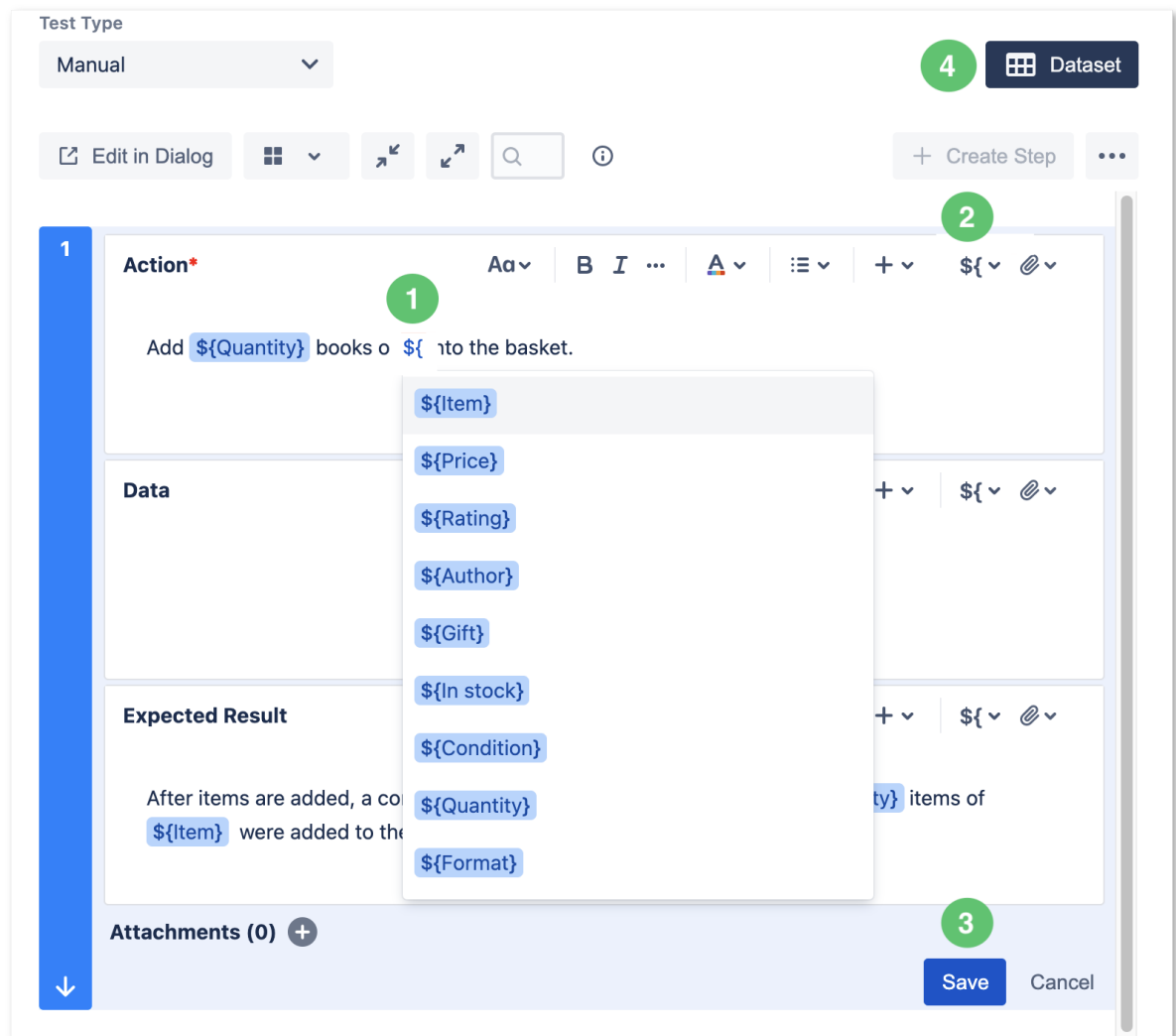


Figure 2 - Defining parameters

3

Once you're finished, click the **Save** button (Figure 2 - 3).

Datasets

Parameters and their values are defined within a dataset. A dataset is a collection of data represented in 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 in 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 (Figure 2 - 4) located in each Xray entity or scope.

Parameter Types

Parameters can have the following types:

- **Text** - the parameter value will be set using an open text field.
- **List** - the parameter value can be selected from a predefined list of options. They can be created using either **ad hoc** or **predefined** lists.

Ad Hoc 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. 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.
- **Project** - managed by project administrators.

Predefined Lists

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 include:

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

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.

Seeding parameters are those parameters that describe fixed Test cases. The seeding parameters will not be combined with each other; they will only be combined with combinatorial parameters.

Example

This is a test to see if you can add books to a shopping cart in your online bookstore. The parameters are: *Item*, *Price*, *Rating*, *In Stock*, *Condition*, and *Format*. There are certain books to be tested (three in this case). However, we will test all combinations of these books with the following parameters: *Gift* and *Quantity*.

In this case, these will be the seeding parameters (Figure 3 - 1):

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

Since we want to test these items with all the combinations of *Gift* and *Quantity* parameters, we can create these as combinatorial parameters (Figure 3 - 2):

- *Gift**
- *Quantity**

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

Dataset for Test BOOK-138

1 There are a total of 24 iterations to execute. [Create parameter](#) Import ▾ ⋮

2

Combinatorial parameters

Gift ... X Quantity ... X

Yes ▮
No ▮
Select... ▾ ✓

1

#	Item	Price	Rating	Author	In stock	Condition	Format	
1	In Search of Lost Time	\$34	5	Marcel Proust	Yes	New	Paperback	
2	One Hundred Years of Soli...	\$20	4.9	Gabriel Garcia M	Yes	Used	Paperback	
3	The Great Gatsby	\$39	4.7	F. Scott Fitzgerald	No	New	Kindle	

New +

Figure 3 - Parameters

Xray will generate all possible combinations upon execution automatically (Figure 4).

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 3

Figure 4 - Combinations

Dataset Scopes

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

- 1. Test (default dataset). If there is the need to override or change this dataset, you can do this at the [Test planning](#) or [Test execution](#) phases.
- 2. Test Plan - Test.
- 3. Test Execution - Test (Test Run).

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

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

To define/see the dataset scopes, click the *Dataset* icon (Figure 5 - 1) and/or the *Actions* button (Figure 5 - 2), and then select the *Dataset* option (Figure 5 - 3).

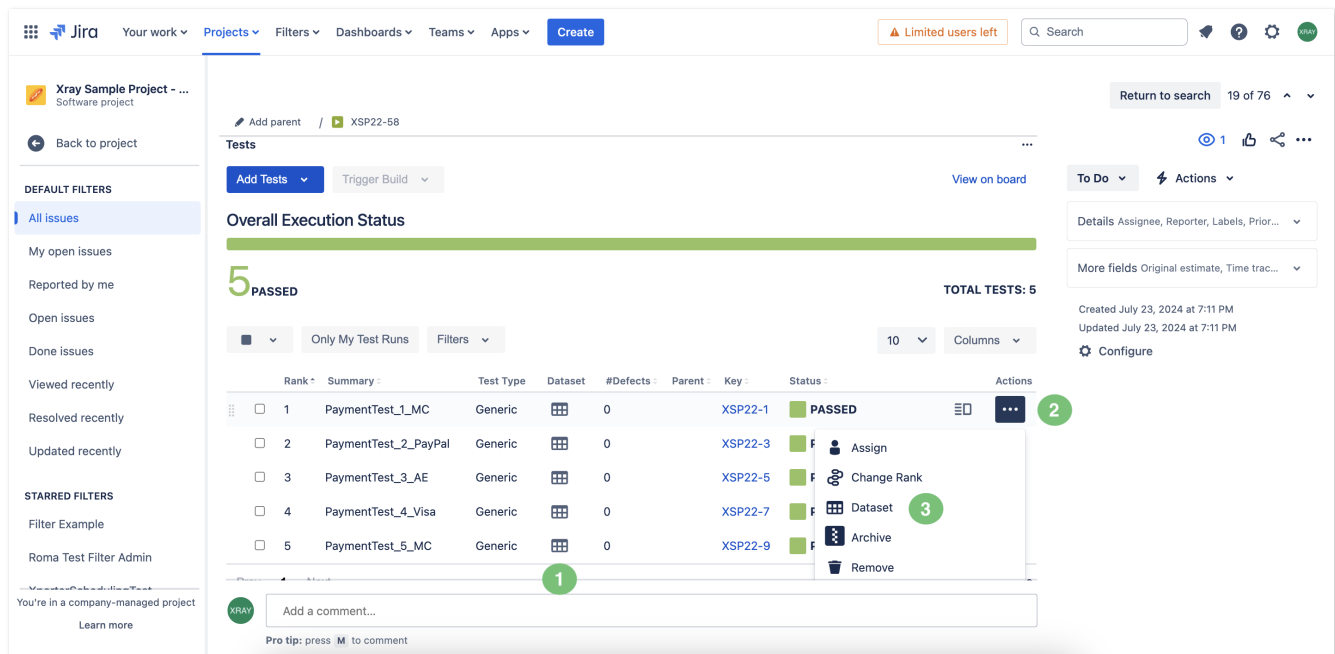


Figure 5 - Scopes

Dataset Limits

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

Creating a Dataset

To create or edit the default dataset (within a Test):

- 1 Click the *Dataset* button (Figure 6 - 1). A modal will open (Figure 7).

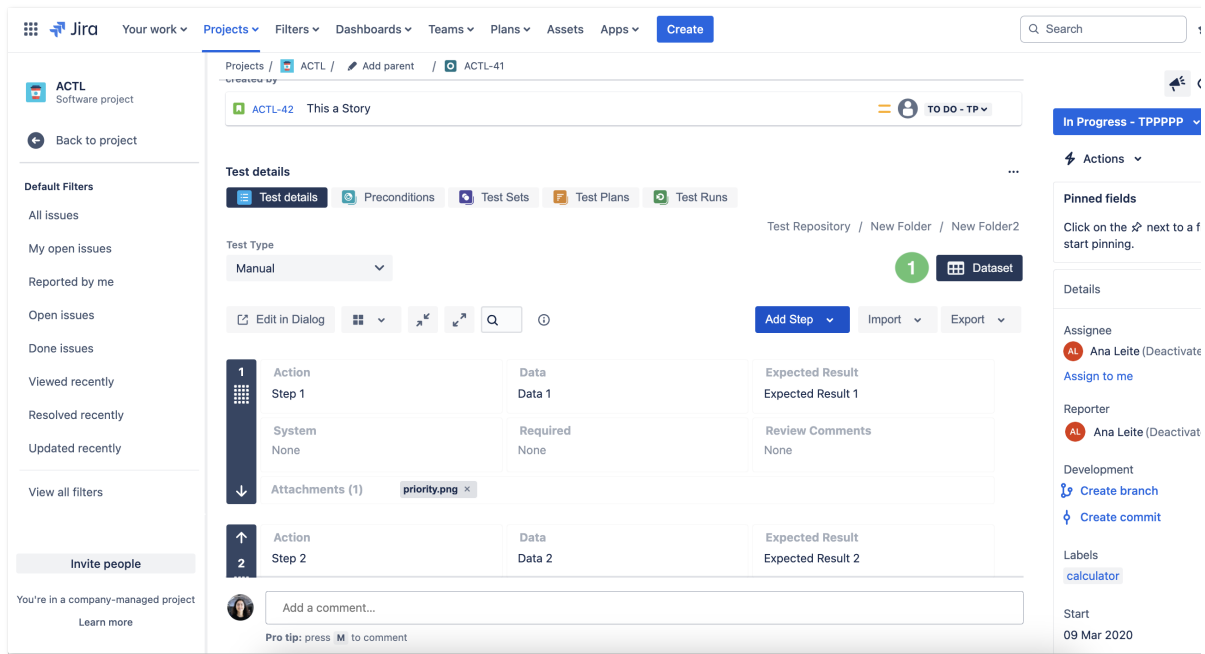


Figure 6 - Dataset

2

Click the *Create parameter* button (Figure 7 - 1). A modal will open for you to specify parameter attributes (Figure 8).

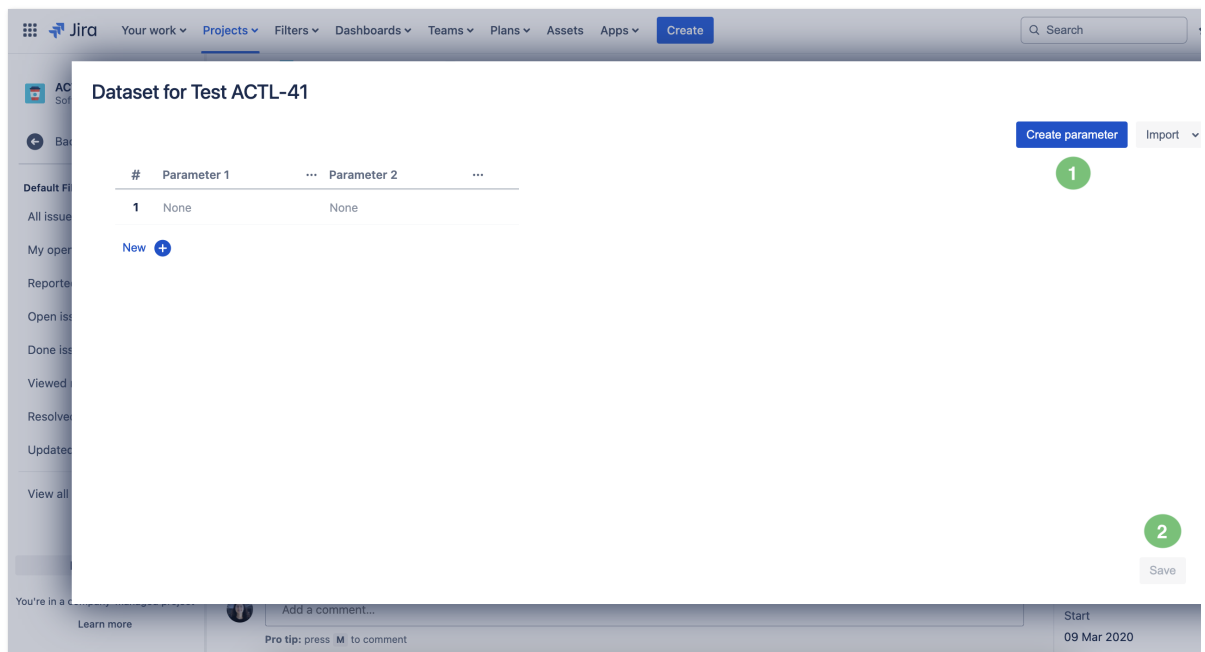


Figure 7 - Parameter

3

Here (Figure 8), you can:

- Specify the parameter's name (Figure 8 - 1; mandatory field). Parameter names must start with a letter or underscore and can only contain letters, numbers, a space between words, "_", "-", and a maximum of 64 characters.
- Check the *Combinatorial* checkbox if you are creating a combinatorial parameter (Figure 8 - 2).
- Choose the parameter type: *Text* or *List* (Figure 8 - 3). If the parameter type is a *List*, you can:
 - Create an *Ad hoc list* just for this parameter (Figure 8 - 4). You need to specify the values for the list (Figure 8 - 5; mandatory field).

- Use a *Project* predefined list (Figure 8 - 4).
- Once you're finished, click *Create* (Figure 8 - 6). You will be redirected to the dataset modal (Figure 9).

Create parameter

Name * 1

Role

☒ Combinatorial 2

Type * 3

List

Ad hoc list Project list 4

Options * 5

User Add

Admin X

6

Create Cancel

Figure 8 - Parameter

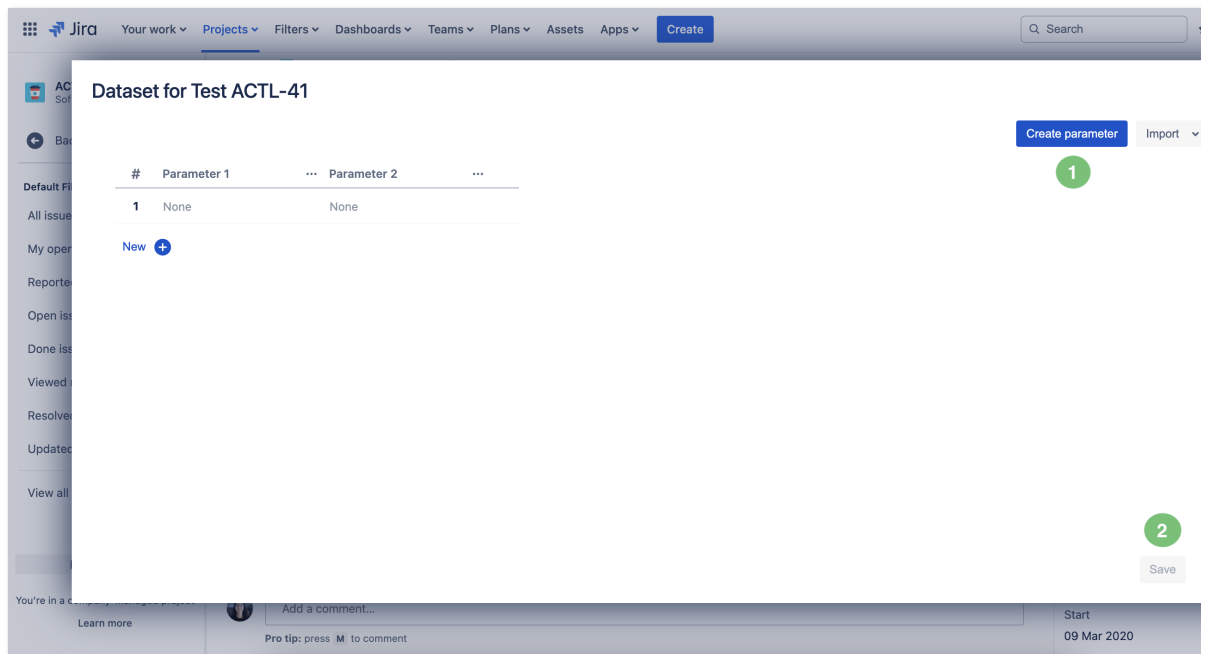


Figure 9 - Modal

More Operations

Adding Combinatorial Parameter Values

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

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

For **text** parameters, type the value and click the *check* icon (Figure 10 - 1).

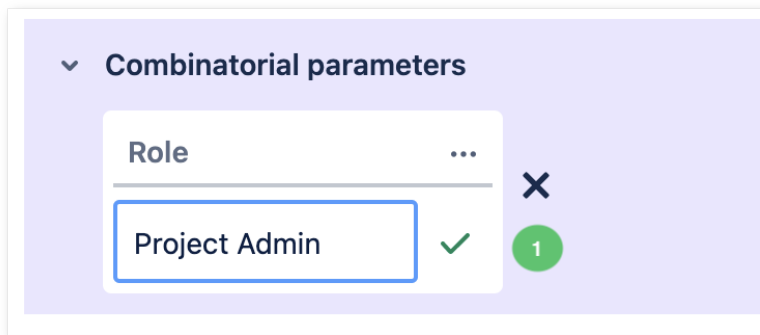


Figure 10 - Text

For **list** parameters, select an option and click the *check* icon (Figure 11 - 1).

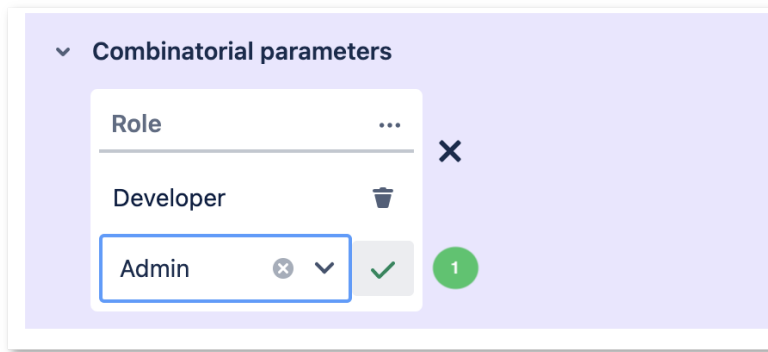


Figure 11 - List

Adding Rows (Filling the Parameter Values)

Once you have non-combinatorial parameters, an empty placeholder row will appear so that the parameters can be populated for the default iteration (Figure 12).

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).

To create new rows, you can click the *New* button (Figure 12 - 1), or navigate using the keyboard from the last row (a new row will be created automatically by navigating forward to the last cell of the last row).

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

New + 1

Figure 12 - Parameters

Converting a Seeding Parameter into a Combinatorial Parameter

You can convert an existing (seeding) parameter into 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.

1 - Next to each column, there is an ellipsis button (Figure 13 - 1). Clicking this button will reveal a menu with options for *Editing*, *Deleting*, and *Converting* parameters (Figure 13 - 2).

2 - Click the *Convert to combinatorial parameter* (Figure 13 - 2) option.

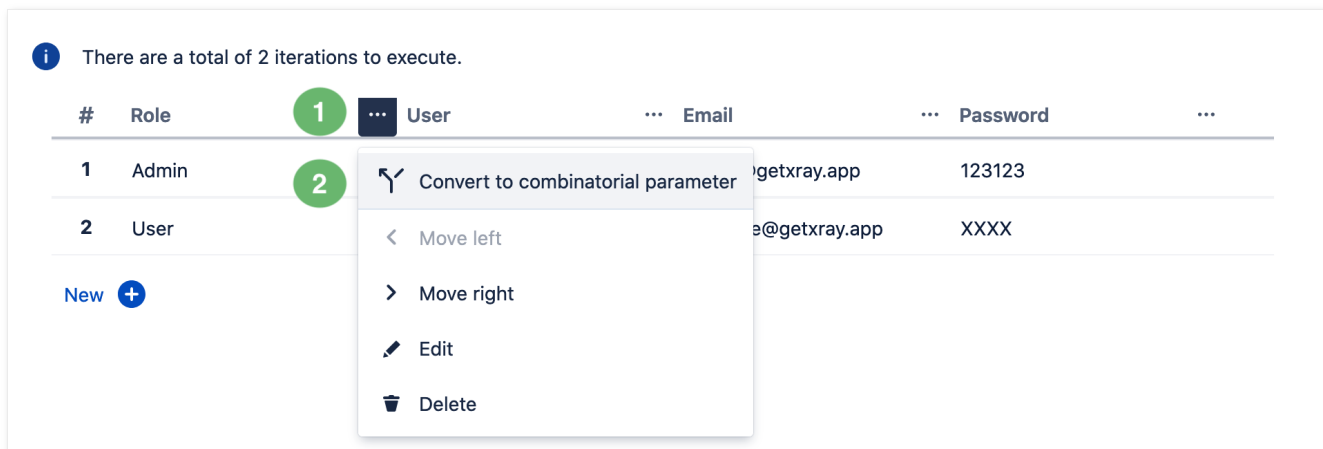


Figure 13 - Menu

3 - A modal will open (Figure 14). Verify the parameters (Figure 14 - 1) and once you're finished, click Save (Figure 14 - 2).

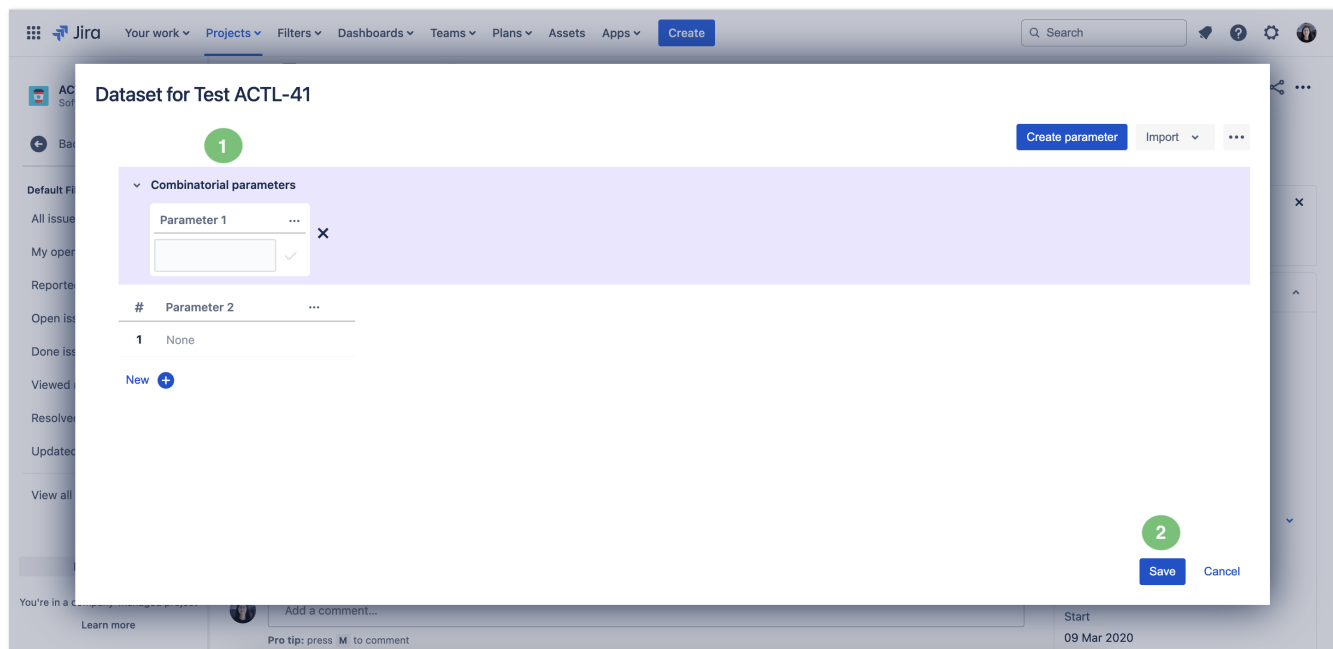


Figure 14 - Parameters

Converting a Combinatorial Parameter into a Seeding Parameter

You can convert a combinatorial parameter back into a seeding parameter.

1 - Click the ellipsis button (Figure 15 - 1). Then, select *Convert to non-combinatorial parameter* (Figure 15 - 3).

2 - Once you're finished, click Save (Figure 15 - 4).

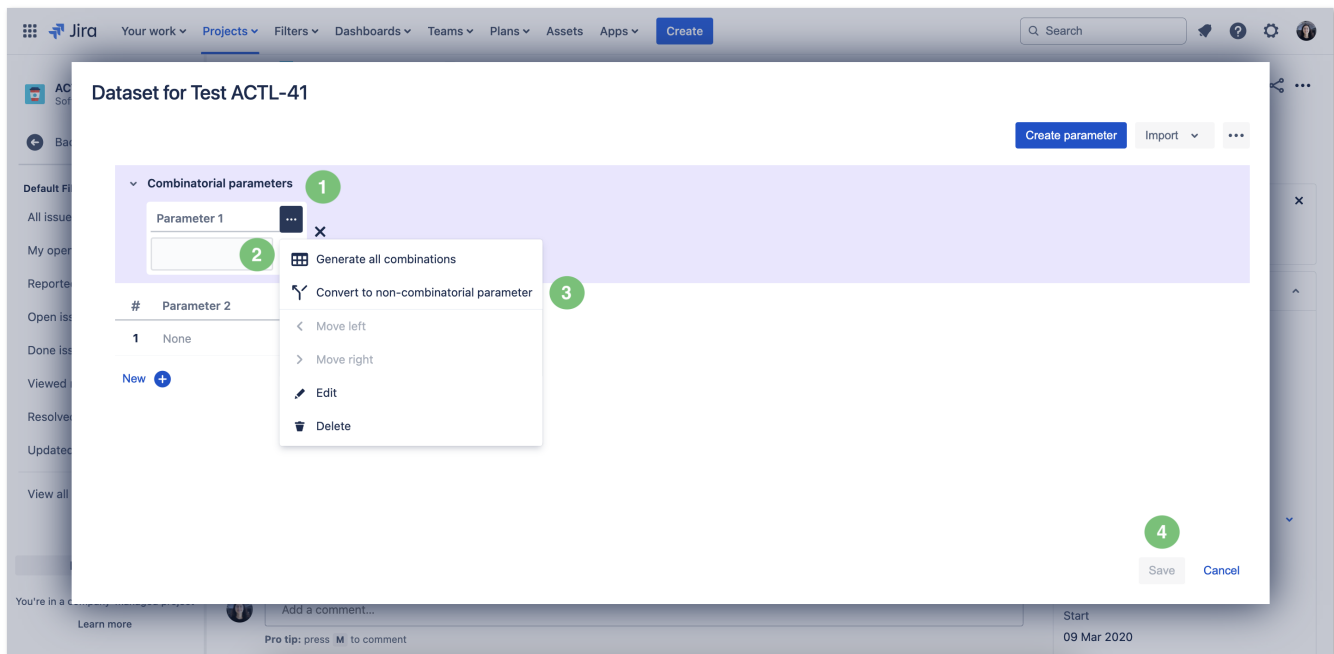


Figure 15 - Parameters

Generating all Combinations

You don't need to generate all the combinations for a dataset to execute all iterations. Xray will do this automatically for all of the combinatorial parameters. However, if you don't need all the combinations, you can generate all combinations and remove some iterations afterward.

1 - Click the ellipsis button (Figure 15 - 1). Then, select *Generate all combinations* (Figure 15 - 2).

2 - A modal will open for you to proceed. Click *Confirm* (Figure 16 - 1). Confirming the changes will apply the Cartesian product between the combinatorial parameter values and the seeding parameter rows.

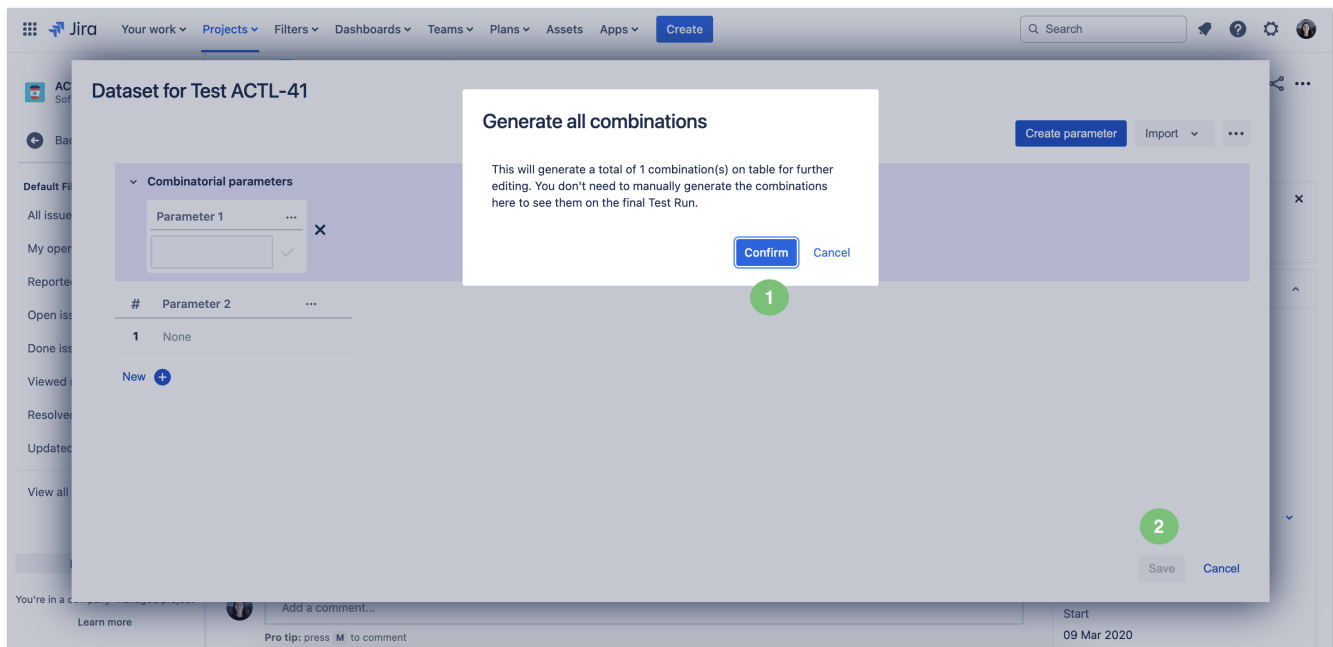


Figure 16 - Modal

3 - Once you're finished, click *Save* (Figure 16 - 2).

Moving Parameters Left/Right

You can reorder any parameters in your dataset, by clicking the ellipsis button (Figure 17 - 1) and selecting the *Move left/right* option. This will swap the parameters' positions.

There are a total of 2 iterations to execute.

#	Role	1	...	User	...	Email	...	Password	...
1	Admin					getxray.app		123123	
2	User					e@getxray.app		XXXX	
New		2							

Convert to combinatorial parameter

< Move left

> Move right

Edit

Delete

Figure 17 - Parameters

After making any changes to the dataset, click the *Save* button (Figure 16 - 2) to keep the dataset information in the database updated.

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.

1

On the Test Issue, click the *Dataset* icon (Figure 18 - 1). A modal will open (Figure 19).

ACTL

Software project

Back to project

Default Filters

All issues

My open issues

Reported by me

Open issues

Done issues

Viewed recently

Resolved recently

Updated recently

View all filters

Invite people

You're in a company-managed project

Learn more

Projects / ACTL / Add parent / ACTL-41

ACTL-42 This a Story

TO DO - TP

Test details

Test details

Preconditions

Test Sets

Test Plans

Test Runs

Test Type

Manual

Test Repository / New Folder / New Folder2

1

Dataset

Add Step

Import

Export

1

Action

Step 1

Data

Data 1

Expected Result

Expected Result 1

System

None

Required

None

Review Comments

None

Attachments (1)

priority.png

2

Action

Step 2

Data

Data 2

Expected Result

Expected Result 2

Add a comment...

Pro tip: press **M** to comment

In Progress - TTPPPP

Actions

Pinned fields

Click on the next to a field to start pinning.

Details

Assignee

Ana Leite (Deactivate)

Assign to me

Reporter

Ana Leite (Deactivate)

Development

Create branch

Create commit

Labels

calculator

Start

09 Mar 2020

Figure 18 - Dataset

2

Click the *Import* button (Figure 19 - 1) and select the *CSV file* option (Figure 19 - 2). A modal will open (Figure 20).

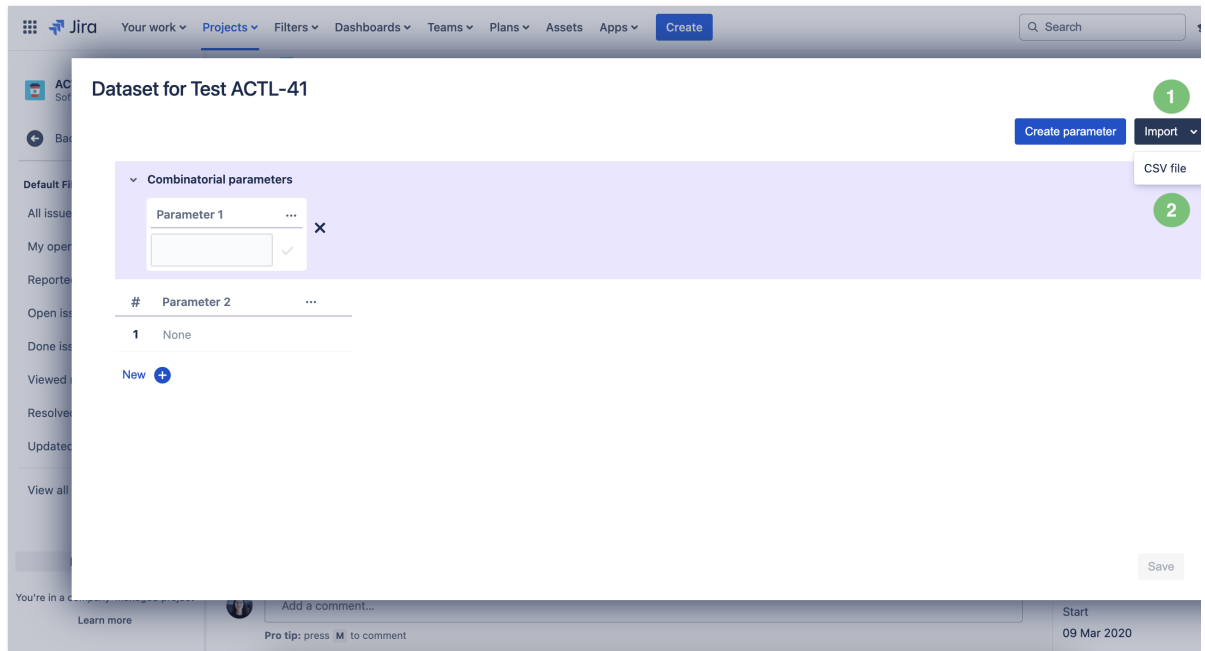


Figure 19 - Import

3

In the Import from CSV modal, select and fill in the fields (Figure 20):

- *Choose file* - click this button to find a local CSV file with the desired dataset (Figure 20 - 1).
- *CSV Delimiter* - defaults to ",", but you can choose any other character (Figure 20 - 2).
- *File Encoding* - defaults to UTF-8 (Figure 20 - 2).
- *Overwrite existing parameter values* - if enabled, this option will delete any values from existing parameters (Figure 20 - 3).
- *Create new parameters* - if enabled, 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 (Figure 20 - 3).

Import from CSV

1 books_dataset.csv

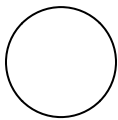
CSV Delimiter File Encoding 2

3 ☐ Overwrite existing parameter values

☒ Create new parameters

4

Figure 20 - CSV



Once you're finished, click the *Import* button (Figure 20 - 4).



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 (Figure 20 - 3) must be enabled.

Exporting a Dataset to a CSV File

1

On the Test Issue, click the *Dataset* icon (Figure 21 - 1). A Dataset modal will open (Figure 22).

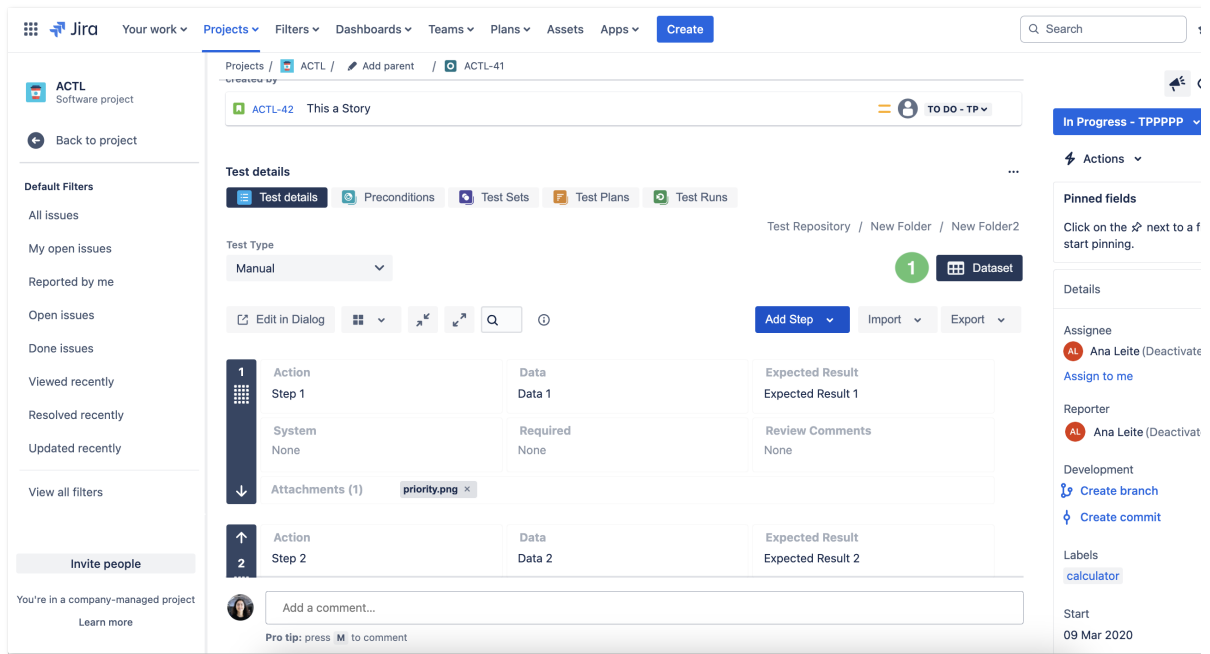


Figure 21 - Dataset

2

In the Dataset modal (Figure 22), click the ellipsis button (Figure 22 - 1), and select the *Export to CSV* option (Figure 22 - 2).

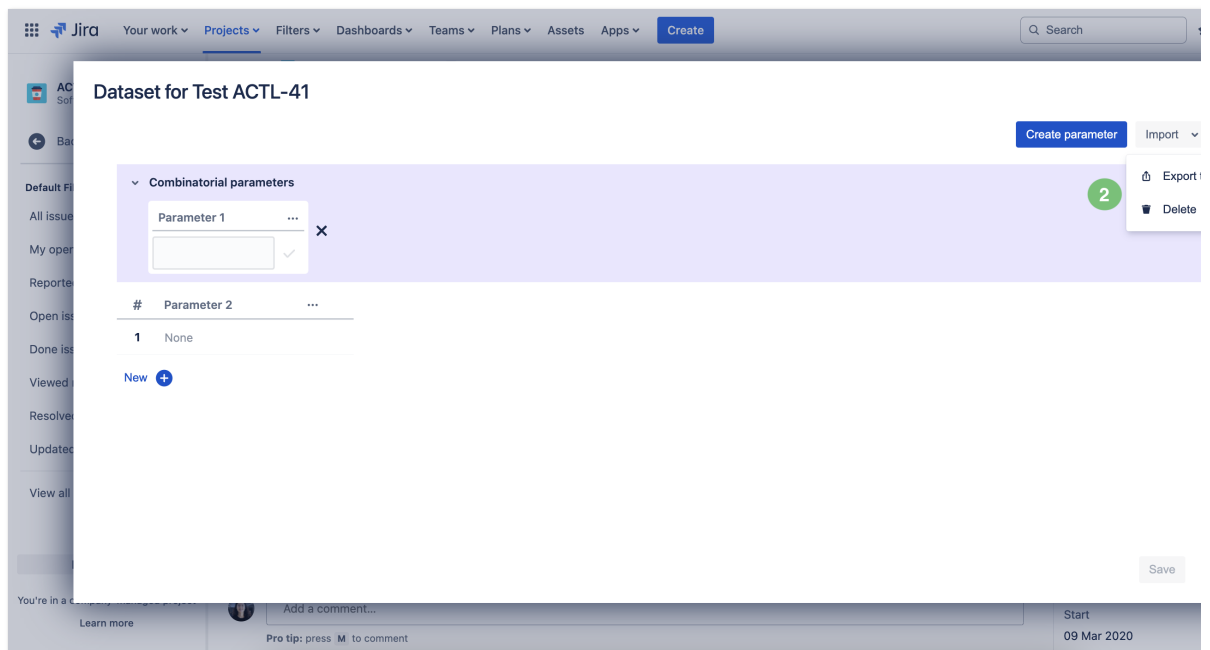



Figure 22 - Export

3

The file download will start right away and you will get the file on your machine.

 The delimiter used to generate the CSV file is a **comma**.

Deleting a Dataset

1

On the Test Issue, click the *Dataset* icon (Figure 21 - 1). The Dataset modal will open (Figure 22).

2

In the Dataset modal, click the ellipsis button (Figure 22 - 1) and select the *Delete* option (Figure 22 - 2). A modal will open for you to confirm the removal of the dataset (Figure 23). Click *Confirm* to proceed (Figure 23 - 1).

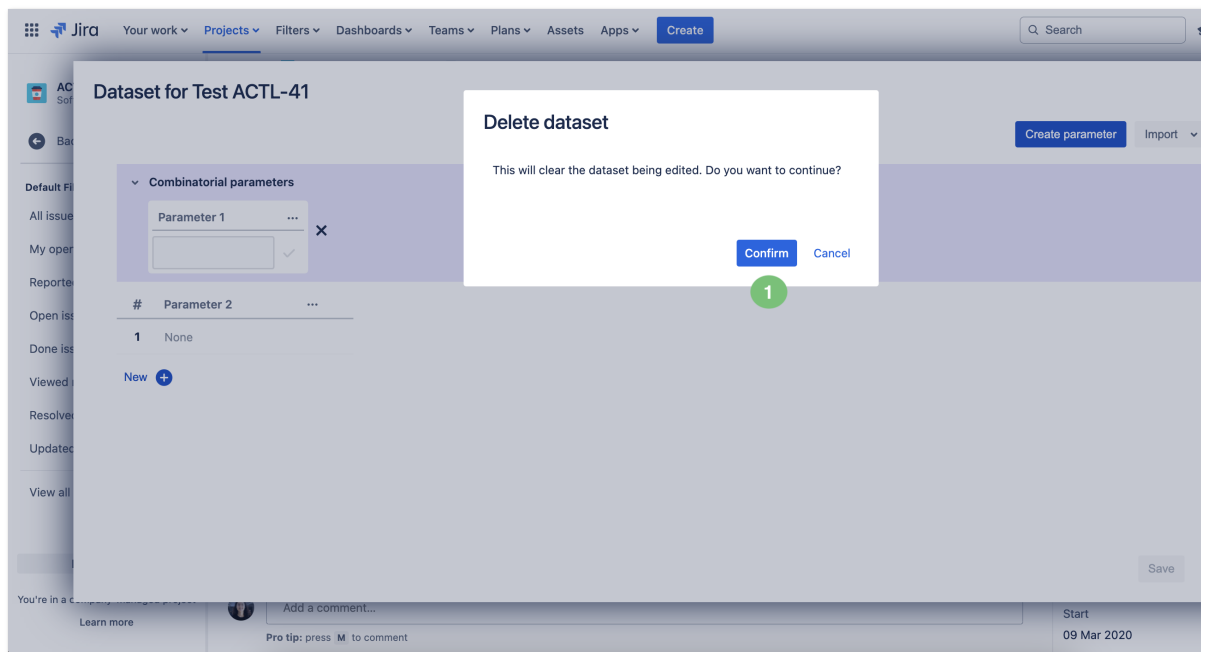


Figure 23 - Delete

Parameterized Preconditions

Precondition Issues can also be parameterized by including parameter names in the precondition specification (Figure 24).

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

Precondition Details

Manual

Steps

We must verify if item `${Item}` is in stock.

Figure 24 - Precondition

Iterations Execution

All iterations for a given Test are **executed** within the context of the same Test run. Each iteration can be expanded by clicking the arrow icon (Figure 25 - 1), and the Test Steps executed individually (Figure 25 - 2). The Step parameters will be replaced by the corresponding iteration values. The Steps affect the iteration status which, in turn, affects the overall Test run status (Figure 25 - 3).

Test Details

MANUAL

Iterations

Iteration 1

In Search of Lost Time

\$34

5

Marcel Proust

Yes

New

Paperback

Yes

1

PASSED

Iteration 2

One Hundred Years of Solitude

\$20

4.9

Gabriel Garcia Marquez

Yes

Used

Paperback

Yes

1

PASSED

Iteration 3

The Great Gatsby

\$39

4.7

F. Scott Fitzgerald

No

New

Kindle

Yes

1

PASSED

Iteration 4

In Search of Lost Time

\$34

5

Marcel Proust

Yes

New

Paperback

No

1

PASSED

Iteration 5

One Hundred Years of Solitude

\$20

4.9

Gabriel Garcia Marquez

Yes

Used

Paperback

No

1

PASSED

Iteration 6

The Great Gatsby

\$39

4.7

F. Scott Fitzgerald

No

New

Kindle

No

1

PASSED

Iteration 7

In Search of Lost Time

\$34

5

Marcel Proust

Yes

New

Paperback

Yes

2

EXECUTING

Steps

1

Action

Add 2 books of In Search of Lost Time into the basket.

Data

None

Expected Result

After items are added, a confirmation message appears mentioning 2 items of In Search of Lost Time were added to the basket.

Actual Result

Comment

Defects

Evidence

Step State

PASSED

2

Action

Click on the basket icon located on the top right toolbar of the app.

Data

None

Expected Result

The basket page is displayed containing all 5 items.

Actual Result

Comment

Defects

Evidence

Step State

PASSED

3

Action

Attachments (1)

Press the Checkout button to start the checkout process.

Data

None

Expected Result

The checkout process is initiated asking the user the address details.

Figure 25 - Iterations

Overriding a Dataset in a Test Plan or Test Execution

- 1
- Navigate to the desired Test plan or Test execution Issue (Figure 26).
- 2
- A menu action is provided on each Test row (Figure 26 - 2). A column named *Dataset* (Figure 26 - 1) 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 (Figure 26 - 1) will be displayed with a selected style.

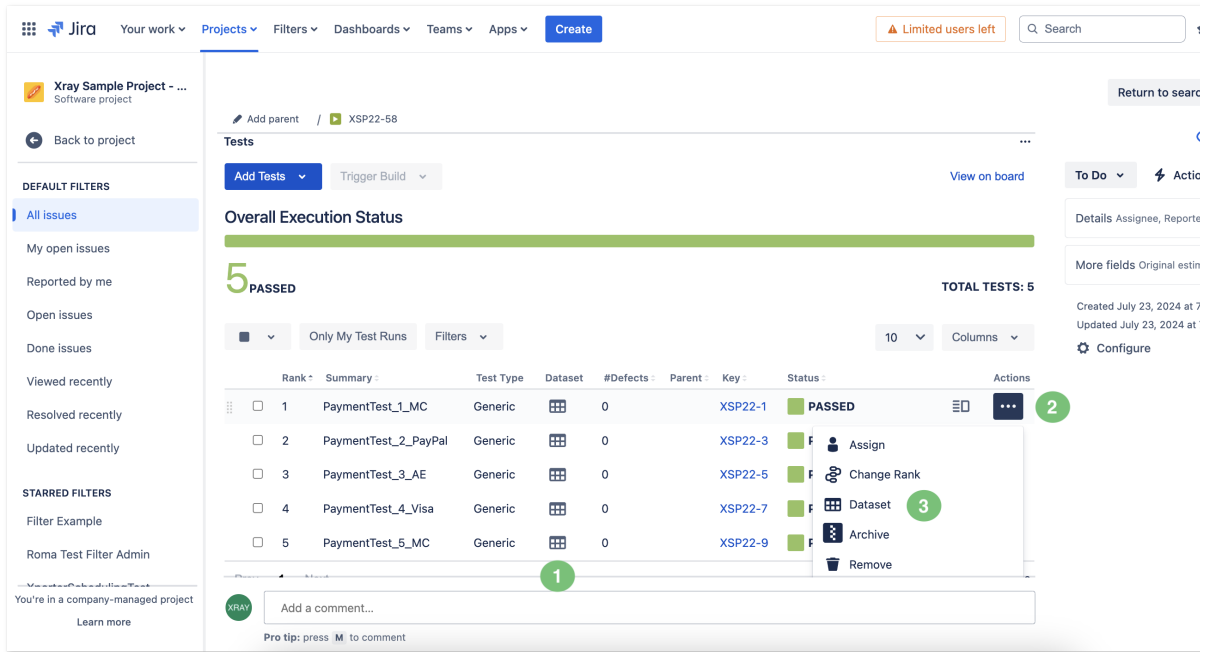


Figure 26 - Execution

3

Click the *Dataset* button (Figure 26 - 1) or click the *Actions* button (Figure 26 - 2) and then select *Dataset* (Figure 26 - 3). The Dataset modal will open (Figure 27).

4

In the Dataset modal (Figure 27), if there is a dataset defined on a parent level, you can override the parent dataset and modify its values by clicking *Override* (Figure 27 - 1). Otherwise, you can start defining a new dataset at this level (Figure 27 - 2).

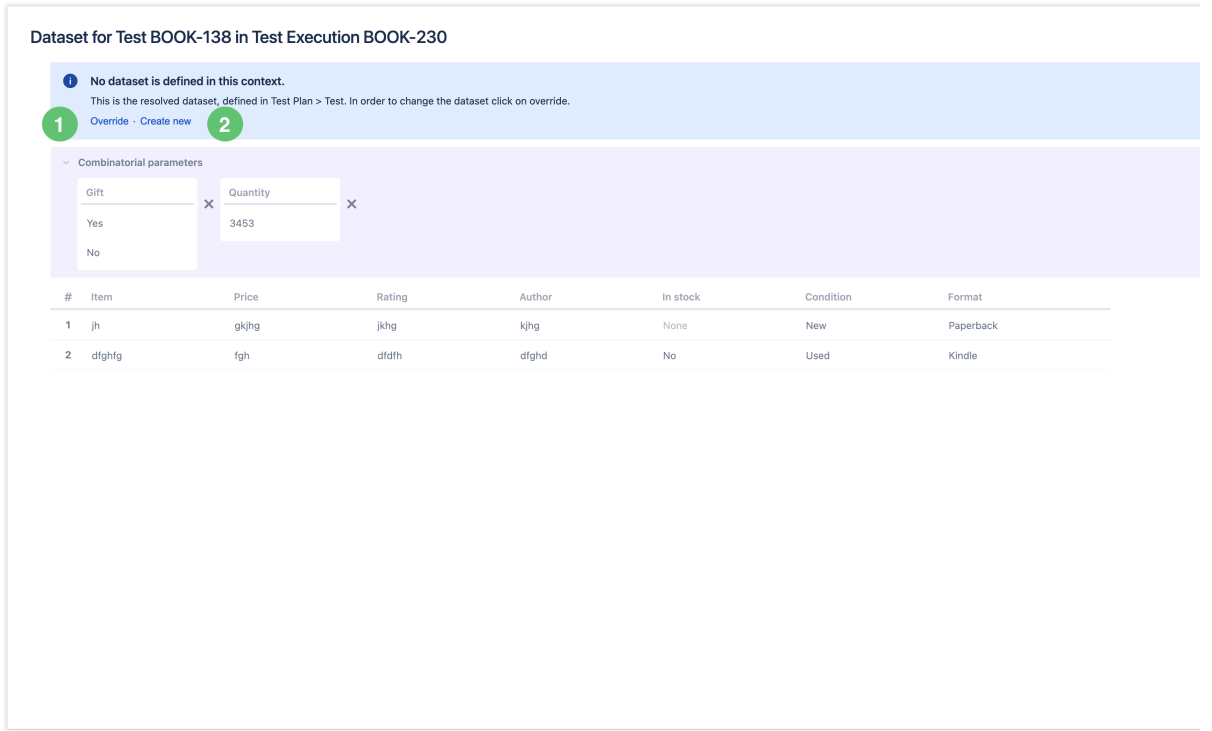


Figure 27 - Override

5

Once the dataset is defined (or overridden), click **Save** (Figure 27 - 3).

If you have questions or technical issues, please [contact the Support team via the Customer Portal \(Jira service management\)](#) or [send us a message using the in-app chat](#).