Create 3 issues in 3 different projects

On this page

Features used to implement the example

- **Create issues and sub-tasks**

Example: Create 3 issues in 3 different projects

This is an example of creation of **multiple issues** based on **seed numbers** using **Create issues and sub-tasks** post-function. This example requires version **2.2.25** or higher.

We want to automatically create 3 tasks in 3 different project (project keys **DCM**, **NA** and **TS**), each time a task is moved to "**In Progress**" status in project with key **ER**.

We insert **Create issues and sub-tasks** post-function in transition "**Start Progress**" of **tasks** in project **ER**. We use the following configuration:

Issues to be created:
Sets the number of issues that will be created.

○ **Only one** issue ◉ **Multiple issues** based on seeds: Math Expression

Check Syntax [Line 1 / Col 3]

Math expression (Syntax Specification and Examples)

```
3
```

Input a mathematic expression returning the number of issues you want to be created. From here on, you will be able to obtain the order of creation (**seed number**, statring by 1) of each issue being created using ^.

String Field Code Injector: Summary - [Text] - %{00000}

Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

Issue Type:
Sets the issue type of the issues to be created.

Test

Project:
Sets the project of the issues to be created.

○ **Current Project** ○ **Selected Project** ○ **Seed Issue's Project** ◉ **Project Key**

Check Syntax [Line 1 / Col 1]

String expression (i.e, advanced parsed text) (Syntax Specification and Examples)

```
getMatchingValue(^, [1, 2, 3], ["DCM", "NA", "TS"])
```

Enter a string expression that returns the project key of the project where the subtask will be created.

String Field Code Injector: Summary - [Text] - %{00000}

Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

Field code injectors reference: ◉ **Current issue** ○ **Seed issue**

Summary:
Sets the summary of the issues to be created.

Parsing mode:
○ **basic** ◉ **advanced**

Check Syntax [Line 1 / Col 1]

```
getMatchingValue(^, [1, 2, 3], ["Install PC for " + %{12700}, "Network account for " + %{12700}, "Telephone set for " + %{12700}])
```

Strings literals are written in double quotes (`"This is a string."`). Operator '+' is used to concatenate strings, and field codes are like in basic mode, e.g., `"Issue key is " + %{00015} + "."`. More information at **parser syntax documentation**.

String Field Code Injector: Summary - [Text] - %{00000}

Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

Field code injectors reference: ◉ **Current issue** ○ **Seed issue** ○ **Parent of new sub-task**

Description:
Sets the description of the issues to be created.

Parsing mode:
○ **basic** ◉ **advanced**

Check Syntax [Line 1 / Col 1]

```
getMatchingValue(^, [1, 2, 3],
[
"Install a PC for the new employee " + %{12700},
"Create network a account for the new employee " + %{12700},
"Install a telephone set for the new employee "  + %{12700}
])
```

Strings literals are written in double quotes (`"This is a string."`). Operator '+' is used to concatenate strings, and field codes are like in basic mode, e.g., `"Issue key is " + %{00015} + "."`. More information at **parser syntax documentation**.

String Field Code Injector: Summary - [Text] - %{00000}

Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

Field code injectors reference: ◉ **Current issue** ○ **Seed issue** ○ **Parent of new sub-task**

Set Fields:
Sets field values in the new issues.

Field to be set:

Due date - [Date] | Add

Field	Type of Value	Value	Actions
Reporter	Field in current issue	Assignee	Edit Remove
Assignee	Parsed text (basic mode)	null	Edit Remove

Inherit Remaining Fields:
Inherit field values from other issues, for those fields that has not been set in the previous section.

Inherit from Current Issue

Issue Links:
The newly created issues can be linked to other issues.

Add Issue Link

Issue Link Type	Linked Issues	Condition	Actions
is caused by	Current Issue		Edit Remove

Additional Actions:
Optional actions that will be exectuted after all issues have been created.

☐ **Save issue keys** of created issues into ***Ephemeral String 3*** virtual field as a comma separated list.

Conditional execution:
Optional boolean expression that should be satisfied in order to actually execute the post-function.
(Syntax Specification)

1

Leave the field empty for executing the post-function unconditionally. Collection of Examples [Line 1 / Col 1]

Logical connectives: `and`, `or` and `not`. Alternatively you can also use `&`, `|` and `!`.
Comparison operators: `=`, `!=`, `>`, `>=`, `<` and `<=`. Operators `in`, `not in`, `any in`, `none in`, `~` and `!~` can be used with *strings*, *multi-valued fields* and *lists*.
Logical literals: `true` and `false`. Literal `null` is used with `=` and `!=` to check whether a field is initialized, e.g. `{00012} != null` checks whether *Due Date* is initialized.

Check Syntax

String Field Code Injector: Summary - [Text] - %{00000}

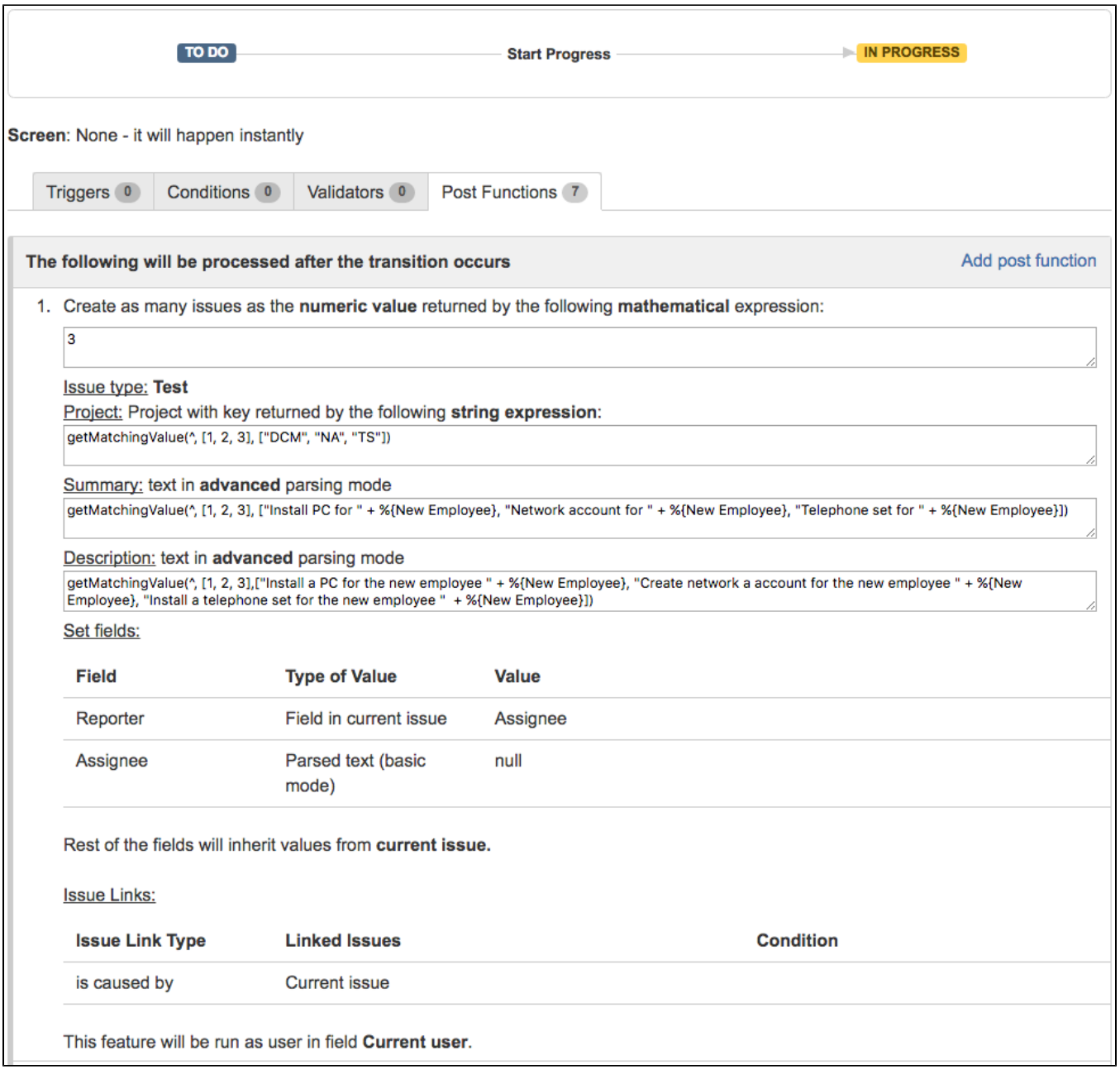
Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

Run as:
Select the user that will be used to execute this feature. JIRA will apply restrictions according to the permissions, project roles and groups of the selected user.

Current user

User defined by a **field**. Input a **specific user**.

Once configured, transition will look like this:



Result screenshots post-function "Create issues and subtasks" - 3 issues in different projects

---

Other examples of that function

Page: Assign new issues to a different project role depending on field value in current issue
Page: Clone an issue and all its subtasks (with additional restrictions)
Page: Create 3 issues in 3 different projects
Page: Create a dynamic set of sub-tasks based on checkbox selection with unique summaries
Page: Create a static set of sub-tasks with unique summaries
Page: Create a story for each component in an epic
Page: Create a sub-task for each user selected in a Multi-User Picker
Page: Create a sub-task in each story of an epic
Page: Create specific sub-tasks for each selected component

Related Usage Examples

- Creating a Jira Service Desk internal comment
  - example
  - post-function
- Limit the number of hours a user can log per day
  - example
  - validator
  - post-function
  - work-log
- Set a date based on current date
  - example
  - post-function
- Setting the priority depending on the multiplication of custom fields