Create a sub-task for each user selected in a Multi-User Picker

On this page

Features used to implement the example

- **Create issues and sub-task**

Example: Create a sub-task for each user selected in a Multi-User Picker custom field

This is an example of creation of **multiple issues** based on **seed strings** using **Create issues and sub-tasks** post-function.

We want to automatically create as many sub-tasks as users are selected in a **Multi-User Picker** custom field called **Approvers** and assign them to a user in that field when a certain transition is executed in parent issue.

We use **Create issues and sub-tasks** post-function in a transition or the workflow of parent issue with the following configuration:

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

○ **Only one** issue ◉ **Multiple issues** based on seeds: String List

Check Syntax [Line 1 / Col 24]

String List expression (Syntax Specification and Examples)

```
1 toStringList(%{13300})
```

Input a expression returning a string list. An issue will be created per each string (**seed string**) in the string list. From here on, you will be able to reference seed strings using ^%.

String Field Code Injector: Approvers - [User Picker (multiple users)] - %{13300}

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

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

Sub-task

Parent Issue:
Sets the parent of the sub-tasks to be created.

Current Issue

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

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

Check Syntax [Line 1 / Col 1]

```
1 "Sub-task for " + userFullName(^%)
```

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]

```
1 "This sub-task has been automatically created for " + userFullName(^%) + " with username '" +
  ^% + "'."
```

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:

Reporter - [User] Add

Field	Type of Value	Value	Actions
Assignee	Parsed text (advanced mode)	^%	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

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 %{00041} = null
```

Leave the field empty for executing the post-function unconditionally. Collection of Examples [Line 1 / Col 17]

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:
Parent's issue key - [Text] - %{00041}

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:

The following will be processed after the transition occurs Add post function

1. Create an issue **per seed string** returned by the following **string list** expression:

```
toStringList(%{Approvers})
```

Issue type: **Sub-task**
Parent issue: **Current Issue**
Summary: text in **advanced** parsing mode

```
"Sub-task for " + userFullName(^%)
```

Description: text in **advanced** parsing mode

```
"This sub-task has been automatically created for " + userFullName(^%) + " with username '" + ^% + "'."
```

Set fields:

Field	Type of Value	Value
Assignee	Parsed text (advanced mode)	`^%`

Rest of the fields will inherit values from **current issue.**
Post-function will only be executed if the following boolean expression is satisfied: `%{Parent's issue key} = null`
This feature will be run as user in field **Current user**. by JWT

Result screenshots post-function "Create issues and subtasks" - Create a subtask per user selected in User Picker

The workflow is shared between parent issue and sub-task, thus we are using **Conditional execution** with boolean expression `%{00041} = null` to avoid the post-function to be executed by sub-tasks.

Note that:

- **%{00041}** is field code for **Parent's issue key**

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

- Create a dynamic set of sub-tasks based on checkbox selection with unique summaries
  - example
  - post-function
  - custom-field
  - sub-task
- Add and remove a single or a set of items from multi valued fields
  - example
  - post-function
  - custom-field
  - issue-links
  - sub-task
- Add all assignees of certain sub-task types to a "Multi-User Picker" custom field
  - example
  - post-function
  - custom-field
  - sub-task
- Update Cascading Select custom field with a value of the field in parent issue
  - example
  - post-function