

Create a sub-task in each story of an epic

On this page

- [Features used to implement the example](#)
- [Example: Create a sub-task in each Story of an Epic](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Create issues and sub-tasks](#)

Example: Create a sub-task in each Story of an Epic

This is an example of creation of **multiple issues** based on **seed issues** using [Create issues and sub-tasks](#) post-function. We are going to insert a post-function in a transition of **Epic's workflow** in order to create one sub-task on each **Story** which is in status **"To Do"** or **"Open"**.

We insert [Create issues and sub-tasks](#) post-function in a transition of Epic's workflow using the following configuration:

Issues to be created: Sets the number of issues that will be created.	<input type="radio"/> Only one issue <input checked="" type="radio"/> Multiple issues based on seeds: <input type="text" value="Issue List"/>	Check Syntax [Line 1 / Col 87]
Issue List expression (Syntax Specification and Examples) <pre>1 filterByStatus(filterByIssueType(linkedIssues("is Epic of"), "Story"), "To Do, Open")</pre> Input an expression returning an issue list. An issue will be created per each issue (seed issue) in the issue list. From here on, you will be able to reference fields in seed issues preceeding field codes with ^.		
String Field Code Injector: <input type="text" value="Summary - [Text] - %{000000}"/>		Numeric/Date Field Code Injector: <input type="text" value="Original estimate (minutes) - [Number] - {00068}"/>
Issue Type: Sets the issue type of the issues to be created.	<input type="text" value="Sub-task"/>	
Parent Issue: Sets the parent of the sub-tasks to be created.	<input type="text" value="Seed Issue"/>	

Summary:

Sets the summary of the issues to be created.

Parsing mode:

basic

advanced

Check Syntax

[Line 1 / Col 1]

1

"Estimate effort of story " + ^%{00000} + " "

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

Perform the effort estimation this story.

Insert field codes anywhere in the text, and they will be replaced with corresponding field values. Field code formats are %{nnnnn}, and %{nnnnn.i} for Cascading Select fields (i = 0 for base level).

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:

Priority - [Issue priority]

Add

Field	Type of Value	Value	Actions
Assignee	Field in seed issue	Assignee	<div>Edit</div> <div>Remove</div>
Reporter	Field in current issue	Reporter	<div>Edit</div> <div>Remove</div>
Due date	Math/Time expression	addDaysSkippingWeekends({Current date and time}, 3, LOCAL)	<div>Edit</div> <div>Remove</div>
Components	Field in seed issue	Components	<div>Edit</div> <div>Remove</div>
New comment	Parsed text (advanced mode)	"This issue was automatically generated by " + %{Issue key} + " epic issue for " + ^%{Issue key} + " story."	<div>Edit</div> <div>Remove</div>

Inherit Remaining Fields:

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

Don't inherit field values

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 executed after all issues have been created. <div> <input type="checkbox"/> Save issue keys of created issues into <i>Ephemeral String 3</i> virtual field as a comma separated list. </div>			
Conditional execution: Optional boolean expression that should be satisfied in order to actually execute the post-function. (Syntax Specification) <div> <div> 1 <div> <code>&{00014} = "Epic"</code> </div> </div> <div> Leave the field empty for executing the post-function unconditionally. Collection of Examples [Line 1 / Col 1] </div> </div> <div> <p><u>Logical connectives:</u> and, or and not. Alternatively you can also use &, and !.</p> <p><u>Comparison operators:</u> =, !=, >, >=, < and <=. Operators in, not in, any in, none in, ~ and != can be used with <i>strings</i>, <i>multi-valued fields</i> and <i>lists</i>.</p> <p><u>Logical literals:</u> true and false. Literal null is used with = and != to check whether a field is initialized, e.g. {00012} != null checks whether <i>Due Date</i> is initialized.</p> <div> <div> String Field Code Injector: <div>Summary - [Text] - %{00000} ▾</div> </div> <div> Numeric/Date Field Code Injector: <div>Original estimate (minutes) - [Number] - {00068} ▾</div> </div> </div> <div>Check Syntax</div> </div>			
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. <div> <div> Current user ▾ </div> <div> <input type="text"/> ▾ </div> </div> <div> User defined by a field. Input a specific user. </div>			

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 issue** returned by the following **issue list** expression:

```
filterByStatus(filterByIssueType(linkedIssues("is Epic of"), "Story"), "To Do, Open")
```

Issue type: **Sub-task**

Parent issue: **Seed Issue**

Summary: text in **advanced** parsing mode

```
"Estimate effort of story " + ^%{Summary} + ""
```

Description: text in **basic** parsing mode

```
Perform the effort estimation this story.
```

Set fields:

Field	Type of Value	Value
Assignee	Field in seed issue	Assignee
Reporter	Field in current issue	Reporter
Due date	Math/Time expression	<code>addDaysSkippingWeekends({Current date and time}, 3, LOCAL)</code>
Components	Field in seed issue	Components
New comment	Parsed text (advanced mode)	<code>"This issue was automatically generated by " + % {Issue key} + " epic issue for " + ^% {Issue key} + " story."</code>

Post-function will only be executed if the following boolean expression is satisfied: `% {Issue type} = "Epic"`
This feature will be run as user in field **Current user**.

Seed Issues

We use the following issue list expression for selecting those issues linked to the Epic (using issue link type **"is Epic of"**), with issue type **"Story"** which are in statuses **"To Do"** or **"Open"**:

```
filterByStatus(filterByIssueType(linkedIssues("is Epic of"), "Story"), "To Do, Open")
```

Parent

As the issue type of the new issues is **"Sub-task"**, we should specify a parent issue for each of them. In this case the parent issues will be the **seed issues**, i.e., each of the stories selected by the former issue list expression.

Assignee

The sub-tasks will be assigned to the same user who has the parent Story assigned.

Reporter

The sub-task will be reported by the assignee of the Epic issue.

Due Date

The due date of the sub-task will be 3 days ahead skipping weekends.

We use the following formula: `addDaysSkippingWeekends({00057}, 3, LOCAL)`

New comment

An automatic comment will be created in each new sub-task.

We use the following text expression: `"This issue was automatically generated by " + %{00015} + " Epic issue for " + ^%{00015} + " Story."`

Conditional execution

We set a condition so that we ensure that the post-function is only executed when current issue is an **Epic**, this way we can use the post-function in workflows shared with other issue types.

[Result screenshots post-function "Create issues and subtasks" - Create subtask for each Story of an Epic](#)

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

- [Validation on sibling sub-tasks depending on issue type and status](#)
 - [example](#)
 - [validator](#)
 - [sub-task](#)
 - [transition](#)
- [Restrict sub-task type creation depending on parent issue status](#)
 - [example](#)
 - [validator](#)
 - [sub-task](#)
- [Require at least one sub-task in status "Resolved" or "Closed" when "Testing required" is selected in Check-Box custom field](#)
 - [example](#)
 - [validator](#)
 - [sub-task](#)
- [Restrict sub-task type creation depending on parent issue type](#)
 - [example](#)
 - [validator](#)
 - [sub-task](#)
- [Block a transition until all sub-tasks have certain fields populated](#)
 - [example](#)
 - [condition](#)
 - [validator](#)
 - [sub-task](#)
 - [transition](#)
- [Create a dynamic set of sub-tasks based on checkbox selection with unique summaries](#)
 - [example](#)
 - [post-function](#)
 - [custom-field](#)
 - [sub-task](#)
- [Transition sub-tasks when parent is transitioned](#)
 - [example](#)
 - [post-function](#)
 - [sub-task](#)
 - [transition](#)
 - [outdated](#)
- [Transition only a sub-task among several ones](#)
 - [example](#)
 - [post-function](#)
 - [sub-task](#)
 - [transition](#)
 - [outdated](#)
- [Moving sub-tasks to "Open" status when parent issue moves to "In Progress"](#)
 - [example](#)
 - [post-function](#)
 - [sub-task](#)
 - [transition](#)
 - [outdated](#)

- Moving story to "Ready for QA" once all its sub-tasks are in "Ready for QA" status
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Change parent's status depending on sub-task's summary
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress"
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Add and remove a single or a set of items from multi valued fields
 - example
 - post-function
 - custom-field
 - issue-links
 - sub-task
- Automatically close resolved sub-tasks when parent issue is closed
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Close parent issue when all sub-tasks are closed
 - example
 - condition
 - validator
 - post-function
 - sub-task
 - transition