Create a static set of sub-tasks with unique summaries

On this page

Features used to implement the example

- **Create issues and sub-tasks**

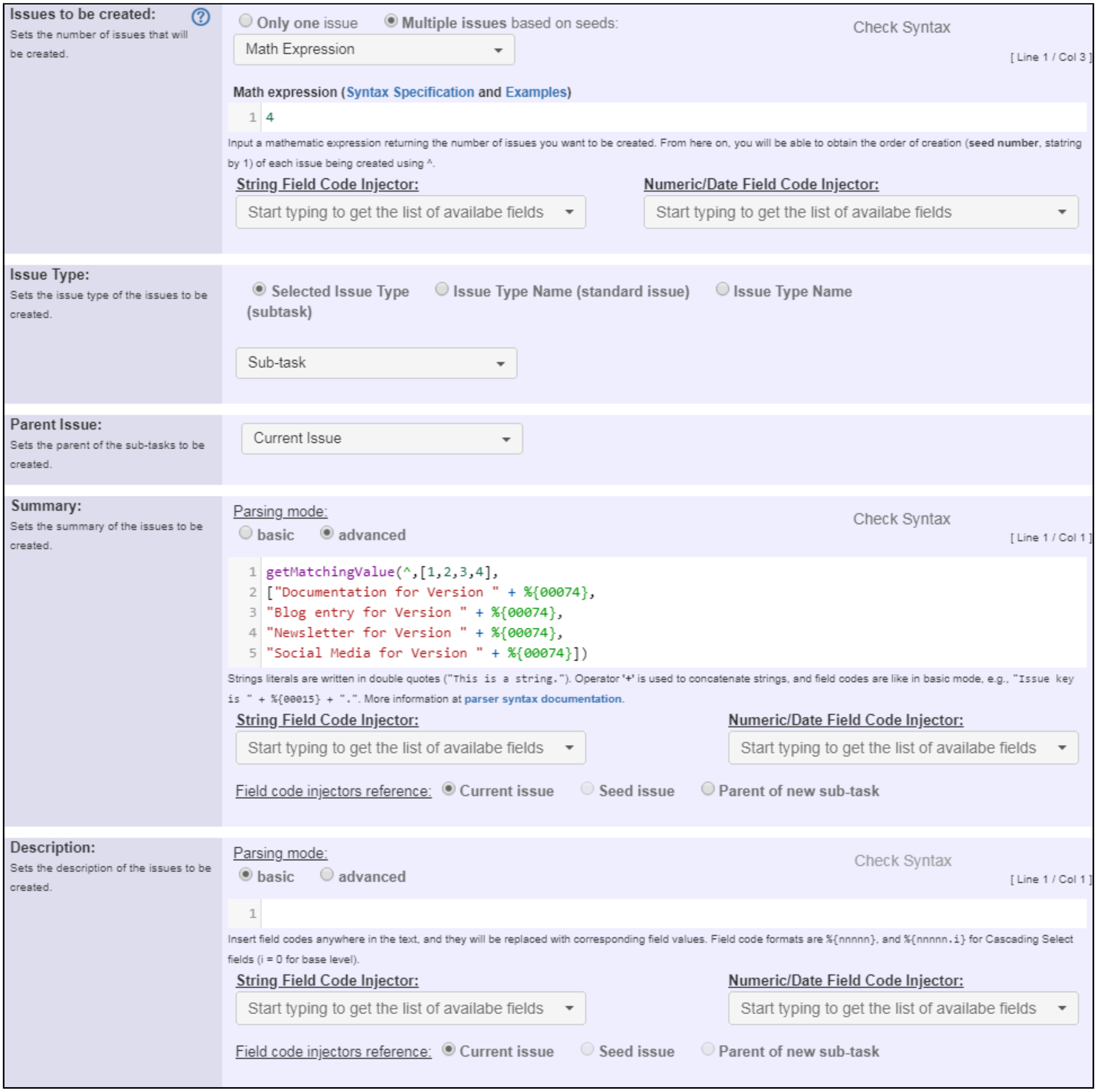
Example: Create a static set of sub-tasks with unique summaries

In this example, multiple sub-tasks with unique summaries are added to an issue on issue creation, e.g.

Creating issue "Release x.y.z" should automatically create sub-tasks

- "Documentation for Version x.y.z"
- "Blog entry for Version x.y.z"
- "Newsletter for Version x.y.z"
- "Social Media for Version x.y.z"

The configuration will look like this:



Issues to be created:

Sets the number of issues that will be created.

○ Only one issue ● **Multiple issues** based on seeds:

Math Expression

[Line 1 / Col 3]

Math expression (Syntax Specification and Examples)

```
4
```

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: Start typing to get the list of availabe fields

Numeric/Date Field Code Injector: Start typing to get the list of availabe fields

Issue Type:

Sets the issue type of the issues to be created.

● Selected Issue Type ○ Issue Type Name (standard issue) ○ Issue Type Name (subtask)

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

[Line 1 / Col 1]

```
getMatchingValue(^,[1,2,3,4],
["Documentation for Version " + %{00074},
"Blog entry for Version " + %{00074},
"Newsletter for Version " + %{00074},
"Social Media for Version " + %{00074}])
```

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: Start typing to get the list of availabe fields

Numeric/Date Field Code Injector: Start typing to get the list of availabe fields

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

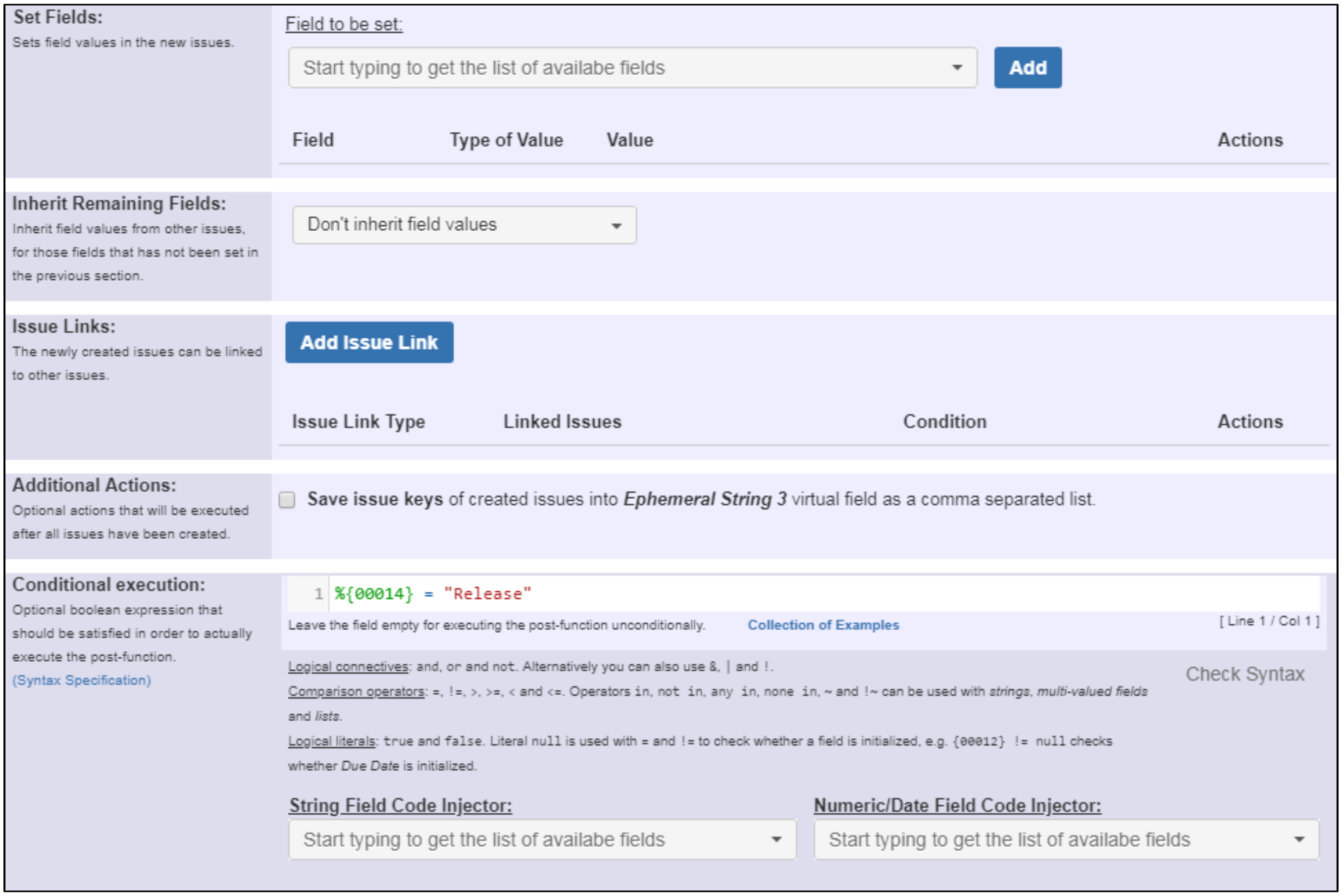
[Line 1 / Col 1]

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: Start typing to get the list of availabe fields

Numeric/Date Field Code Injector: Start typing to get the list of availabe fields

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:
Start typing to get the list of availabe fields

Field	Type of Value	Value	Actions

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.

Issue Link Type	Linked Issues	Condition	Actions

Additional Actions:
Optional actions that will be executed 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)

```
%{00014} = "Release"
```

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.

String Field Code Injector:
Start typing to get the list of availabe fields

Numeric/Date Field Code Injector:
Start typing to get the list of availabe fields

As four subtasks are always to be created in total, the **Issues to be created** option is set to **Math Expression** with the value **4**.

The dynamic summaries are parsed with the following expression:

```
getMatchingValue(^,[1,2,3,4],
["Documentation for Version " + %{00074},
"Blog entry for Version " + %{00074},
"Newsletter for Version " + %{00074},
"Social Media for Version " + %{00074}])
```

Note that:

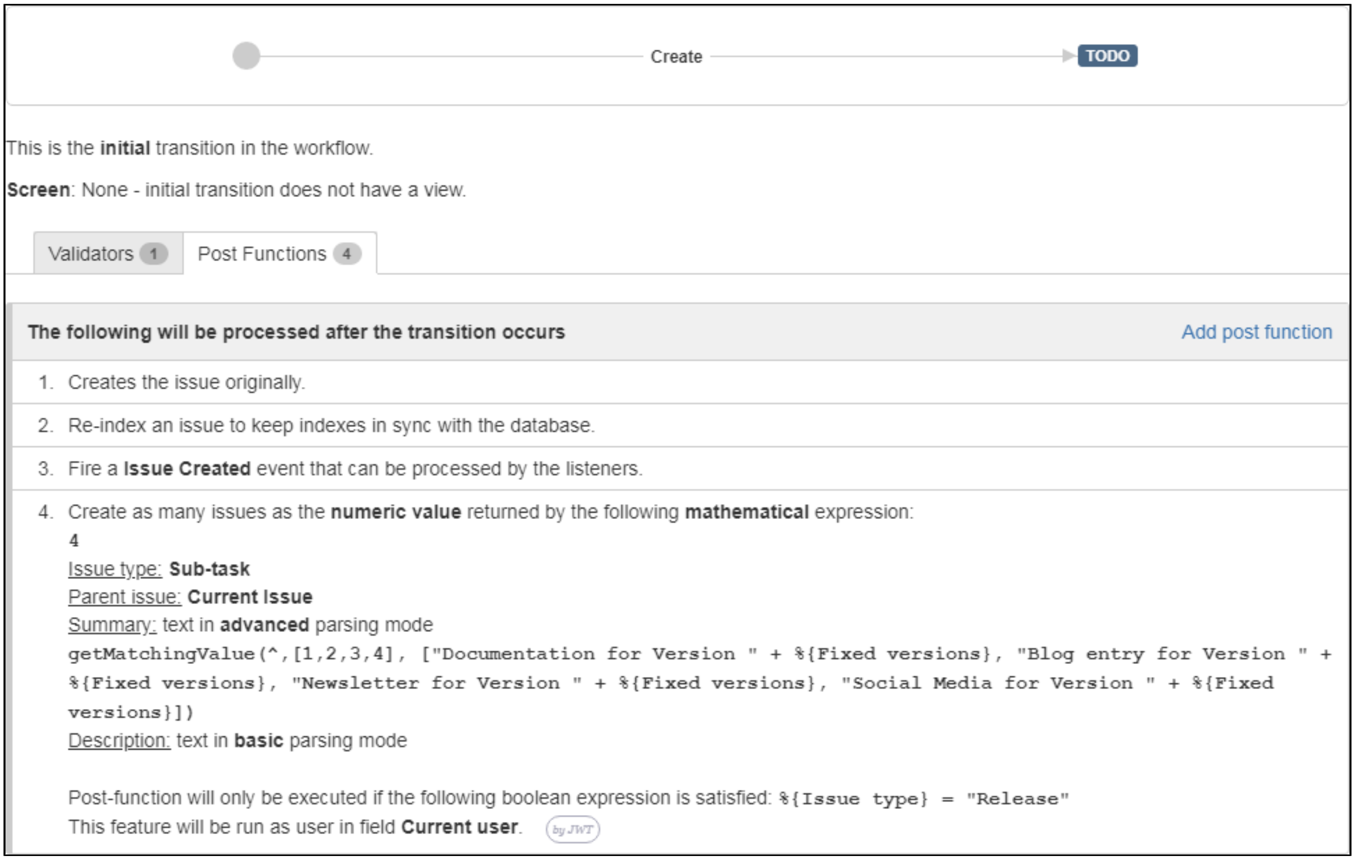
- **%{00074}** is the field code for "**Fix Version/s**"

Since this behaviour should only occur with release issues, the following Conditional execution is added: **`%{00014} = "Release"`**

Note that:

- **%{00074}** is the field code for "**Issue Type**"

Once configured, the transition will look like this:



This is the **initial** transition in the workflow.

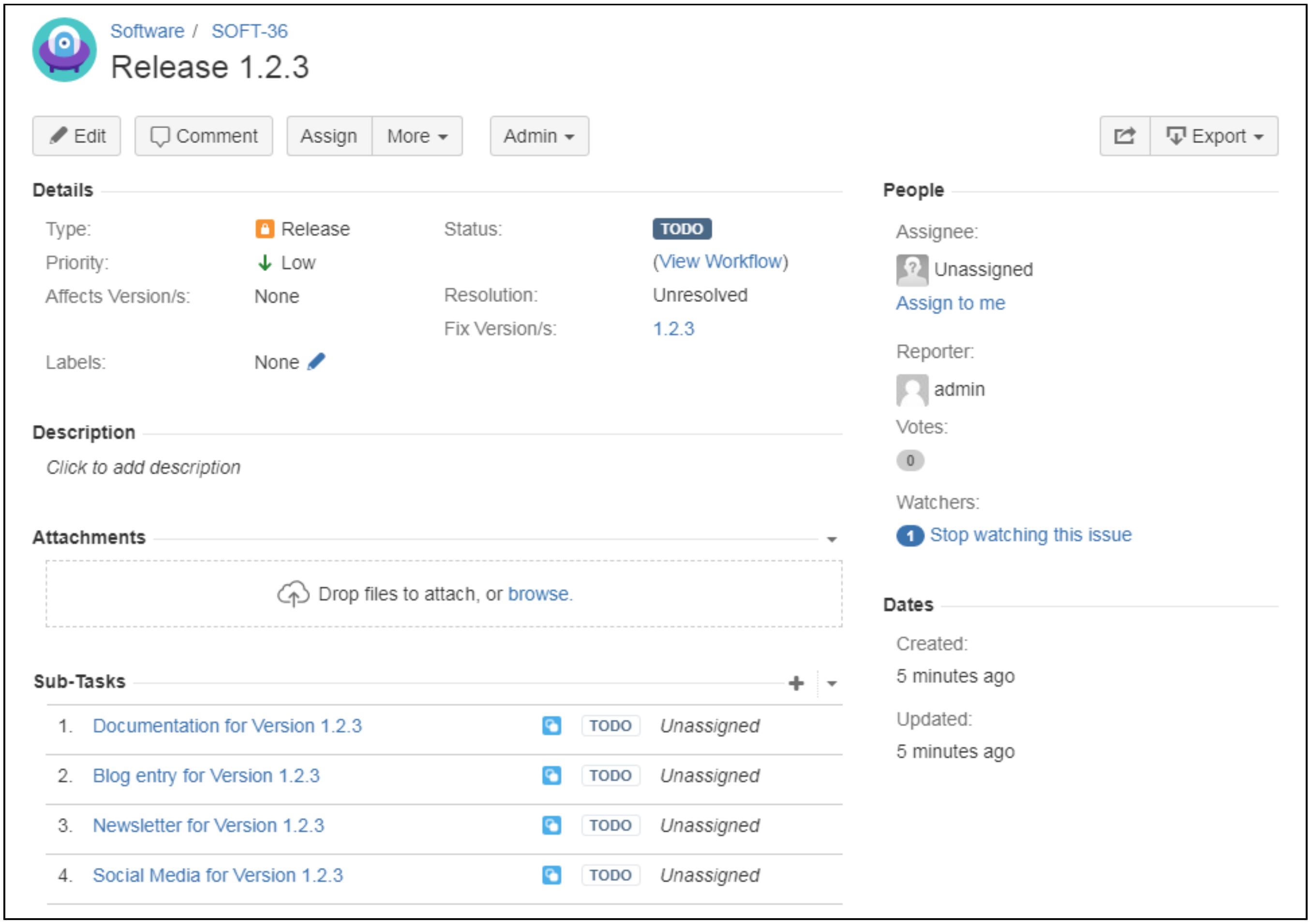
Screen: None - initial transition does not have a view.

The following will be processed after the transition occurs

1. Creates the issue originally.
2. Re-index an issue to keep indexes in sync with the database.
3. Fire a **Issue Created** event that can be processed by the listeners.
4. Create as many issues as the **numeric value** returned by the following **mathematical** expression:
   `4`
   Issue type: **Sub-task**
   Parent issue: **Current Issue**
   Summary: text in **advanced** parsing mode
   ```
   getMatchingValue(^,[1,2,3,4], ["Documentation for Version " + %{Fixed versions}, "Blog entry for Version " + %{Fixed versions}, "Newsletter for Version " + %{Fixed versions}, "Social Media for Version " + %{Fixed versions}])
   ```
   Description: text in **basic** parsing mode

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

After issue creation, the issue will look like this:



Software / SOFT-36

Release 1.2.3

Details

Type:	Release	Status:	TODO
Priority:	Low		(View Workflow)
Affects Version/s:	None	Resolution:	Unresolved
		Fix Version/s:	1.2.3
Labels:	None		

Description

Click to add description

Attachments

Drop files to attach, or browse.

Sub-Tasks

1.	Documentation for Version 1.2.3	TODO	*Unassigned*
2.	Blog entry for Version 1.2.3	TODO	*Unassigned*
3.	Newsletter for Version 1.2.3	TODO	*Unassigned*
4.	Social Media for Version 1.2.3	TODO	*Unassigned*

People

Assignee:
Unassigned
Assign to me

Reporter:
admin

Votes:
0

Watchers:
1 Stop watching this issue

Dates

Created:
5 minutes ago

Updated:
5 minutes ago

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
- Using project properties to calculate custom sequence numbers
    - example
    - post-function
    - calculated-field
    - project-properties
- Set a date based on current date
    - example
    - post-function
- Setting the priority depending on the multiplication of custom fields
    - example
    - calculated-field
    - post-function
- Parse Email adresses to watchers list
    - example
    - post-function
- Set the assignee based on a condition
    - example
    - post-function
- Create a dynamic set of sub-tasks based on checkbox selection with unique summaries
    - example
    - post-function

    - custom-field
    - sub-task
- Create a static set of sub-tasks with unique summaries
    - example
    - post-function
- Triage Jira Service Desk email requests (Move issues)
    - example
    - post-function
    - move
    - transition-issue
- Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress" (Transition issues)
    - example
    - post-function
    - transition
- 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