Create a dynamic set of sub-tasks based on checkbox selection with unique summaries

On this page

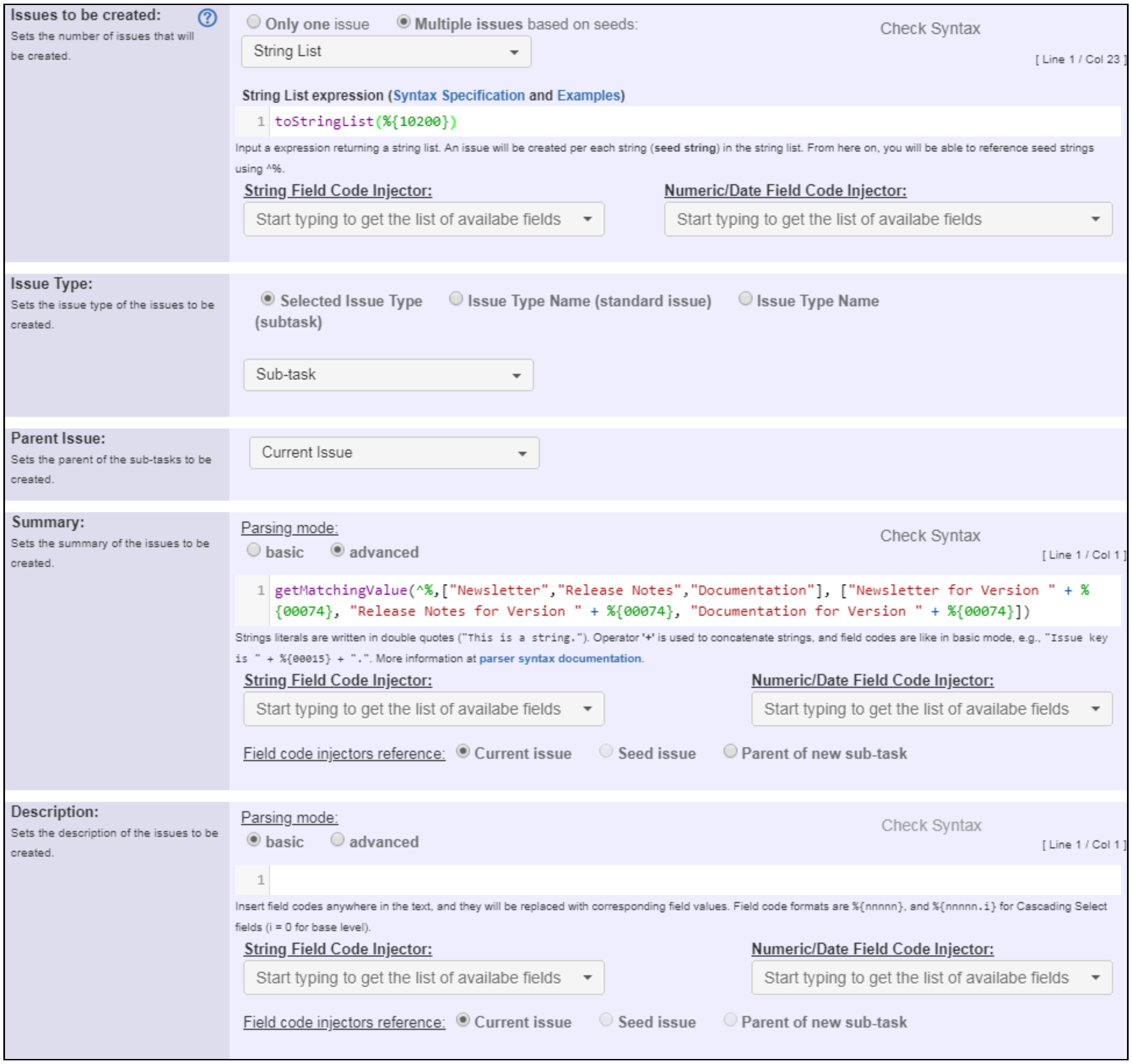
Features used to implement the example

- **Create issues and sub-tasks**

Example: Create a dynamic set of sub-tasks based on checkbox selection with unique summaries

In this example, multiple sub-tasks with unique summaries are added to an issue on issue creation based on the options selected in a custom checkboxes field **"Additional issues"**. Whenever one of the options Newsletter, Release Notes or Documentation is selected, the corresponding sub-task will be created with the correct **"Fixed Version"** added to the summary of each sub-task.

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:]

String List

Check Syntax

[Line 1 / Col 23]

String List expression (Syntax Specification and Examples)

```
toStringList(%{10200})
```

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: 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 (subtask)**] **Issue Type Name (standard issue)** **Issue Type Name**

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]

```
getMatchingValue(^%,["Newsletter","Release Notes","Documentation"], ["Newsletter for Version " + %
{00074}, "Release Notes for Version " + %{00074}, "Documentation 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

Check Syntax

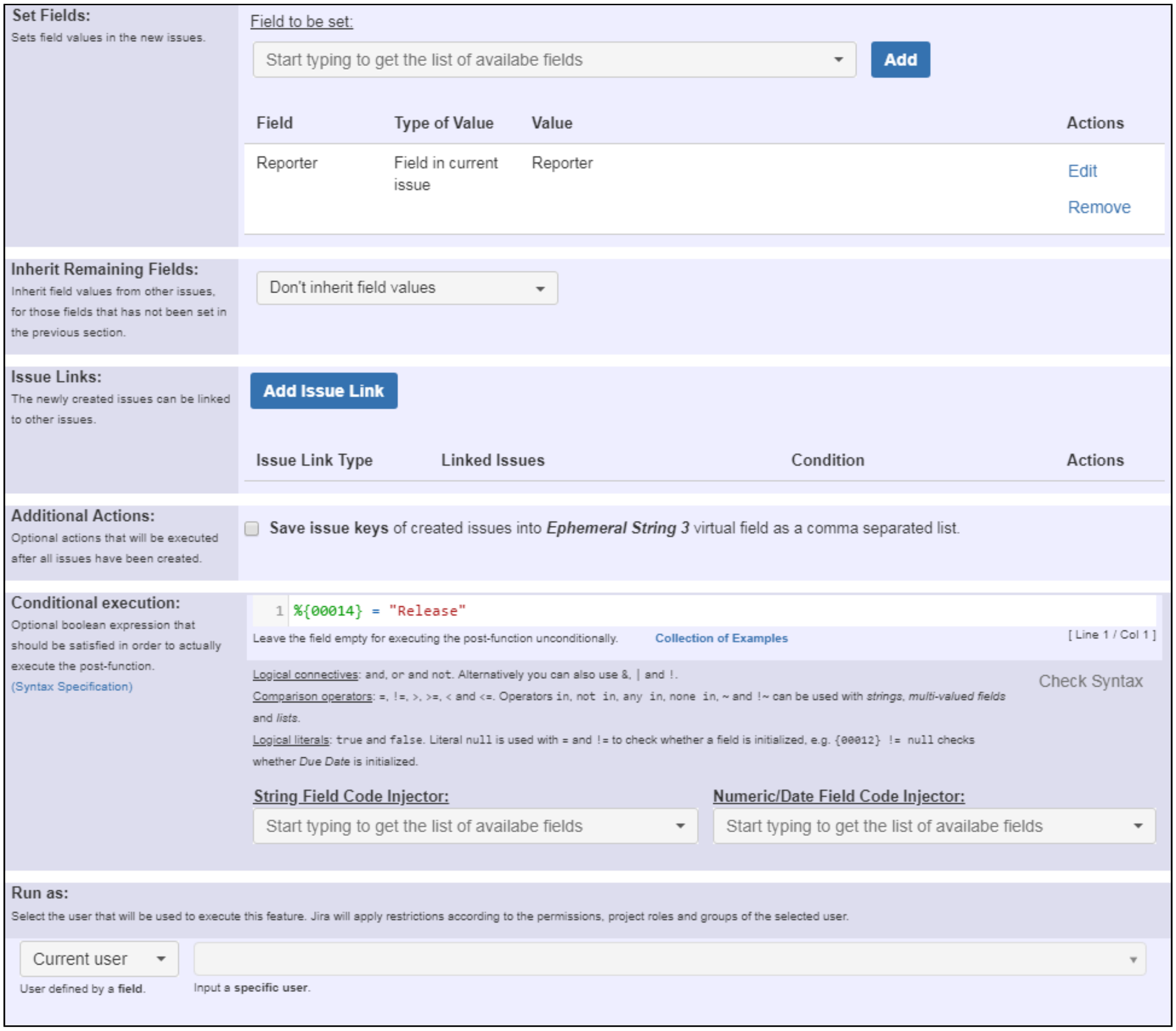
[Line 1 / Col 1]

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
Reporter	Field in current issue	Reporter	Edit Remove

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

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**.

As the number of sub-tasks should be dependant on the option selection, the **Issues to be created** option is set to **String List** with the value**: `toStringList(%{10200})`**

Note that

- **%{10200}** is the field code for the **"Additional options"** custom field (this field code might differ on your instance)

The dynamic summaries are parsed with the following expression:

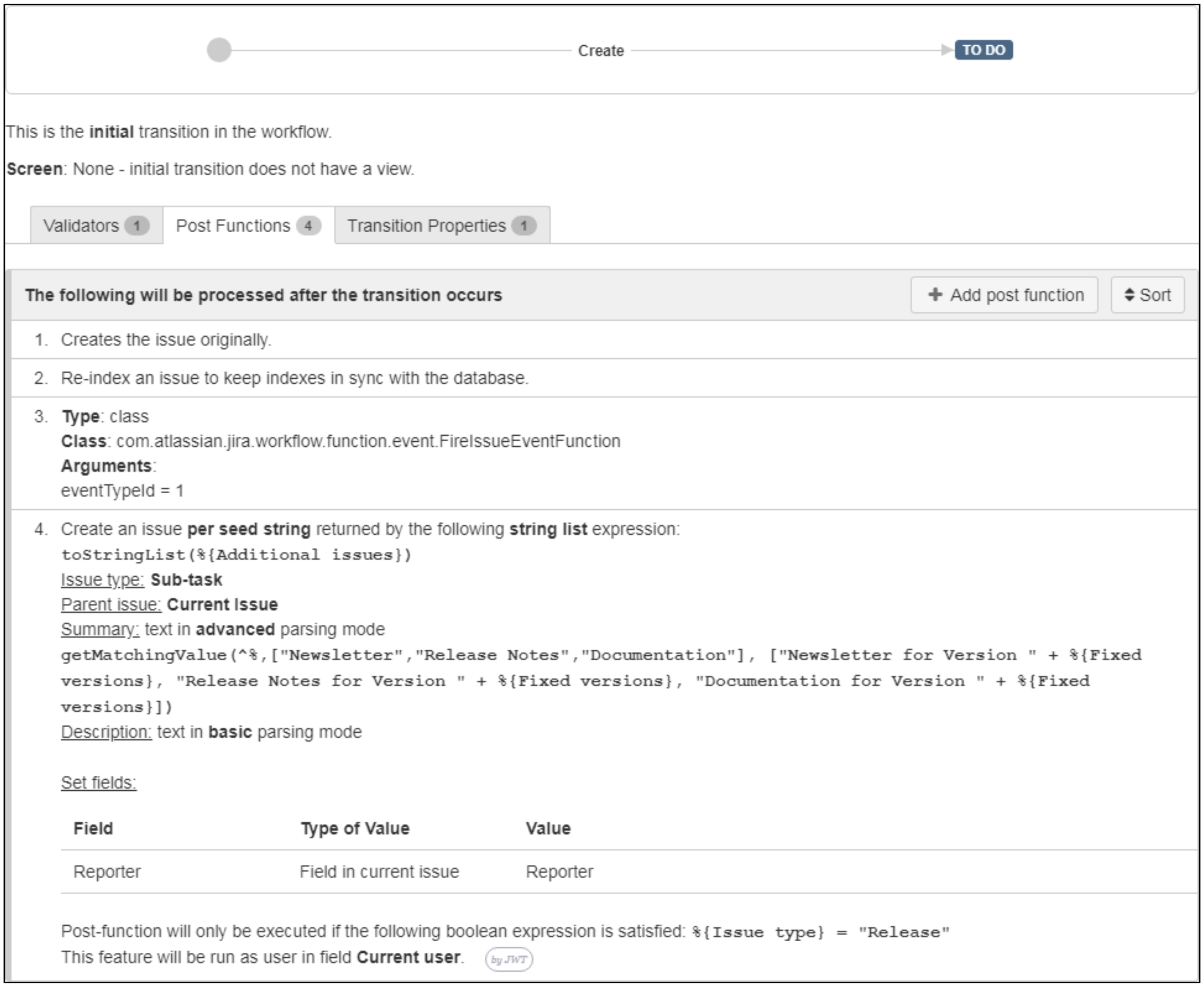
```
getMatchingValue(^%,["Newsletter","Release Notes","Documentation"], ["Newsletter for Version " + %
{00074}, "Release Notes for Version " + %{00074}, "Documentation for Version " + %{00074}])
```

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"`**

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. **Type**: class
   Class: com.atlassian.jira.workflow.function.event.FireIssueEventFunction
   Arguments:
   eventTypeId = 1
4. Create an issue **per seed string** returned by the following **string list** expression:
   `toStringList(%{Additional issues})`
   Issue type: **Sub-task**
   Parent issue: **Current Issue**
   Summary: text in **advanced** parsing mode
   ```
   getMatchingValue(^%,["Newsletter","Release Notes","Documentation"], ["Newsletter for Version " + %{Fixed
   versions}, "Release Notes for Version " + %{Fixed versions}, "Documentation for Version " + %{Fixed
   versions}])
   ```
   Description: text in **basic** parsing mode

   Set fields:

Field	Type of Value	Value
Reporter	Field in current issue	Reporter

   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**. by JWT

See **Result screenshots post-function "Create multiple subtasks based on options checked in a checkbox"**.

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 (Jira Workflow Toolbox (Server / DC) 2.x)
  - example
  - post-function
  - custom-field
  - sub-task
- Add and remove a single or a set of items from multi valued fields (Jira Workflow Toolbox (Server / DC) 2.x)
  - example
  - post-function
  - custom-field
  - issue-links
  - sub-task
- Add all assignees of certain sub-task types to a "Multi-User Picker" custom field (Jira Workflow Toolbox (Server / DC) 2.x)
  - example
  - post-function
  - custom-field
  - sub-task
- Update Cascading Select custom field with a value of the field in parent issue (Jira Workflow Toolbox (Server / DC) 2.x)

- example
    - post-function
    - custom-field
    - sub-task
- Create a sub-task for each user selected in a Multi-User Picker (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - post-function
    - custom-field
    - sub-task
- Validation on sibling sub-tasks depending on issue type and status (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - validator
    - sub-task
    - transition
- Restrict sub-task type creation depending on parent issue status (Jira Workflow Toolbox (Server / DC) 2.x)
    - 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 (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - validator
    - sub-task
- Restrict sub-task type creation depending on parent issue type (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - validator
    - sub-task
- Block a transition until all sub-tasks have certains fields populated (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - condition
    - validator
    - sub-task
    - transition
- Transition sub-tasks when parent is transitioned (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - post-function
    - sub-task
    - transition
    - outdated
- Transition only a sub-task among several ones (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - post-function
    - sub-task
    - transition
    - outdated
- Moving sub-tasks to "Open" status when parent issue moves to "In Progress" (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - post-function
    - sub-task
    - transition
    - outdated
- Moving story to "Ready for QA" once all its sub-tasks are in "Ready for QA" status (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - post-function
    - sub-task
    - transition
    - outdated
- Automatically close resolved sub-tasks when parent issue is closed (Jira Workflow Toolbox (Server / DC) 2.x)
    - example
    - post-function
    - sub-task
    - transition
    - outdated