Create a story for each component in an epic

On this page

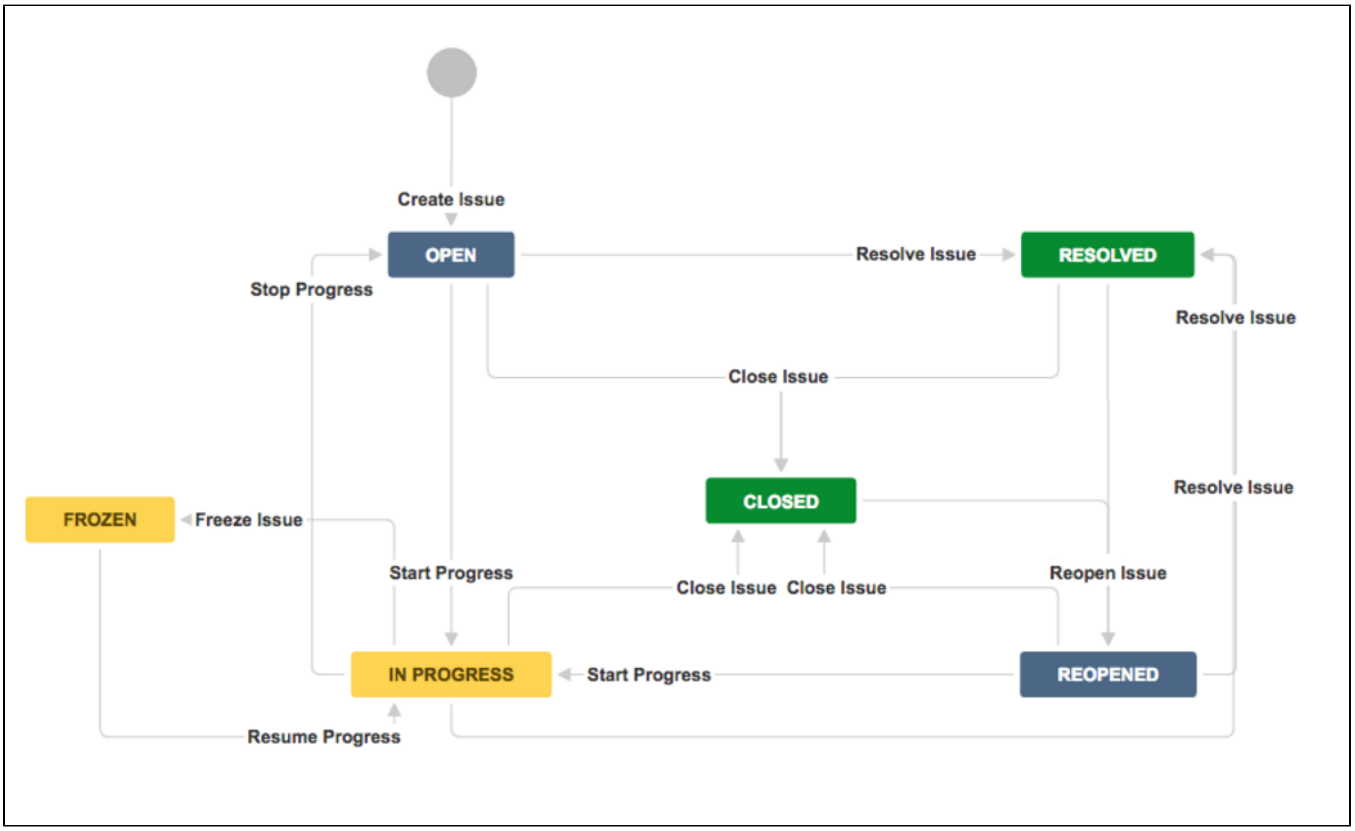
Features used to implement the example

- **Create issues and sub-tasks**

Example: Create a story for each component in an epic

In this example we will show how to use **Create issues and sub-tasks** for creating a Story for each Component selected in Epic issue.

We have 3 components in our project ("**Component A**", "**Component B**" and "**Component C**"). **Epic** issues and **Stories** share the same workflow:



We want to implement a post-function in "**Start Progress**" transition, that will be executed only by **Epic** issues. This post-function will create a **Story** for each **Component** selected in **Epic** story. The newly created **stories** will have the following characteristics:

- **Summary** will be "**Story for component component_name.**"
- **Description** will be "**This is a Story automatically created for component component_name.**"
- **Assignee** will be the leader of the component corresponding to the new story
- **Reporter**: current user, i.e., the user who executes transition "**Start Progress**" in **Epic** issue
- **Components**: we select in each new Story only the component that motivated its creation
- **Priority**: Stories for "**Component A**" and "**Component C**" will have priority **Blocker**, the rest of stories will inherit priority from **Epic** issue
- **Security Level**: all the **Stories** created will inherit the **Security Level** from **Epic** issue

- **Labels**: we will automatically add some labels depending on the **Component** that motivated the creation of the **Story**:
  Component A: `web-resource`.
  Component B: `sales-strategy` and `marketing`.
  Component C: `customer-service` and `user-experience`.
- **Automatic transition** execution after Story creation: Just after a **story** is created, it will be moved through the workflow to status "**Frozen**". To do it we execute the following 2 transitions: "**Start Progress**" and "**Freeze Issue**".
- **Comment**: a comment with the following text will be added: "**Issue frozen just after creation on current_date_time.**"

In order to achieve the described behavior we will use the following configuration:

Set Fields:
Sets field values in the new issues.

Field to be set:

Due date - [Date] | Add

Field	Type of Value	Value	Actions
Assignee	Parsed text (advanced mode)	`nthElement(toStringList(%{Components leaders}), indexOf(^%, toStringList(%{Components})))`	Edit Remove
Reporter	Field in current issue	Current user	Edit Remove
Epic Link	Field in current issue	Epic Name	Edit Remove
Components	Parsed text (advanced mode)	`^%`	Edit Remove
Priority	Parsed text (advanced mode)	`getMatchingValue(^%, ["Component A", "Component C"], ["Blocker", "Blocker", %{Priority}])`	Edit Remove
Security level	Field in current issue	Security level	Edit Remove
New labels	Parsed text (advanced mode)	`getMatchingValue(^%, ["Component A", "Component B", "Component C"], ["web-resource", "sales-strategy marketing", "customer-service user-experience"])`	Edit Remove
Execute transition	Parsed text (basic mode)	Start Progress	Edit Remove
Execute transition	Parsed text (basic mode)	Freeze Issue	Edit Remove
New comment	Parsed text (basic mode)	Issue frozen just after creation on %{Current date and time}.	Edit Remove

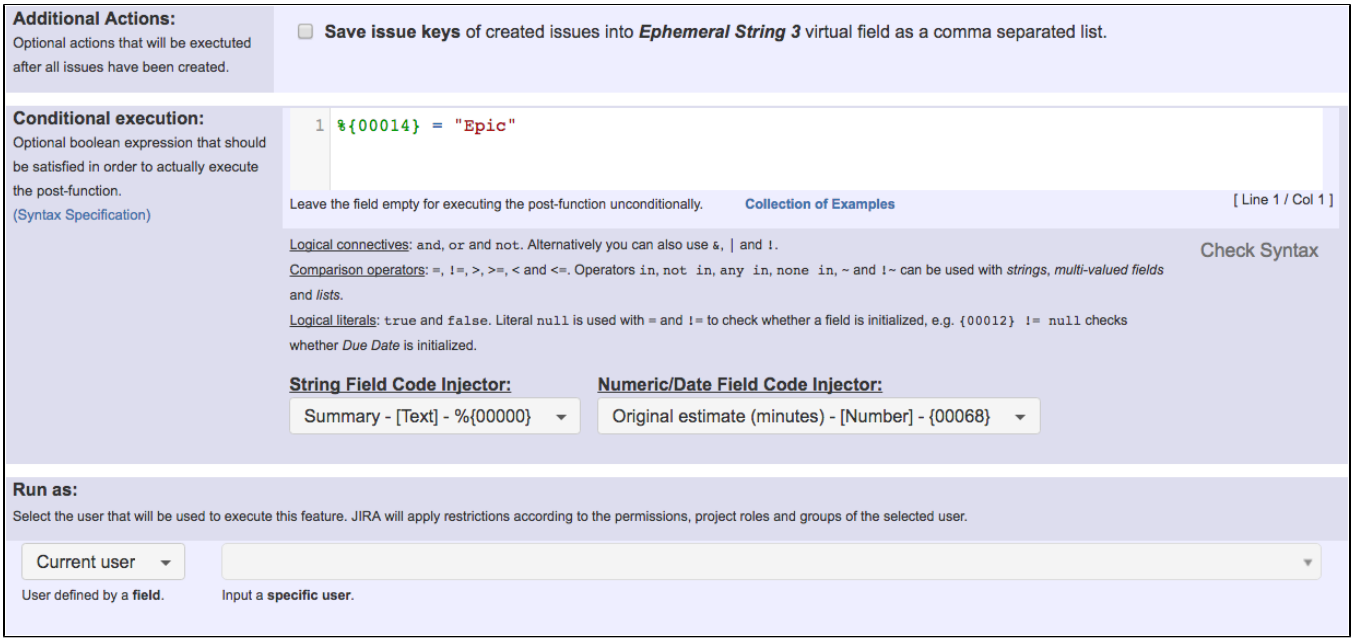
Inherit Remaining Fields:
Inherit field values from other issues, for those fields that has not been set in the previous section.

Don't inherit field values

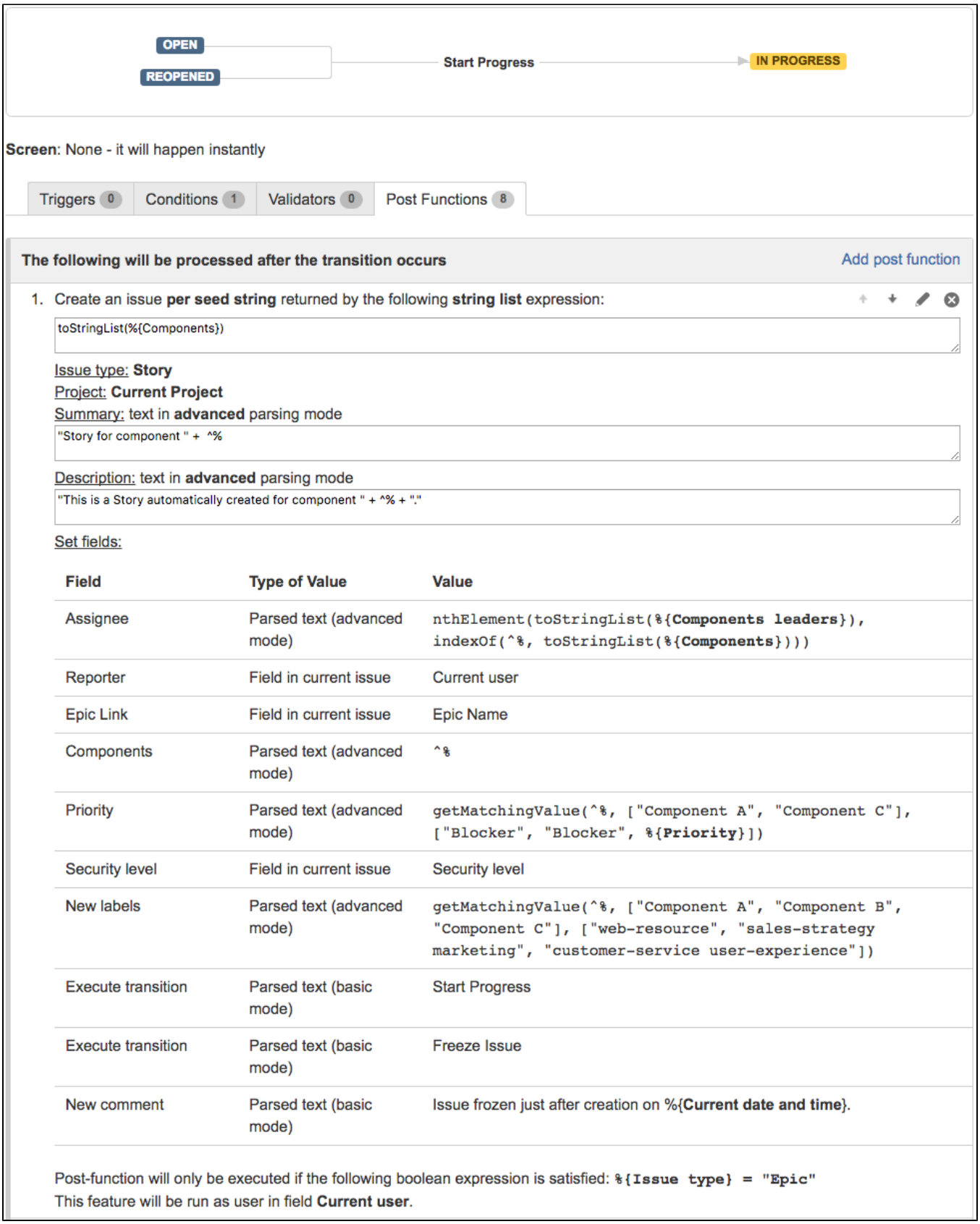
Issue Links:
The newly created issues can be linked to other issues.

Add Issue Link

Issue Link Type	Linked Issues	Condition	Actions



Once configured, transition "**Start Progress**" will look like this:



Result screenshots post-function "Create issues and subtasks" - Create a Story for each Component in Epic

Issues to be created

We use **multiple issue** creation based on **seed strings**. We need to generate a **string list** with each component selected in the **Epic**. To do it we use the following expression: **`toStringList(%{00094})`**, where **%{00094}** is field code for **Components**.

From now on, we can reference each **seed string** in expressions using **^%**. We won't be able to reference **seed strings** when we use **basic parsing mode**.

Summary

We use a parsed text in **advanced** parsing mode with the following string expression: **`"Story for component " + ^%`**.

Description

We use a parsed text in **advanced** parsing mode with the following string expression: **`"This is a Story automatically created for component " + ^% + "."`**.

Assignee

We obtain the component leader for the component in the seed string by using **advanced** parsing mode with the following string expression: **`nthElement(toStringList(%{00125}), indexOf(^%, toStringList(%{00094})))`**, where %**{00125}** returns the **Component leaders** for selected components, and **%{00094}** returns selected **Components** in the same order as the previous field.

Reporter

We use the **Current user**, i.e., the user who is executing the "**Start Progress**" transition in **Epic** issue.

Epic Link

We link newly created **Stories** with **Epic** issue by setting field **Epic Link** in newly created **Stories** with the **Epic Name** in **Epic** issue.

Components

We set field **Components** in newly created **Stories** with the **Component** that motivated the creation of the **Story**, which is in seed string, i.e., **^%**.

Priority

We set field **Priority** using the following text in **advanced** parsing mode: **`getMatchingValue(^%, ["Component A", "Component C"], ["Blocker", "Blocker", %{00017}])`**.

With this expression we are using priority **Blocker** for issues with **Component A** and **Component C**, and priority in **Epic** issue for issues with the rest of components. Priority in **Epic** is represented by field code **%{00017}**.

Security Level

We use the **Security Level** of the **Epic** issue.

New Labels

We create different labels for each component using the following text expression: **`getMatchingValue(^%, ["Component A", "Component B", "Component C"], ["web-resource", "sales-strategy marketing", "customer-service user-experience"])`**.

Automatic Transition Execution

We write twice into virtual field "**Execute transition**" with the **names of the transitions** we want to be executed after each **Story** is created. The order of execution of transitions is the same as the order of writing into virtual field "**Execute transition**". As we are simply writing string literals (transition names), we can use **basic parsing mode**.

New Comment

We write into virtual field "**New comment**" for creating a comment in newly created stories. We use the following text in **basic** parsing mode: **`Issue frozen just after creation on %{00057}.`**, where **%{00057}** is field code for **Current date and time**.

We can use **basic parsing mode** because we are only using field codes to be replaced by their corresponding field values. If we needed to use advanced parsing function, or seed string references (i.e., ^%), we would need to use **advanced parsing mode**.

Conditional Execution

Only when boolean expression **`%{00014} = "Epic"`** is satisfied, post-function will only be executed. This way we can use the post-function in workflows shared with other issue types.

Note that:

- **%{00014}** is field code for i**ssue type**

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