

Add current user to comment

On this page

- [Features used to implement the example](#)
- [Example: Add current user to comment](#)
- [Another implementation](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Copy parsed text to a field](#)

Example: Add current user to comment

I'm looking for a functionality in workflow that automatically added the current user name after a user has enter a comment. Toolbox has to add: Comment was created by : Andreas Beekma

I found the "Copy a parsed text to a field" and I can copy the %{Current user} to every space in JIRA but every "comments" be ignored.

Example: "The following text parsed in basic mode will be copied to Last comment: %{Current user} This feature will be run as Current user."

You can use [Copy a parsed text to a field](#) post-function with the following configuration:

Target field:

Last comment - [Text]

Field to be written with the resulting parsed text.

Don't overwrite target field if it's already set.

Parsing Mode:

Basic **Basic mode:** 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).

Advanced **Advanced mode:** 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](#).

Text to be parsed and then copied to target field: [Syntax Specification](#)

```
1  %{00127} != null ? %{00109} + "\n Comment was created by: " + %{00021} : %{00109}
```

Text to be parsed in this post-function is:

```
%{00127} != null ? %{00109} + "\n Comment was created by: " + %{00021} : %{00109}
```

Note that:

- `%{00127}` is field code for **Transition's comment**
- `%{00109}` is field code for **Last comment**
- `%{00021}` is field code for **Current user's full name**

Once configured, your transition will look like this:

Triggers 0 Conditions 0 Validators 0 Post Functions 6

The following will be processed after the transition occurs [Add post function](#)

1. Set issue status to the linked status of the destination workflow step.
2. Add a comment to an issue if one is entered during a transition.
3. The following text parsed in **advanced** mode will be copied to **Last comment**:
`%{Transition's comment} != null ? %{Last comment} + "\n Comment was created by: " + %
 {Current user's full name} : %{Last comment}`
 This feature will be run as **Current user**.
4. Update change history for an issue and store the issue in the database.
5. Re-index an issue to keep indexes in sync with the database.
6. Fire a **Work Started On Issue** event that can be processed by the listeners.

WARNING: Be careful to insert your post-function AFTER "Add a comment to an issue if one is entered during a transition." post-function, like shown in the former screenshot.

This solution works only for comments introduced in transition's screens, i.e., if the user insert comments by editing the issue, it won't work, since [Copy a parsed text to a field](#) post-function only works when the transition were its inserted is executed.

Another implementation

The special way to use the script in Jira service desk, it doesn't work. When I start the transition via SD Portal. In Jira SD directly it works.

There is a solution for your use case: use [Copy a parsed text to a field](#) post-function with the following configuration:

Target field:

Last comment - [Text]

Field to be written with the resulting parsed text.

Don't overwrite target field if it's already set.

Parsing Mode:

Basic **Basic mode:** 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).

Advanced **Advanced mode:** 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](#).

Text to be parsed and then copied to target field: [Syntax Specification](#)

```
1 matches(%{00109}, ".*Comment was created by:.*") ? %{00109} : %{00109} + "\n Comment was created by: " + %{00164}
```

Note that:

- `%(00109)` is field code for "Last comment"

- `%{00164}` is field code for "Last commenter"

Other examples of that function

Page: [Add all assignees of certain sub-task types to a "Multi-User Picker" custom field](#)

Page: [Add and remove a single or a set of items from multi valued fields](#)

Page: [Add current user to comment](#)

Page: [Add or remove request participants](#)

Page: [Add watchers from a part of the issue summary: "Summary_text - watcher1, watcher2, watcher3, ..."](#)

Page: [Assign issue based on the value of a Cascading Select custom field](#)

Page: [Assign issue to last user who executed a certain transition in the workflow](#)

Page: [Automatically close resolved sub-tasks when parent issue is closed](#)

Page: [Automatically reopen parent issue when one of its sub-tasks is reopened](#)

Page: [Calculate the time elapsed between 2 transition executions](#)

Page: [Close parent issue when all sub-tasks are closed](#)

Page: [Combine the values of several Multi-User picker fields](#)

Page: [Compose a parsed text including the "full name" or a user selected in a User Picker custom field](#)

Page: [Compose dynamic text by inserting field values in a text template](#)

Page: [Copy issue labels to a custom field](#)

Page: [Copy the value of a user property into a user picker](#)

Page: [Create a comment in sub-tasks when parent transitions](#)

Page: [Execute transition in epic](#)

Page: [Getting the number of selected values in a custom field of type Multi Select](#)

Page: [Limit the number of hours a user can log per day](#)

Page: [Make a sub-task's status match parent issue's current status on creation](#)

Page: [Make parent issue progress through its workflow](#)

Page: [Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress"](#)

Page: [Moving story to "Ready for QA" once all its sub-tasks are in "Ready for QA" status](#)

Page: [Parse Email addresses to watchers list](#)

Page: [Parsing text from last comment and appending it to issue's summary](#)

Page: [Remove versions selected in a version picker custom field](#)

Page: [Replace certain issue link types with different ones](#)

Page: [Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status](#)

Page: [Set a Select or Multi-Select field using regular expression to express the values to be assigned](#)

Page: [Set assignee depending on issue type](#)

Page: [Set field depending on time passed since issue creation](#)

Page: [Set priority for issues that have been in a certain status for longer than 24 hours](#)

Page: [Set security level based on groups and project roles the reporter or creator are in](#)

Page: [Transition linked issues in currently active sprint](#)

Page: [Transition only a sub-task among several ones](#)

Page: [Transition parent issue only when certain issue sub-task types are done](#)

Page: [Update Cascading Select custom field with a value of the field in parent issue](#)

Page: [Update checkboxes custom field if a file has been attached during a transition](#)

Page: [Validation on issue attachments](#)

Page: [Validation on MIME types of issue attachments](#)

Page: [Writing a comment to blocked issues when blocking issues are resolved](#)

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 addresses 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](#)

