

Compose dynamic text by inserting field values in a text template

On this page

- [Features used to implement the example](#)
- [Example: Compose dynamic text by inserting field values in a text template](#)
- [Other examples of that functions](#)
- [Related Usage Examples](#)

Features used to implement the example

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

Example: Compose dynamic text by inserting field values in a text template

You can compose dynamic texts by inserting field codes among fixed parts of text. These field codes will be replaced with field values at transition execution time. Use post-function [Copy parsed text to a field](#) to set a field (custom or virtual) with a text template where inserted field codes (using format `%{xxxxx}`) will be replaced with field values at transition execution time. You will often use [Copy parsed text to a field](#) in combination with post-functions [Write field on linked issues or sub-tasks](#), [Update issue fields](#) and [Set a field as a function of other fields](#).

In those cases you select an **ephemeral field** ("Ephemeral string 1", "Ephemeral string 2", ... "Ephemeral string 5") as **Target field** in post-function [Copy parsed text to a field](#), in order to hold temporarily the result text once parsed. Then you can use post-functions [Write field on linked issues or sub-tasks](#) or [Update issue fields](#) to write it into other issues' fields (e.g., linked issues, transitively linked issues, sub-tasks, sibling sub-tasks, or any issue returned by a JQL query). Also post-function [Set a field as a function of other fields](#) can be used in order to conditionally write the parsed text according to simple or complex setting rules that depend on the value of other fields' values.

When [Copy parsed text to a field](#) post-function is executed, field codes (using format `%{xxxxx}`) inserted in the text template are replaced with current fields' values. We want a dynamically compose a text that will be added as a comment at the moment of issue closing.

Using post-function [Copy parsed text to a field](#) we store into auxiliary field "Ephemeral number 1" the result of having parsed the text template we use to compose the comment.

Target field:

Ephemeral string 1 - [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:

[Line 1 / Col 135]

Syntax Specification

Check Syntax

1



"The issue has been closed on " + `%{00057}` + " by " + `%{00021}` + ", and has remained " + `((%{00057} - {00009}) / {DAY})` + " days open."

Text to be parsed used in this example is: "The issue has been closed on " + %{00057} + " by " + %{00021} + ", and has remained " + (({00057} - {00009}) / {DAY}) + " days open."

Note that:

- %{00057} is code for string value of "Current date and time"
- {00057} is code for numeric value of "Current date and time"
- %{00021} is field code of "Current user's full name"
- {00009} is code for numeric value of "Date and time of creation"

Using post-function [Add a comment](#) we insert a comment with text stored in field "Ephemeral string 1", setting desired visibility for the comment, and the user who will appear as comment's author.

Comment's text: 	<input type="radio"/> Parsed Text Entered text will be used as comment's body.	1
Field code injector: Summary - [Text] - %{00000}  Field codes with format %{nnnnn} will be replaced with the corresponding values. Specific levels of Cascading Select fields can be referenced with %{nnnnn.0} for parent level, and %{nnnnn.1} for child level.		
	<input checked="" type="radio"/> Field Value of selected field will be used as comment's body.	Ephemeral string 1 - [Text] 
Comment's author:	Current user - [User]  User that will appear as comment's author. For anonymous comment you should select a non-initialized field.	
Comment visibility:	Jira Service Desk Internal Comment (not visible to customers)  Comment's visibility can be limited to users in a group , in a project role . For Jira Service Desk, internal comment and public comment visibility options are also available.	
Group:	jira-administrators 	
Project Role:	Administrators 	
Notify by email:	<input type="checkbox"/> An Issue Commented event will be triggered.	

Once configured the transition looks like this:

OPEN

IN PROGRESS

REOPENED

Close Issue

CLOSED

Screen: [Resolve Issue Screen](#)

Triggers 0

Conditions 2

Validators 0

Post Functions 7

The following will be processed after the transition occurs

Add post function

- The following text parsed in **advanced** mode will be copied to **Ephemeral string 1**:
"The issue has been closed on " + %{{Current date and time}} + " by " + %{{Current user's full name}} + ", and has remained " + ((({{Current date and time}} - {{Date and time of creation}}) / {{DAY}}) + " days open."
 This feature will be run as user in field **Current user**.
- Create a comment using text in field **Ephemeral string 1**, with Jira Service Desk **internal** visibility (not visible to customers), and **Current user** as author.

More about this example

- Since version **2.1.19** you have available a virtual field called **New comment**. You can use it to insert a new comment using post-function [Copy parsed text to a field](#) simply by selecting it as target field.
- Since version **2.2.1** post-function [Add a comment](#) contains a text box where the text can be directly composed (using basic parsing mode). Since this version of the plugin, the 2 steps approach explained in this example only makes sense when you need to compose a text using **advanced parsing** mode.

Other examples of that functions

[Add a comment](#)

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

Page: [Creating a Jira Service Desk internal comment](#)

Page: [Creating a Jira Service Desk internal comment on linked issues](#)

[Copy parsed text to a field](#)

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](#)

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](#)

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

- 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