

Format field value

This function has been **renamed** with the [JWT 3.0](#) release.

Find the new documentation at:

[Format field value](#)

On this page

- [Purpose](#)
- [Example: Formatting issue summary and replacing contained terms](#)
- [Configuration Parameters](#)
- [Usage Examples](#)
- [Related Features](#)

Purpose

This post-functions allows to give format to current value of a field using a set of formatting operations

Example: Formatting issue summary and replacing contained terms

Now, let's see a configuration example for formatting issue **Summary** for removing undesired white spaces, capitalizing words and replacing the word "**Client**" by "**Customer**":

Field to be formatted:	Summary - [Text] ▼	?
Blanks trimming operations:	☒ Remove all leading white spaces ☒ Remove all trailing white spaces ☒ Leave only one white space between words ☐ Remove all white spaces in the text	
Case changing operations:	☐ Do nothing ☐ Convert all the characters to lower case ☐ Convert all the characters to upper case ☒ Capitalize all the whitespace separated words	
Find & Replace operations:	<u>Substring to be found:</u> ☐ literal text ☒ text literal (ignoring case) ☐ regular expression ☐ regular expression (ignoring case)	
	Client Field code injector: Summary - [Text] - %{00000} ▼ Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding field values.	
	<u>Replacement:</u> ☐ replace first ☒ replace all	
	Customer Field code injector: Summary - [Text] - %{00000} ▼ Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding field values.	
Conditional execution: Optional boolean expression that should be satisfied in order to actually execute the post-function. (Syntax Specification)	1 Leave the field empty for executing the post-function unconditionally. Collection of Examples [Line 1 / Col 1] <u>Logical connectives:</u> and, or and not. Alternatively you can also use &, and !. <u>Comparison operators:</u> =, !=, >, >=, < and <=. Operators in, not in, any in, none in, ~ and != can be used with <i>strings</i>, <i>multi-valued fields</i> and <i>lists</i>. <u>Logical literals:</u> true and false. Literal null is used with = and != to check whether a field is initialized, e.g. {00012} != null checks whether <i>Due Date</i> is initialized. <u>String Field Code Injector:</u> Summary - [Text] - %{00000} ▼ <u>Numeric/Date Field Code Injector:</u> Original estimate (minutes) - [Number] - {00068} ▼	

Once configured, post-function looks like this:

Triggers 0	Conditions 1	Validators 1	Post Functions 9
The following will be processed after the transition occurs Add post function - Field Summary will be formatted using the following operations: - Leading blanks will be removed. - Trailing blanks will be removed. - Repeated blanks between words will be removed. - All the whitespace separated words will be capitalized. - Find all the occurrences of literal text "Client" ignoring the case, and replace with "Customer".			

Configuration Parameters

Blanks trimming operations

- Remove all **leading** white spaces
- Remove all **trailing** white spaces
- Leave only **one white space between words**
- Remove **all** white spaces in the text

Case changing operations

- Convert all the characters **to lower case**
- Convert all the characters **to upper case**
- **Capitalize** all the whitespace separated words

Find & Replace (literal strings or regular expressions)

- **Replace first** found substring found
- **Replace all** found substrings

Supported Field Types

The following field and custom field types can be formatted:

- **Text** fields: Summary, Description, Environment, etc. Also Single-line and Multi-line text custom fields.
 - **Versions** fields: Fix Version/s, Affects Version/s and Version Pickers custom fields (single and multi select).
 - **Labels** fields: issue's Labels field and Labels custom fields.
 - **Selectable fields**: Select List, Multi-Select List, Radio Button
 - **Cascading Select List**
-

Usage Examples

Related Features

- [Copy parsed text to a field](#): with this post-function much more complex text forming operation can be implemented. To do it use **advanced parsing mode**.
- [Parse field for extracting data](#)
- [Regular expression renderer](#)