

Parse field for extracting data

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

Find the new documentation at:

[Copy excerpted value](#)

On this page

- [Purpose](#)
- [Example: Extracting email address from comment and write it to a custom field](#)
- [Usage Examples](#)
- [Related Features](#)

Purpose

Post-function **Parse field for extracting data** is aimed to make it possible for a workflow transition to **extract specific pieces of information** from a field by means of parsing the field's value.

It uses **delimiters** or/and **format specifications** of the value to be extracted for selecting the data fragments that will be extracted from the field's value.

Example: Extracting email address from comment and write it to a custom field

This is an example of the post-function's configuration for extracting an email address from the last comment of an issue and writing it into a custom field called "**Email**". The email is expected to be preceded by string "**SENDER =** ", with an uncertain number of spaces and ignoring the case. This is the configuration to do it:

Source field: Field containing the text that will be parsed for extracting certain value(s).	Last comment - [Text]
Leading delimiter: Text literal or pattern expressed as a regular expression that should be matched immediately before the value to be extracted. By default, delimiters are not included in returned values.	SENDER\s*=\s* ☒ literal text ☐ text literal (ignoring case) ☐ regular expression ☐ regular expression (ignoring case) 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.
Format of the value to be extracted: Format of the expected value to be extracted. Some pattern types require a specification of the format to be entered.	☐ any text between leading and trailing marks (i.e., no pattern) ☐ number ☐ url ☒ email Number, url and email formats are filtered by regular expressions that match the vast majority of possible values, but there might be valid values that don't match these regular expressions. If avoiding any undesired value rejection is a priority, any text should be selected. ☐ regular expression ☐ regular expression (ignoring case) ☐ date-time (specifying date-time formats: Date and Time Patterns) Format types regular expression, regular expression (ignoring case) and date-time require a format specification to be entered. Format specification: 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.
Trailing delimiter: Text literal or pattern expressed as a regular expression that should be matched immediately after the value to be extracted. By default, delimiters are not included in returned values.	☒ text literal ☐ text literal (ignoring case) ☐ regular expression ☐ regular expression (ignoring case) 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.
Target field: Field that will be set with the extracted value(s).	Email - [Text Field (single line)] ☐ Don't overwrite target field if it's already set. ☐ Include the leading delimiter in the output. ☐ Include the trailing delimiter in the output.
Ocurrences to be extracted: Values to be extracted in case that more than one data matching extraction patterns are found.	☒ first occurrence in source field ☐ last occurrence in source field ☐ all occurrences (values are returned as a comma separated list of 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. String Field Code Injector: Summary - [Text] - %{00000} Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

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.

Leading delimiter: **SENDER\s*=\s***

Once configured, post-function's configuration look like this:

The following will be processed after the transition occurs
Add post function

1. Write into **Email** a piece of information extracted from **Last comment**.
Format of value to be extracted is an **email** address.
Leading delimiter is the following **string literal**:

SENDER\s*=\s*

In case of multiple matches extract only **first** occurrence.
This feature will be run as user in field **Current user**.

An example of the result of the execution of this post-function:

Human Resources / HR-71

Bug A

Edit
Comment
Assign
More
Stop Progress
Resolve Issue
Workflow
Admin

Affects Version/s:None
Component/s:None
Labels:None
Email:support@workflowarts.com

Resolution:Unresolved
Fix Version/s:None

Reporter:Administrator
Votes:0

Description

Click to add description

Activity

All
Comments
Work Log
History
Activity

Administrator added a comment - 4 minutes ago - edited

This is a foo message using for testing new "Parse field's value for extracting data" post-function

Sender = support@workflowarts.com

This is the end of this message.

Dates
Created:23/Jun/15 4:39 PM
Updated:Just now

Agile
View on Board

Usage Examples

Page: [Parse description for creating issue links](#)
Page: [Parse summary for setting "Due date"](#)
Page: [Parse summary for setting issue priority](#)
Page: [Parsing text from last comment and appending it to issue's summary](#)

Related Features

- Copy parsed text to a field**: using advanced parsing mode you have a comprehensive set of forming functions at your disposal.
- Format field value**