

# Parsing text from last comment and appending it to issue's summary

## On this page

- [Features used to implement the example](#)
- [Example: Parsing text from last comment and appending it to issue's summary](#)
- [Other examples of that functions](#)
- [Related Usage Examples](#)

## Features used to implement the example

- [Parse field for extracting data](#)
- [Copy parsed text to a field](#)

## Example: Parsing text from last comment and appending it to issue's summary

I wonder if you can help me again with parsing text from an email that we receive (which shows in the comment obviously) to the end of the summary field of the the Jira ticket. An example of reply is:

**Original issue summary:** GCI (GI-20265) - Da Vinci Diamonds Dual - Broken- VIP 13

**Last comment contains the following reply:** Subject: RE: [GCI Jira] (OCC-14813) GCI (GI-20265) - Da Vinci Diamonds Dual - Broken- VIP 13 [ ref: \_00D30iVRS.\_50014smU7L:ref ]

**Desire result of the summary:** [GCI Jira] (OCC-14813) GCI (GI-20265) - Da Vinci Diamonds Dual - Broken- VIP 13 [ ref:\_00D30iVRS.\_50014smU7L:ref ]

- The invariant parts of the prefix is the summary of the original ticket
- The suffix should be everything which is within the brackets

Different example:

- **Original summery:** "Bonus Game Issue. Indiajaya"
- **Response from IGT received as an email which contain the following Subject:** "Subject: RE: [GCI Jira] (OCC-14436) Bonus Game Issue. Indiajaya [ ref:\_00D30iVRS.\_50014sIPqN:ref ]"
- **Desired summary:** "Bonus Game Issue. Indiajaya [ ref:\_00D30iVRS.\_50014sIPqN:ref ]

We will implement it with 2 post-functions:

[Parse field for extracting data](#) with the following configuration:

<b>Source field:</b> Field containing the text that will be parsed for extracting certain value(s).	Last comment - [Text]
<b>Leading delimiter:</b> Text literal or pattern expressed as a <a href="#">regular expression</a> that should be matched immediately <b>before</b> the value to be extracted. Delimiters are not included in returned values.	[ ref: <input checked="" type="radio"/> literal text <input type="radio"/> text literal (ignoring case) <input type="radio"/> regular expression <input type="radio"/> regular expression (ignoring case) <b>Field code injector:</b> Summary - [Text] - %{00000} Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding values.
<b>Format of the value to be extracted:</b> Format of the expected value to be extracted.  Some pattern types require a specification of the format to be entered.	<input checked="" type="radio"/> any text between leading and trailing marks (i.e., no pattern) <input type="radio"/> number <input type="radio"/> url <input type="radio"/> 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, <b>any text</b> should be selected. <hr/> <input type="radio"/> regular expression <input type="radio"/> regular expression (ignoring case) <input type="radio"/> date-time (specifying date-time formats: <a href="#">Date and Time Patterns</a> ) Format types <b>regular expression</b> , <b>regular expression (ignoring case)</b> and <b>date-time</b> require a format specification to be entered. <b>Format specification:</b>  <b>Field code injector:</b> Summary - [Text] - %{00000} Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding values.
<b>Trailing delimiter:</b> Text literal or pattern expressed as a <a href="#">regular expression</a> that should be matched immediately <b>after</b> the value to be extracted. Delimiters are not included in returned values.	:ref ] <input checked="" type="radio"/> text literal <input type="radio"/> text literal (ignoring case) <input type="radio"/> regular expression <input type="radio"/> regular expression (ignoring case) <b>Field code injector:</b> Summary - [Text] - %{00000} Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding values.
<b>Target field:</b> Field that will be set with the extracted value(s).	Ephemeral string 1 - [Text] <input type="checkbox"/> Don't overwrite target field if it's already set.
<b>Ocurrences to be extracted:</b> Values to be extracted in case that more than one data matching extraction patterns are found.	<input type="radio"/> first occurrence in source field <input checked="" type="radio"/> last occurrence in source field <input type="radio"/> all occurrences (values are returned as a comma separated list of values)
<b>Run as:</b> 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 certain <b>fixed user</b> . Expected value is a <b>user name</b> , don't confuse it with user's full name.

Copy a parsed text to a field with the following configuration:

Target field:

Summary - [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  %{00000} + ({%{00000} !~ "[ :ref" AND %{00061} != null ? " [ ref:" + %{00061} + ":ref ]" : "")

```

Text to be parsed is:

```
%{00000} + ({%{00000} !~ "[ ref:" AND %{00061} != null ? " [ ref:" + %{00061} + ":ref ]" : "" )
```

Once configured, your post-functions will look like this:

The following will be processed after the transition occurs

Add post function

1. Write into **Ephemeral string 1** a piece of information extracted from **Last comment**.

Format of value to be extracted: **any text**

Leading delimiter: `[ ref:` string literal.

Trailing delimiter: `:ref ]` string literal.

In case of multiple matches: extract **last** occurrence.

This feature will be run as user in field **Current user**.

2. The following text parsed in **advanced** mode will be copied to **Summary**:

`%{Summary} + ({%{Summary} !~ "[ :ref" AND %{Ephemeral string 1} != null ? " [ ref:" + %`

`{Ephemeral string 1} + ":ref ]" : "" )`

This feature will be run as user in field **Current user**.

## Other examples of that functions

### Parse field for extracting data

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](#)

### Copy parsed text to a field

## 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](#)

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

- 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