

Writing a comment to blocked issues when blocking issues are resolved

On this page

- [Features used to implement the example](#)
- [Example: Writing a comment to blocked issues when blocking issues are resolved](#)
- [Other examples of that functions](#)
- [Related Usage Examples](#)

Features used to implement the example

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

Example: Writing a comment to blocked issues when blocking issues are resolved

Is there any way to write a comment, for instance, "All blockers have been resolved, you may proceed with this issue", to the issue, say ABC-1234, blocked by the current issue, which is now being resolved, if ABC-1234 won't have unresolved blocking issues right after the current issue is resolved?

Yes, you can implement that behavior adding 3 post-functions to **"Resolve Issue"** transition in your blocker issues workflow:

Post-function [Copy a parsed text to a field](#) with the following configuration:

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:

Syntax Specification

1

All blockers have been resolved, you may proceed with this issue.

Post-function [Copy a parsed text to a field](#) with the following configuration:

Target field:

Ephemeral string 2 - (Text) - [00062]

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 Self

☒ 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., "Is

Text to be parsed and then copied to target field:

1

toString(filterByPredicate(linkedIssues("blocks"), count(filterByResolution(linkedIssues("is blocked by", ^{%00015}), " ")) = 0))

2

3

4

5

6

Text to be parsed in advanced mode is:

```
toString(filterByPredicate(linkedIssues("blocks"), count(filterByResolution(linkedIssues("is blocked by", ^{%00015}), " ")) = 0))
```

Note that:

- ^{%00015} is field code for virtual field "Issue Key" in issues returned by function `linkedIssues("blocks")`

Post-function [Update issue fields](#) with the following configuration:

Target fields and Source values: ?

Select the target fields that will be set and the source values for each of them.

Target field:

Summary - [Text]

Add

Add a field to be set in selected issues.

Target Field	Type of Value	Source Value	Don't Overwrite	Actions
New comment	Field in current issue	Ephemeral string 1		<a>Edit <a>Remove

Target Issues:

Select the issue(s) to be updated.

Issue Selection Mode:

☐ Current Issue
 ☐ Parent Issue
 ☐ Linked Epic
 ☐ Linked Issues
 ☐ Subtasks
 ☐ Sibling Subtasks
 ☐ Issues under Epic
 ☐ Sibling issues under Epic
 ☒ JQL Query
 ☐ Issue List

[Line 1 / Col 23]

1

issuekey in ({00062})

String Field Code Injector:

Summary - [Text] - %{00000}

Numeric/Date Field Code Injector:

Original estimate (minutes) - [Number] - {00068}

Check Syntax

- Field codes with format %{nnnn} may be inserted in the JQL Query, and will be replaced with field values at runtime. Most times it's a good idea to write field codes between double quotes (e.g. "{00001}"), since field values may contain blank spaces that will produce JQL parsing errors at runtime.

- Cascading Select fields and Multi-level Cascading Select fields specific levels can be referenced with %{nnnn.0} for parent level, %{nnnn.1} for child level, etc.

Additional options:

☐ Enable email notifications on issues to be written, according to applicable notification scheme.

☐ Update issue immediately after field writing. A specific entry will be created in issue history for each field writing.

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]

Logical connectives: and, or and not. Alternatively you can also use &, | and !.

Comparison operators: =, !=, >, >=, < and <=. Operators in, not in, any in, none in, ~ and !~ can be used with strings, multi-valued fields and lists.

Logical literals: true and false. Literal null is used with = and != to check whether a field is initialized, e.g. {00012} != null checks whether Due Date is initialized.

String Field Code Injector:

Summary - [Text] - %{00000}

Numeric/Date Field Code Injector:

Original estimate (minutes) - [Number] - {00068}

Check Syntax

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.

Once configured, transition "Resolve Issue" looks like this:

IN PROGRESS

Resolve

RESOLVED

Screen: [Resolve Issue Screen](#)

Conditions 0

Validators 0

Post Functions 3

The following will be processed after the transition occurs

Add post function

- The following text parsed in **basic** mode will be copied to **Ephemeral string 1**:
All blockers have been resolved, you may proceed with this issue.
 This feature will be run as user in field **Current user**. by JWT
- The following text parsed in **advanced** mode will be copied to **Ephemeral string 2**:
`toString(filterByPredicate(linkedIssues("blocks"), count(filterByResolution(linkedIssues("^Issue key", "is blocked by"), "")) = 0))`
 This feature will be run as user in field **Current user**. by JWT
- Update the following target issues:
JQL query
`issuekey in ({Ephemeral string 2})`
Target fields and Source values:

Target Field	Type of Value	Source Value	Don't Overwrite
New comment	Field in current issue	Ephemeral string 1	

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

Other examples of that functions

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](#)

Related Usage Examples

- [Validate only issue links created in transition screen](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
- [Require issue link when resolving as duplicate](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
- [Ensure that all issues linked with a certain issue link type have "Due Date" field set](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
- [Block an epic's transition depending on linked issues status and due date](#)
 - [example](#)

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

Update issue fields

- validator
 - issue-links
 - transition
- Writing a comment to blocked issues when blocking issues are resolved
 - example
 - post-function
 - issue-links
- Add and remove a single or a set of items from multi valued fields
 - example
 - post-function
 - custom-field
 - issue-links
 - sub-task
- Block or hide a transition for an issue depending on its issue links
 - example
 - validator
 - issue-links
 - transition
- Prevent transitioning when there is a blocking issue
 - example
 - validator
 - issue-links
 - sub-task
 - transition
- Prevent issue from being "Closed" if blocking issues aren't yet closed
 - example
 - validator
 - issue-links
 - transition
- Block creation of issue type X if it has not been linked with link type Y to issue type Z on the "Create Issue" screen
 - example
 - validator
 - issue-links
- Enforce linked issues in a specific project to be "Closed" before closing issue
 - example
 - validator
 - issue-links
 - transition
- Prevent issue from being closed if it has links of type "is blocked by" to open issues
 - example
 - condition
 - validator
 - issue-links
 - transition
- Prevent issue from moving forward if it's dependent on non-accepted tickets
 - example
 - validator
 - issue-links
 - transition
- Parse description for creating issue links
 - example
 - post-function
 - issue-links
- Propagate highest priority from blocked issues to blocking issues
 - example
 - post-function
 - issue-links