

Write field on linked issues or sub-tasks

On this page

- [Purpose](#)
- [Example: Add attachments from current issue to cloning issues](#)
- [Configuration Parameters](#)
- [Usage Examples](#)
- [Related Features](#)

Purpose

This post-function allow setting the value of a field in **linked issues** or **sub-tasks** using any of the following **source values**:

- **Field** in current issue: the value a field in current issue.
- **Parsed text (basic mode)**: a text composition where value of fields in current issue can be inserted.
- **Parsed text (advanced mode)**: a string expression where we can use values of fields in current issue (syntax `%{nnnnn}`), and in linked issues and sub-tasks (syntax `^{nnnnn}`). Here we can use all the functions available in the [Expression Parser](#).
- **Math** or **Date-Time** expression: an expression returning a numeric value where we can use values of fields in current issue (syntax `{nnnnn}`), and in linked issues and sub-tasks (syntax `^{nnnnn}`). Here we can use all the functions available in the [Expression Parser](#).

Issues that can be written:

- **Linked Issues**: issues linked to current issue
- **Sub-task**: current issue's sub-tasks
- **Sibling Sub-tasks**: when current issue is a sub-task, its parent's other sub-tasks

Example: Add attachments from current issue to cloning issues

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
Attachments (only new attachments will be added)	Field in current issue	Attachments		Edit Remove

Filtering by issue link type:

- ☐ is blocked by
- ☐ blocks
- ☒ is cloned by
- ☐ clones
- ☐ is duplicated by
- ☐ duplicates
- ☐ relates to
- ☐ relates to

Only issues linked to current issue by selected issue link types will be written.







Write also subtasks fulfilling condition on issue type, status and project:☐

This option only makes sense when current issue itself is not a subtask.

Write also sibling subtasks fulfilling condition on issue type, status and project:☐















Sibling subtasks are understood as subtasks with the same parent as current issue. This option only makes sense when current issue is itself a subtask.

Filtering linked issues or subtasks by issue type:

- ☐  Bug
- ☐  Epic
- ☐  Improvement
- ☐  New Feature
- ☐  Story
- ☐ ☒ Task
- ☐  Sub-task

Selected issue types will be written, but if you don't select any, it won't be applied any filter by issue type. In that case all the issue types will be written.

Filtering linked issues or subtasks by status:

- ☒  Open
- ☒  In Progress
- ☒  Reopened
- ☐  Resolved
- ☐  Closed
- ☒  To Do
- ☐  Done
- ☐  Acceptance
- ☐  Fail
- ☐  Pass
- ☐  Retest
- ☐  Active
- ☐  Inactive
- ☐  Cancelled

Selected statuses will be written, but if you don't select any, it won't be applied any filter by status. In that case issues in any status will be written.

Linked issues or subtasks belong to:

- ☒ any project
- ☐ current project
- ☐ any but current project

Filtering by field values:

Optional boolean expression that should be satisfied by linked issues and subtasks. [\(Syntax Specification\)](#)

1

Leave field empty for no filtering.

[Line 1 / Col 1]

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

Comparison operators: **=**, **!=**, **>**, **>=**, **<** and **<=**. Operators **!~**, **in**, **not in**, **any in** and **none in** 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.

[Check Syntax](#)

String Field Code Injector:

Summary - [Text] - %{00000} ▾

Field Code for **Current Issue**

Field Code for **Linked Issues / Subtasks**

Numeric/Date Field Code Injector:

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

Field Code for **Current Issue**

Field Code for **Linked Issues / Subtasks**

Example 1: `{00012} <= ^{00012}` will require that linked issues and subtasks have **Due Date** equal or later than current issue's **Due Date**.

Example 2: `!{00074} ~ ^{00074} AND ^{00017} in ["Blocker", "Critical"]` will require that linked issues and subtasks have **Fixed versions** contained in current issue's **Fixed versions** and **Priority** is **Blocker** or **Critical**.

Write linked issues and subtasks recursively:



Issues and subtasks transitively linked will also be written, provided they fulfill stated filtering conditions. Issues are written recursively without depth limit, but each selected issue is written only once.

Write linked issues and subtasks recursively:

☐

Issues and subtasks transitively linked will also be written, provided they fulfill stated filtering conditions. Issues are written recursively without depth limit, but each selected issue is written only once.

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:

Numeric/Date Field Code Injector:

Summary - [Text] - %{00000}

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.

Once configured, post-function looks like this:

The following will be processed after the transition occurs

Add post function

1. Value of field **Attachments** in current issue will be copied to field **Attachments (only new attachments will be added)** in linked issues or subtasks filtering by:

Inward issue link types: **is cloned by**.

Outward issue link types: **none**

Subtasks won't be written.

Sibling subtasks won't be written.

Issue types: **any**

Statuses: **To Do, Open, Reopened and In Progress.**

Linked issues or subtasks may belong to **any** project.

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

Configuration Parameters

Filtering Conditions

Issues to be written can be filtered by:

- **Issue link type:** only for linked issues.
- **Issue types:** if no issue type is selected, then no filter by issue type is applied.
- **Statuses:** if no status is selected, then no filter by status is applied.
- **Project:** three possible options are available ("**any project**", "**current project**" and "**any but current project**").
- **Field values:** when a [boolean expression](#) is entered, only those issues fulfilling the expression are selected. In this expression we use ^ prefix for field values in foreign issues (linked issues, sub-tasks and sibling sub-tasks): **^{nnnnn}** and **^{nnnnn}**, while field codes

without prefix correspond to current issue's field values.

Example 1: `boolean condition {00012} <= ^{00012}` will require that issues have "**Due Date**" equal or later than current issue's "**Due Date**".

Example 2: `boolean condition %{00074} ~ ^%{00074} AND ^%{00017} in ["Blocker", "Critical"]` will require that issues have "**Fixed versions**" contained in current issue's "**Fixed versions**" and that "**Priority**" has values "Blocker" or "Critical".

Additional Options

- **Don't overwrite target field if it's already set:** when checked, this parameter will make the post-function do nothing in case target field is not empty in current issue.
 - **Write linked issues and subtasks recursively:** transitively linked issues and its subtasks are also selected provided they fulfill filtering conditions. This recursive operation is performed with no depth limit, but each selected issue is written only once.
 - **Run as:** Jira user post-function is going to be executed as. This parameter can be set to a **fixed user** (e.g. "john.nash"), or to a **user field** (e.g. "Reporter", "Assignee", etc). This parameter is important when we have permission or security restrictions that might prevent fields from being read or written.
-

Usage Examples

Page: [Add and remove a single or a set of items from multi valued fields](#)

Page: [Automatically become watcher of every issue blocking an issue assigned to you](#)

Page: [Automatically close resolved sub-tasks when parent issue is closed](#)

Page: [Automatically resolve an epic when all its stories are resolved](#)

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

Page: [Copy "Due date" into a date type custom field in a linked issue if it's greater than current issue's "Due date"](#)

Page: [Copy attachments from one issue to another](#)

Page: [Create a comment in sub-tasks when parent transitions](#)

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

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

Page: [Execute transition in epic](#)

Page: [Make linked issues, sub-tasks and JQL selected issues progress through its workflows](#)

Page: [Moving sub-tasks to "Open" status when parent issue moves to "In Progress"](#)

Page: [Sum sub-task's "Time Spent" \(work logs\) and add it to a certain linked issue](#)

Page: [Transition sub-tasks when parent is transitioned](#)

Related Features

- [Read fields from linked issues or sub-tasks](#)
- [Update issue fields](#)
- [Read field from issues returned by JQL query or issue list](#)