

Propagate highest priority from blocked issues to blocking issues

On this page

- [Features used to implement the example](#)
- [Example: Propagate highest priority from blocked issues to blocking issues](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Read fields from linked issues or sub-tasks](#)

Example: Propagate highest priority from blocked issues to blocking issues

We want that current issue inherits the highest priority among those issues being blocked by her. Issues blocked by current issue are those issues linked (directly or transitively) to current issue using an "**is blocked by**" issue link, that is:

- current issue **blocks** another issue
- another issue **is blocked by** current issue

We insert post-function [Read fields from linked issues or sub-tasks](#) in "**Create Issue**" transition for setting current issue's priority using linked issues priority.

We can also include this post-function in other transitions of the workflow, in order to refresh the priority in any other moment of life cycle of the issue.

Source value that will be written into target field: ?	Select a source type: ☒ field in linked issues or subtasks ☐ parsed text (advanced mode) ☐ math or date-time expression Priority - [Issue priority] ▾ The value of selected field in linked issues or subtasks will be used as source.
Target field to be written in current issue:	Priority - [Issue priority] ▾ ☐ Don't overwrite target field if it's already set.
Value to write in target field in case source field is of type Number, Priority, Date or Date and Time:	highest priority ▾
Filtering by issue link type:	☐ is blocked by ☒ blocks ☐ is cloned by ☐ clones ☐ is duplicated by ☐ duplicates ☐ has Epic ☐ is Epic of ☐ is caused by ☐ causes ☐ relates to ☐ relates to Only issues linked to current issue by selected issue link types will be read.
Read also subtasks fulfilling condition on issue type, status and project:	☐ This option only makes sense when current issue itself is not a subtask.
Read 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:

- ☐  Epic
- ☐  Story
- ☐  Test Plan
- ☐  Bug
- ☐  New Feature
- ☐  Task
- ☐  Improvement
- ☐  QA Sub-task
- ☐  Sub-task

Selected issue types will be read, 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 read.

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 read, 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 read.

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*.

Read linked issues and subtasks recursively:

☒

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

Read also current issue:

☒

Current issue will be included in the issue selection, i.e., current issue's field value will also be read.

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.

Check Syntax

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**.

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

Create

→

TO DO

This is the **initial** transition in the workflow.

Screen: None - initial transition does not have a view.

Validators 1

Post Functions 4

The following will be processed after the transition occurs

Add post function

- Highest priority** of fields **Priority** in linked issues or subtasks will be copied to field **Priority** in current issue, filtering issues by:

Inward issue link types: **none**
 Outward issue link types: **blocks**.
Subtasks won't be read.
Sibling subtasks won't be read.
 Issue types: **any**
 Statuses: **any**
 Linked issues or subtasks may belong to **any** project.
 Linked issues or subtasks will be **read recursively**.
Current issue will also be read.
 This feature will be run as user in field **Current user**.

Other examples of that function

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: [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: [Make an issue inherit highest priority among those of linked issues](#)

Page: [Propagate highest priority from blocked issues to blocking issues](#)

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

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](#)
 - [validator](#)
 - [issue-links](#)
 - [transition](#)
- [Add and remove a single or a set of items from multi valued fields](#)
 - [example](#)
 - [post-function](#)
 - [custom-field](#)
 - [issue-links](#)
 - [sub-task](#)
- [Writing a comment to blocked issues when blocking issues are resolved](#)
 - [example](#)
 - [post-function](#)
 - [issue-links](#)

- Prevent issue from moving forward if it's dependent on non-accepted tickets
 - example
 - validator
 - issue-links
 - transition
- Enforce linked issues in a specific project to be "Closed" before closing issue
 - example
 - validator
 - issue-links
 - transition
- 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
- Prevent issue from being closed if it has links of type "is blocked by" to open issues
 - example
 - condition
 - validator
 - issue-links
 - transition
- Transition linked issues in currently active sprint
 - example
 - post-function
 - issue-links
 - transition
- Automatically become watcher of every issue blocking an issue assigned to you
 - example
 - post-function
 - issue-links