

# Transition sub-tasks when parent is transitioned

## On this page

- [Features used to implement the example](#)
- [Example: Automatically transition sub-tasks to certain status when parent issue is transitioned to another status](#)
- [Other examples of that functions](#)
- [Related Usage Examples](#)

## Features used to implement the example

- [Write field on linked issues or sub-tasks](#)
- Virtual fields "**Issue status**" or "**Execute transition**": writing into these fields the **name of an status** or the **name of a transition** respectively, makes an issue progress through its workflow, provided conditions and validators in the transition are satisfied



This example is considered **outdated**. Please head over to the one using our [Transition issues](#) post function [here](#).

## Example: Automatically transition sub-tasks to certain status when parent issue is transitioned to another status

We are going to explain how we can automatically move all sub-tasks to a certain status in sub-task's workflow, when parent issue is moved to another status in parent's issue workflow.

In this example we suppose that both parent's and sub-task's workflows have a status called "**Cancelled**". We want to automatically cancel all sub-tasks when parent issue is cancelled. We also suppose that there is a global transition called "**Cancel Issue**" in both workflows (parent's and sub-task's) which has as target status "**Cancelled**".

To implement this use case we insert [Write field on linked issues or sub-tasks](#) post-function in transition "**Cancel Issue**" in parent's workflow, using the following configuration:

Source value that will be written into target field:



**Select a source type:**

field in current issue  parsed text (basic mode)  parsed text (advanced mode)  math or date-time

[ Line 1 / Col 14 ]

expression

1 Cancel Issue

Field codes with format `{nnnn}` will be replaced with the corresponding field values. With **Cascading Select** fields use `{nnnn.0}` and `{nnnn.1}` for referencing **base** level and **child** levels respectively. [Check Syntax](#)

**String Field Code Injector:**

Summary - [Text] - %{00000} ▾

Field Code for **Current Issue**

Field Code for **Linked Issues / Subtasks**

**Numeric/Date-Time Field Code Injector:**

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

Field Code for **Current Issue**

Field Code for **Linked Issues / Subtasks**

Target field that will be set in linked issues or subtasks:

Execute transition - [Workflow transition] ▾

Don't overwrite target field if it's already set.

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 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:**

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

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](#))

[ Line 1 / Col 1 ]

Leave field empty for no filtering.

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.

**String Field Code Injector:**

**Numeric/Date Field Code Injector:**

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*.

[Check Syntax](#)

**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.

**Conditional execution:**  
Optional boolean expression that should be satisfied in order to actually execute the post-function. ([Syntax Specification](#))

[ Line 1 / Col 1 ]

Leave the field empty for executing the post-function unconditionally. [Collection of Examples](#)

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:**

[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.

User defined by a field.

Input a specific user.

Once configured, the transition looks like this:

DONE
Cancel Issue
CANCELLED

**Screen:** None - it will happen instantly

Triggers 0
Conditions 1
Validators 1
Post Functions 8

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

1. Text parsed in **basic** mode **Cancel Issue** will be copied to field **Execute transition** in linked issues or subtasks filtering by:
  - Inward issue link types: **none**
  - Outward issue link types: **none**
  - Subtasks** fulfilling conditions on issue type, status and project **will be written**.
  - Sibling subtasks won't be written**.
  - Issue types: **any**
  - Statuses: **any**
  - Linked issues or subtasks may belong to **any** project.
  - This feature will be run as user in field **Current user**.

## Other examples of that functions

- 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 Usage Examples

- [Validation on sibling sub-tasks depending on issue type and status](#)
  - [example](#)
  - [validator](#)
  - [sub-task](#)
  - [transition](#)
- [Block a transition until all sub-tasks have certain fields populated](#)
  - [example](#)
  - [condition](#)
  - [validator](#)
  - [sub-task](#)
  - [transition](#)
- [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](#)
- [Transition only a sub-task among several ones](#)
  - [example](#)
  - [post-function](#)
  - [sub-task](#)
  - [transition](#)
  - [outdated](#)
- [Transition sub-tasks when parent is transitioned](#)
  - [example](#)
  - [post-function](#)
  - [sub-task](#)
  - [transition](#)

- outdated
- Change parent's status depending on sub-task's summary
  - example
  - post-function
  - sub-task
  - transition
  - outdated
- Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress"
  - example
  - post-function
  - sub-task
  - transition
  - outdated
- Automatically close resolved sub-tasks when parent issue is closed
  - example
  - post-function
  - sub-task
  - transition
  - outdated
- Close parent issue when all sub-tasks are closed
  - example
  - condition
  - validator
  - post-function
  - sub-task
  - transition
- Proceed with a task only when all sub-tasks are completed
  - example
  - condition
  - validator
  - sub-task
  - transition
- Prevent transitioning when there is a blocking issue
  - example
  - validator
  - issue-links
  - sub-task
  - transition
- Transition parent issue only when certain issue sub-task types are done
  - example
  - validator
  - sub-task
  - transition
- Enforce certain type of sub-tasks to be "Resolved" to allow executing a transition
  - example
  - validator
  - sub-task
  - transition
- Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status
  - example
  - validator
  - post-function
  - sub-task
  - transition