

# Move issues

This function has been **renamed** with the **JWT 3.0** release.

Find the new documentation at:

[Move issue](#)

## On this page

- [Purpose](#)
- [Example: Moving current issue to project Accounting and converting it into a Task issue type](#)

## Purpose

SINCE VERSION 2.5.0

The **Move issues** post function allows you to move an issue to a different project, issue type or status inside a workflow transition. This post function must be the last post function inside a transition.

The following options are available:

1. **Project** (Select the target project.)
2. **Issue type** (Select the target issue type.)
3. **Status** (Select the target status.)
4. **Set fields** (Select additional fields to be set.)
5. **Conditional execution** (Optional boolean expression that should be satisfied in order to actually execute the post function.)
6. **Run as** (Select the user that will be used to execute the post function.)

**i** In case the configuration is not correct, e.g. the issue type is not available in the target project or the target status is not available for the new issue type, nothing will happen.

## Example: Moving current issue to project Accounting and converting it into a Task issue type

In this example we show how to configure the **Move issues** post function for moving the current issue to the project **ACCOUNTING**, and at the same time converting it from any issue type into a **Task**.

**Project:**  
Select the target project.

☒ Selected project ☐ Retain the project  

ACCOUNTING

**Issue type:**  
Select the target issue type.

☒ Selected issue type ☐ Retain the issue type  

Task

**Status:**  
Select the target status.

☒ Selected status ☐ Retain the status  

In Progress

**Set fields:**  
Select additional fields to be set. All the remaining field values will be inherited.

Field to be set:  

Start typing to get the list of available fields

Add

Add a field to be set in selected issues.

Target Field	Type of Value	Source Value	Don't Overwrite	Actions
Summary	Parsed text (basic mode)	%{Summary} (moved from project %{Project key})		<a>Edit</a> <a>Remove</a>
New comment	Parsed text (basic mode)	This issue was moved from project %{Project key} by %{Current user} on %{Current date and time}.		<a>Edit</a> <a>Remove</a>

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

1
%{00094} any in [ "Budget", "Payment", "Cash" ]

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:**  

Start typing to get the list of available fields

**Numeric/Date Field Code Injector:**  

Start typing to get the list of available fields

**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.

We are conditioning the execution of the post function to the presence of any of the following components in the issue: "Budget", "Payment" or "Cash". To execute the post function unconditionally, field **Conditional execution** should remain empty.

We are also adding string "(moved from project <original\_project\_key>)" to the summary of the moved issue, and automatically creating a comment like this: "The issue was moved from project <original\_project\_key> by <user\_who\_executes\_the\_post-function> on <date\_time\_of\_issue\_moving>."

Once configured, the post function will look like this:

The following will be processed after the transition occurs

[Add post function](#)

1. Set issue status to the linked status of the destination workflow step.
2. Add a comment to an issue if one is entered during a transition.
3. Update change history for an issue and store the issue in the database.
4. Re-index an issue to keep indexes in sync with the database.
5. Fire a **Generic Event** event that can be processed by the listeners.

6. Move **current issue** to



Project: **ACCOUNTING**  
 Issue type: **Task**  
 Status: **In Progress**

Target fields and Source values:

Target Field	Type of Value	Source Value	Don't Overwrite
Summary	Parsed text (basic mode)	%{Summary} (moved from project %{Project key})	
New comment	Parsed text (basic mode)	This issue was moved from project %{Project key} by %{Current user} on %{Current date and time}.	

Post-function will only be executed if the following boolean expression is satisfied: %{Components} any in ["Budget", "Payment", "Cash"]  
 This feature will be run as user in field **Current user**. by JWT

**i** Note that the post function has to be moved to the last position, otherwise nothing will happen.

**i** After the execution of the post function you might need to refresh your browser in order to view the the issue fully updated to the new project and issue type.