

# Update issue fields

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

Find the new documentation at:

[Update or copy field values](#)

## On this page

- [Purpose](#)
- [Example 1: Adding attachments uploaded in current issue's transition to other issues](#)
- [Example 2: Add 3 days to due date in issues blocked by current issue](#)
- [Example 3: Set priority in non-resolved tasks and subtasks to epic's priority if it's higher](#)
- [Configuration Parameters](#)
- [Usage Examples](#)
- [Related Features](#)

 **Renamed: Former "Write field on issues returned by JQL query or issue list"** SINCE VERSION 2.4.0

## Purpose

This post-function is used for **setting fields** in issues selected by **predefined options**, a **JQL Query** or an **Issue List** expression, using any of the target fields.

- **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 JQL selected issues (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 JQL selected issues (syntax `^{nnnnn}`). Here we can use all the functions available in the [Expression Parser](#).

## Example 1: Adding attachments uploaded in current issue's transition to other issues

Files attached to current issue in current transition's screen will be added to issues in status **"In Progress"** and assigned to the same user that has current issue assigned:

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

Target field:

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		<a href="#">Edit</a> <a href="#">Remove</a>

**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 49 ]

1 assignee = "%{00003}" AND status = "In Progress"

**String Field Code Injector:**  **Numeric/Date Field Code Injector:**

- Field codes with format %{nnnnn} 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 %{nnnnn.0} for parent level, %{nnnnn.1} for child level, etc.

**Additional options:**  
 Enable email notifications on issues to be written, according to applicable notification scheme.

**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:**

**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.

JQL query used is: `assignee = "%{00003}" AND status = "In Progress"`

Note that:

- `%{00003}` if field code for "Assignee"

Once configured, your post-function will look like this:

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

- Update the following issues:  
 Target issues: **JQL** query  
**assignee = "%{Assignee}" AND status = "In Progress"**  
 Target fields and Source values:
 

Target Field	Type of Value	Source Value	Don't Overwrite
Attachments (only new attachments will be added)	Field in current issue	Attachments	

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

## Example 2: Add 3 days to due date in issues blocked by current issue

**Due date** of issues linked to current issue with **"blocks"** issue link type, will be added 3 days, skipping weekend's days. In case the **Due date** is not set in blocked issues, it will be set using current issue's due date as value.

Described action will be carried out only into **blocked** issues in statuses **"Open"** or **"In Progress"**.

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

Target field:  Add

Add a field to be set in selected issues.

Target Field	Type of Value	Source Value	Don't Overwrite	Actions
Due date	Math/Time expression	<code>^{Due date} != null ? addDaysSkippingWeekends(^{Due date}, 3, LOCAL) : {Due date}</code>		<a href="#">Edit</a> <a href="#">Remove</a>

**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 80 ]

```
1 filterByStatus(linkedIssues("blocks"), "Open, In Progress")
```

**String Field Code Injector:**   
 **Numeric/Date Field Code Injector:** 
[Check Syntax](#)

Input an expression that returns an issue lists. (Syntax Specification)   [Collection of Examples](#)

**Additional options:**  **Enable email notifications** on issues to be written, according to applicable notification scheme.

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

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:** 
[Check Syntax](#)

Input an expression that returns an issue lists. (Syntax Specification)   [Collection of Examples](#)

**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.

Text to be parsed is:

```
^{00012} != null ? addDaysSkippingWeekends(^{00012}, 3, LOCAL) : {00012}
filterByStatus(linkedIssues("blocks"), "Open, In Progress")
```

Note that

- {00012} is field code for field "Due date"
- ^{00012} is field code for field "Due date" in blocked issues

Once configured, your post-function will look like this:

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

1. Update the following issues:  
Target issues: **Issue List** expression  
`filterByStatus(linkedIssues("blocks"), "Open, In Progress")`

Target fields and Source values:

Target Field	Type of Value	Source Value	Don't Overwrite
Due date	Math/Time expression	<code>^{Due date} != null ? addDaysSkippingWeekends(^{Due date}, 3, LOCAL) : {Due date}</code>	

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

### Example 3: Set priority in non-resolved tasks and subtasks to epic's priority if it's higher

This post-function is intended to be executed in Epic's workflow, i.e., current issue is an Epic.

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

Target field:

Add a field to be set in selected issues.

Target Field	Type of Value	Source Value	Don't Overwrite	Actions
Priority	Math/Time expression	min({Priority}, ^{Priority})		Edit 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 08]

1 filterByResolution(subtasks() UNION linkedIssues("is Epic of"), "")

**String Field Code Injector:**  **Numeric/Date Field Code Injector:**

Input an expression that returns an issue lists. (Syntax Specification) [Collection of Examples](#)

**Additional options:**  
 Enable email notifications on issues to be written, according to applicable notification scheme.

**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:** =, !=, >, >=, <, <= 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:**

**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.

Text to be parsed is:

```
min({00017}, ^{00017})
```

```
filterByResolution(subtasks() UNION linkedIssues("is Epic of"), "")
```

Note that:

- {00017} is field code for field "Priority"

Once configured, your post-function will look like this:

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

1. Update the following issues:  
 Target issues: **Issue List** expression  
`filterByResolution(subtasks() UNION linkedIssues("is Epic of"), "")`  
 Target fields and Source values:
 

Target Field	Type of Value	Source Value	Don't Overwrite
Priority	Math/Time expression	<code>min({Priority}, ^{Priority})</code>	

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

## Configuration Parameters

### Issue Selection Modes

There are several modes for selecting the issues whose field values are going to be set.

**New since Version 2.4.0:** **Current Issue, Parent Issue, Linked Epic, Linked Issues, Subtasks, Sibling Subtasks, Issues under Epic and Sibling issues under Epic.**

These new options simplify the usage of this post function a lot. For this cases no **JQL Query** or **Issue List** has to be configured.

### JQL Query

In this issue selection mode we use JQL, which is a language provided by Jira for doing [advanced issue searching](#).

You can insert field codes with format `%{nnnnn}` in your JQL query. These field codes will be replaced with the values of the corresponding fields in current issue at execution time, and the resulting JQL query will be processed by Jira JQL Parser. This way you can write dynamic JQL queries that depend on values of fields of current issue. Example: `issuetype = "{00014}" AND project = "{00018}"` will return issues in same project and with same issue type as current issue.

When you write your JQL for selecting the issues, take into account the following advices:

- If field values are expected to have **white spaces** or **JQL reserved words or characters**, you should write field code **between quotes** (double or simple). Example: `summary ~ "{00021}"` will return issues with current user's full name. As full name can contain spaces, we have written the field code between double quotes.
- In general we will write field codes between quotation marks, since in most cases it doesn't hurt and it's useful for coping with field values containing white spaces or reserved JQL words. Anyway, there is an exception to this general rule: when our field contains a **comma separated list of values**, and we want to use it with JQL operator **IN**. In those cases we will not write the field code between quotes, since we want the content of the field to be processed as a **list of values**, not as a single string value.

Example: Let's assume that "**Ephemeral string 1**" (field code `%{00061}`) contains a comma separate list of issue keys like "**CRM-1, HR-2, HR-3**". JQL Query `issuekey in (" "{00061} ")` will be rendered in runtime like `issuekey in ("CRM-1, HR-2, HR-3")`, which is syntactically incorrect. On the other hand, JQL Query `issuekey in (%{00061})` will be rendered in runtime like `issuekey in (CRM-1, HR-2, HR-3)`, which is correct.

### Disabling JQL Syntax Pre-Checking

When we enter our JQL query, a syntax pre-checking is carried out in order to verify that it's correctly written. But when we insert field codes in our JQL query, the definitive form of the query that will be executed is unknown, since it depends on the actual values of the fields in runtime. In these cases the syntax pre-checking is done with speculative values given to the fields, and it might happen that fake syntax errors are reported. In order to inhibit the JQL syntax pre-checking you should enter `//` at the beginning of the line. Those characters will be removed in the actual JQL query that will be executed.

Example:

```
JQL Query: 1 // issuekey = %{00061} [ Line 1 / Col 24 ]
```

## Issue List expression

In this issue selection mode we use an **issue list expression** according to the [syntax of the Expression Parser](#). Here you can find [Examples of Issue List expressions](#).

## 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.
- **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 particularly important in this feature since JQL query will return issues according to the browse permission this user has in the different project of the instance of Jira.

## Usage Examples

Page: [Add and remove a single or a set of items from multi valued fields](#)  
Page: [Compose dynamic text by inserting field values in a text template](#)  
Page: [Creating a Jira Service Desk internal comment](#)  
Page: [Creating a Jira Service Desk internal comment on linked issues](#)  
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: [Parse Email addresses to watchers list](#)  
Page: [Set priority for issues that have been in a certain status for longer than 24 hours](#)  
Page: [Transition linked issues in currently active sprint](#)  
Page: [Transition only a sub-task among several ones](#)  
Page: [Using project properties to calculate custom sequence numbers](#)  
Page: [Writing a comment to blocked issues when blocking issues are resolved](#)

## Related Features

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