Break issue link

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

Find the new documentation at:

Delete issue link

On this page

- Purpose
- Configuration Parameters
- Usage Examples
- Related Features

Purpose

Post-function "**Break issue link**" is aimed to make it possible for a workflow transition to selectively break issue links with current issue, or with any pair of issues in a Jira instance. Issue selection can be done by **JQL query** or **comma separated issue keys** in a field.

Configuration Parameters

In the following example we configure the post-function for breaking "**is blocked by**" issue links of **current issue** to **Improvement** or **New Feature** issues, in statuses **Open**, **In Progress** or **Reopened** with **Trivial** or **Minor** priorities, and **Due Date** greater (later) than current issue's Due Date:

Issue whose issue links are going to be broken:

This parameter sets issue(s) at issue link's **source end**.

This is the end of the issue link from which selected issue link types are named. e.g., source_end *is blocked by* destination_end.

- (●) Current issue
- ( ) Issues in field — Summary - [Text]

Selected field is expected to contain a comma or blank separated **list of issue keys**, e.g., *"CRM-1, CRM-2, CRM-3"* or *"CRM-1 CRM-2 CRM-3"*.

- ( ) Issues returned by JQL

1

[Line 1 / Col 1]

Field code injector:

Summary - [Text] - %{00000}

- 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.

Check Syntax

Issue link types to be broken:

Issue link types susceptible to being broken, provided the rest of filtering conditions are satisfied (issue status and project belonging).

- [x] is blocked by
- [ ] is cloned by
- [ ] is duplicated by
- [ ] is caused by
- [ ] relates to
- [ ] blocks
- [ ] clones
- [ ] duplicates
- [ ] causes
- [ ] relates to

If all issue link types are kept unchecked, there won't be applied any filter by issue link type, i.e., all issue link types will be considered as selected.

Issue types for the linked issues:

Issue types required for issues at the other end of the issue link. Only if issue at the other end of the link belongs to one of selected issue types, the issue link will be broken.

- [ ] Epic
- [ ] Story
- [ ] Bug
- [x] New Feature
- [ ] Task
- [x] Improvement
- [ ] QA Sub-task
- [ ] Sub-task

If all issue types are kept unckecked, there won't be applied any filter by issue type, i.e., all issue types will be considered as selected.

Statuses for the linked issues:
Statuses required for issues at the other end of the issue link. Only if issue at the other end of the link is in one of selected statuses, the issue link will be broken.

- [x] Open
- [x] In Progress
- [x] Reopened
- [ ] Resolved
- [ ] Closed
- [ ] To Do
- [ ] Done
- [ ] Acceptance
- [ ] Fail
- [ ] Pass
- [ ] Retest

If all statuses are kept unckecked, there won't be applied any filter by issue status, i.e., all statuses will be considered as selected.

Break only issue links within same project:
[ ]
When checked, issues at both sides of the link should belong to the same project, otherwise, no condition will be required in relation to project belonging.

Linked issues are in:
This is an **optional parameter** for setting which issues are at the **destination end** of the issue links which are going to be broken.

[x] Enable issue filtering at destination end

( ) Issues in field — Summary - [Text]

Selected field is expected to contain a comma or blank separated **list of issue keys**, e.g., *"CRM-1, CRM-2, CRM-3"* or *"CRM-1 CRM-2 CRM-3"*.

(•) Issues returned by JQL

```
priority in (Minor, Trivial) AND duedate < %{00012}
```

[Line 1 / Col 52]

Field code injector:
Due date - [Date] - %{00012}

- 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.

Check Syntax

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**.

Note that:

- **%{00012}** is field code for **Due Date**

Once configured, post-function looks like this:

Triggers 0 | Conditions 1 | Validators 1 | Post Functions 8

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

1. Break issue links of **current issue** fulfilling the following conditions:
   Inward issue link types: **is blocked by**.
   Issue types: **Improvement** and **New Feature**.
   Statuses: **Open**, **Reopened** and **In Progress**.
   Linked issues can belong to **any** project.
   Linked issues must be among issues returned by JQL query "`priority in (Minor, Trivial) AND duedate < %{Due date}`".
   This feature will be run as user in field **Current user**.

Issues returned by JQL

We use JQL queries for selecting issues. The syntax is the same used by Jira for **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.

Usage Examples

Page: Replace certain issue link types with different ones

Related Features

- **Create issue link**