Create issue link

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

Find the new documentation at:

Create issue link

On this page

Purpose

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

Example: Link issue across projects using reporter and user picker custom field

In the following example we configure a post-function to automatically create "**relates to**" issue links from **current issue** to issues in a project named **CONTACTS** with user picker custom field "**Contact User**" equal to current issue's **Reporter**. This is a typical usage for linking issues reported by a user, with its corresponding issue in a Jira project which is used as contacts directory:

Issues at source end:
This parameter sets issues at the **source end** of the issue links that will be created.

- (•) Current issue
  Current issue will be linked to destination issues.
- ( ) Issues in field
  Issues in field will be linked to destination issues.
  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
  Issues returned by JQL query will be linked to destination issues.

```
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 type:
This parameter sets the **issue link type** of the issue links that will be created.

- ( ) is blocked by
- ( ) blocks
- ( ) is cloned by
- ( ) clones
- ( ) is duplicated by
- ( ) duplicates
- (•) relates to
- ( ) relates to

Issue Link Direction: *source_end_issues* **issue link type** *destination_end_issues*
Example: *source_end_issue* **is blocked by** *destination_end_issue*

Issues at destination end:
This parameter sets issues at the **destination end** of the issue links that will be created.

- ( ) Issues in field
  Issues in field will be linked to source issues.
  Summary - [Text]
  Field is expected to contain a **comma** or **blank** separated **list of issues keys**, e.g., *"CRM-1, CRM-2, CRM-3"* or *"CRM-1 CRM-2 CRM-3"*.
- (•) Issues returned by JQL
  Issues returned by JQL query will be linked to source issues.

```
1 project = CONTACTS and "Contact User" = %{00006}
```

[Line 1 / Col 50]

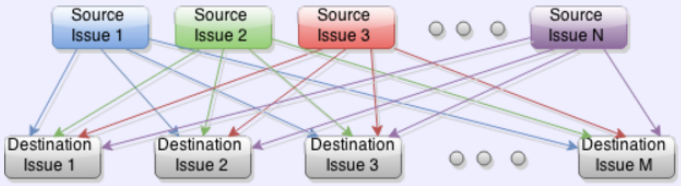
Field code injector:
Reporter - [User] - %{00006}

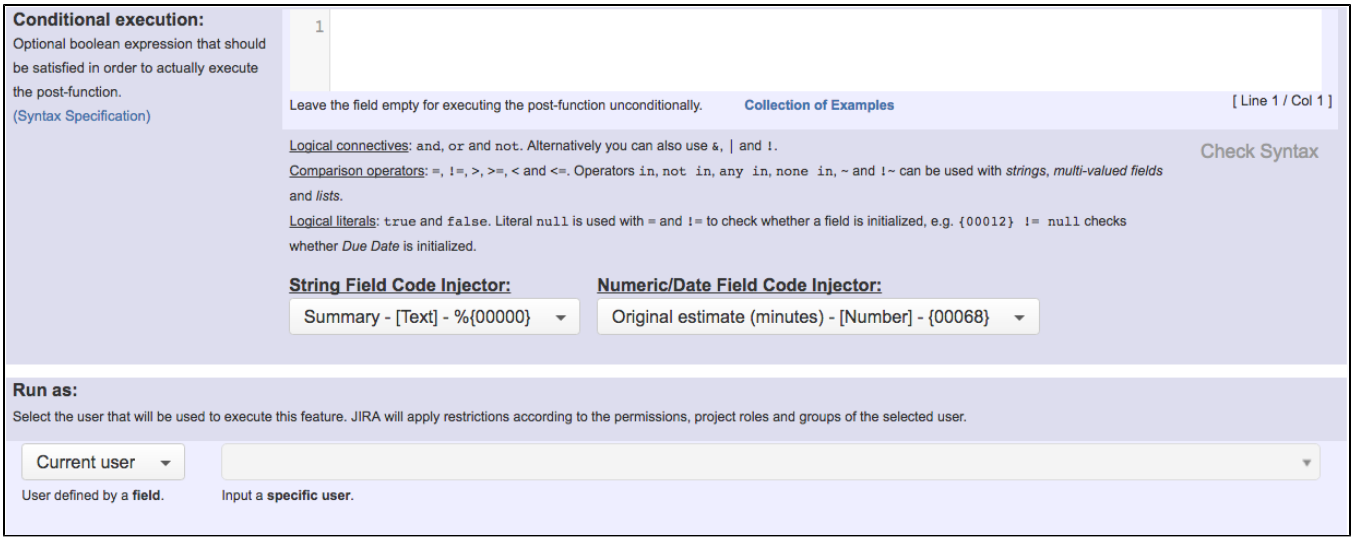
- Field codes with format `%{nnnnn}` may be inserted in the JQL Query, and will be replaced with field values at runtime. Most times it's a good idea to write field codes between double quotes (e.g. `"%{00001}"`), since field values may contain blank spaces that will produce JQL parsing errors at runtime.
- **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 linking operation:
This parameter sets which kind of issue linking operation will be carried out.

- (•) **Each source issue** will be linked to **all destination issues**.
- ( ) **Each source issue** will be linked to **one destination issue** according to order of appearance.

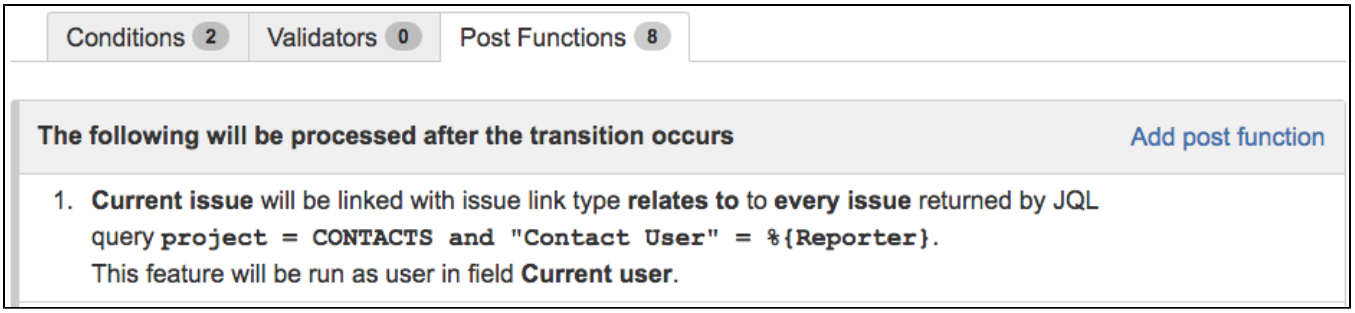




Note that:

- **%{00006}** is field code for **Reporter**

Once configured, post-function looks like this:



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: Automatically create an issue link after issue creation on email by "Enterprise Mail Handler for Jira" app
Page: Create issue links based on a custom field value avoiding duplicates
Page: Creating issue links to issues with the same "Summary"
Page: Parse description for creating issue links
Page: Replace certain issue link types with different ones

Related Features

- **Break issue link**