

Parse description for creating issue links

On this page

- [Features used to implement the example](#)
- [Example: Parse description for creating issue links](#)
- [Other examples of that functions](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Parse field for extracting data](#)
 - [Create issue link](#)
-

Example: Parse description for creating issue links

Let's see a configuration that allows creating issue links of type "**relates to**" simply when we insert sentences like "**relates to** " + comma separated **list of issue keys** (e.g. **relates to CRM-34, HR-12, HR-25**) in an issue's **Description**. We need to use two post-functions in "**Create Issue**" transition (after "**Creates de issue originally**" post-function):

We insert [Parse field for extracting data](#) post-function in "Create Issue" transition using the following configuration:

Source field: Field containing the text that will be parsed for extracting certain value(s).	Description - [Text]
Leading delimiter: Text literal or pattern expressed as a regular expression that should be matched immediately before the value to be extracted. By default, delimiters are not included in returned values.	relates{s+to{s+ ☐ text literal ☐ text literal (ignoring case) ☐ regular expression ☒ regular expression (ignoring case) Field code injector: Summary - [Text] - %{00000} Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding field values.
Format of the value to be extracted: Format of the expected value to be extracted. Some pattern types require a specification of the format to be entered.	☐ any text between leading and trailing marks (i.e., no pattern) ☐ number ☐ url ☐ email Number, url and email formats are filtered by regular expressions that match the vast majority of possible values, but there might be valid values that don't match these regular expressions. If avoiding any undesired value rejection is a priority, any text should be selected. ☒ regular expression ☐ regular expression (ignoring case) ☐ date-time (specifying date-time formats: Date and Time Patterns) Format types regular expression , regular expression (ignoring case) and date-time require a format specification to be entered. <u>Format specification:</u> ([W+-[d+{s*,?}[s*])+ Field code injector: Summary - [Text] - %{00000} Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding field values.
Trailing delimiter: Text literal or pattern expressed as a regular expression that should be matched immediately after the value to be extracted. By default, delimiters are not included in returned values.	 ☒ text literal ☐ text literal (ignoring case) ☐ regular expression ☐ regular expression (ignoring case) Field code injector: Summary - [Text] - %{00000} Dynamic patterns can be created through insertion of field codes that will be replaced with the corresponding field values.
Target field: Field that will be set with the extracted value(s).	Ephemeral string 1 - [Text] ☐ Don't overwrite target field if it's already set. ☐ Include the leading delimiter in the output. ☐ Include the trailing delimiter in the output.
Ocurrences to be extracted: Values to be extracted in case that more than one data matching extraction patterns are found.	☒ first occurrence in source field ☐ last occurrence in source field ☐ all occurrences (values are returned as a comma separated list of values)
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 <i>strings</i>, <i>multi-valued fields</i> and <i>lists</i>. Logical literals: true and false. Literal null is used with = and != to check whether a field is initialized, e.g. {00012} != null checks whether <i>Due Date</i> 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:

- Leading delimiter is regular expression: **relates\s+to\s+**
- Value format is regular expression for matching issue keys, optionally followed by a comma character: **(\w+-\d+\s*,?\s*)+**

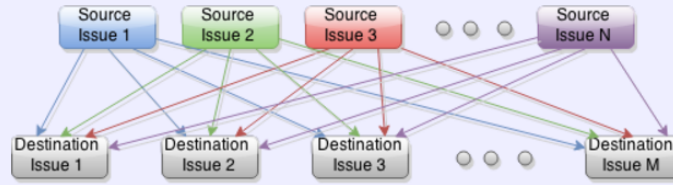
We insert **Create issue link** post-function in **"Create Issue"** transition using the following configuration:

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. ☐ Issues returned by JQL Issues returned by JQL query 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". 1 Field code injector: [Line 1 / Col 1] 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 ☐ is caused by ☐ causes ☒ relates to ☐ relates to <u>Issue Link Direction:</u> <i>source_end_issues</i> issue link type <i>destination_end_issues</i> Example: <i>source_end_issue</i> is blocked by <i>destination_end_issue</i>	
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. ☐ Issues returned by JQL Issues returned by JQL query will be linked to source issues.	Ephemeral string 1 - [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". 1 Field code injector: [Line 1 / Col 1] 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 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.



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.

Once configured, "Create Issue" transition look like this:



Create



TO DO

This is the **initial** transition in the workflow.

Screen: None - initial transition does not have a view.

Validators **1**

Post Functions **5**

The following will be processed after the transition occurs

[Add post function](#)

1. Creates the issue originally.
2. Write into **Ephemeral string 1** a piece of information extracted from **Description**.
Format of value to be extracted is a string matching the following **regular expression**:

```
(\w+--\d+\s*;\s*)+
```


Leading delimiter is the following **regular expression ignoring case**:

```
relates\s+to\s+
```


In case of multiple matches extract only **first** occurrence.
This feature will be run as user in field **Current user**.
3. **Current issue** will be linked with issue link type **relates to** to **every issue** in field **Ephemeral string 1**.
This feature will be run as user in field **Current user**.
4. Re-index an issue to keep indexes in sync with the database.
5. Fire a **Issue Created** event that can be processed by the listeners.

Example result:

Human Resources / HR-116

Example of Parsing Description for Issue Link Creation

Edit

Comment

Assign

More

Start Progress

Resolve Issue

Workflow

Admin

Export

Export

Component/s: None

Labels: None

Reporter: Administrator

Votes: 0

Watchers: 1 Stop watching this issue

Dates

Created: 4 minutes ago

Updated: 4 minutes ago

Agile

View on Board

HipChat discussions

Do you want to discuss this issue? Connect to HipChat.

Description

This is an issue example that relates to CRM-10, HR-100, HR-105

more text...

it also relates to CRM-14, CRM-15 and bla, bla, bla...

more text...

Bye, bye.

Issue Links

relates to

CRM-10 Improve functionality	↑	OPEN
CRM-14 Printing fails	↑	OPEN
CRM-15 Can't save document	↑	OPEN
HR-100 Fatal error at user creation	↑	OPEN
HR-105 Screen doesn't look as expected	↑	OPEN

Other examples of that functions

Parse field for extracting data

Page: [Parse description for creating issue links](#)

Page: [Parse summary for setting "Due date"](#)

Page: [Parse summary for setting issue priority](#)

Page: [Parsing text from last comment and appending it to issue's summary](#)

Create issue link

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 Usage Examples

- [Validate only issue links created in transition screen](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
- [Require issue link when resolving as duplicate](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
- [Ensure that all issues linked with a certain issue link type have "Due Date" field set](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
- [Block an epic's transition depending on linked issues status and due date](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
 - [transition](#)
- [Add and remove a single or a set of items from multi valued fields](#)
 - [example](#)
 - [post-function](#)
 - [custom-field](#)
 - [issue-links](#)
 - [sub-task](#)
- [Writing a comment to blocked issues when blocking issues are resolved](#)
 - [example](#)
 - [post-function](#)
 - [issue-links](#)
- [Prevent issue from moving forward if it's dependent on non-accepted tickets](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)

- transition
- Enforce linked issues in a specific project to be "Closed" before closing issue
 - example
 - validator
 - issue-links
 - transition
- Block or hide a transition for an issue depending on its issue links
 - example
 - validator
 - issue-links
 - transition
- Prevent transitioning when there is a blocking issue
 - example
 - validator
 - issue-links
 - sub-task
 - transition
- Prevent issue from being "Closed" if blocking issues aren't yet closed
 - example
 - validator
 - issue-links
 - transition
- Block creation of issue type X if it has not been linked with link type Y to issue type Z on the "Create Issue" screen
 - example
 - validator
 - issue-links
- Prevent issue from being closed if it has links of type "is blocked by" to open issues
 - example
 - condition
 - validator
 - issue-links
 - transition
- Transition linked issues in currently active sprint
 - example
 - post-function
 - issue-links
 - transition
- Automatically become watcher of every issue blocking an issue assigned to you
 - example
 - post-function
 - issue-links