

Set assignee depending on issue type

On this page

- [Features used to implement the example](#)
- [Example: Set assignee depending on issue type](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Set a field as a function of other fields](#)
- [Copy parsed text to a field](#)

Example: Set assignee depending on issue type

My immediate question is: How can I set assignee base on issue type? e.g. issue type Bug = assignee X. I looked at the post-function [Set a field as a function of other fields](#), but the "Target field to be set:" list does not contain Assignee. My not so immediate question is: How can I set assignee base on combinations of issue type and component? e.g. issue type Bug + Component A = assignee X, issue type Bug + Component B = assignee Y.

As you can see in the screenshot, "Assignee" is the third option in "Target field to be set" parameter of post-function [Set a field as a function of other fields](#).

The screenshot shows the 'Add Parameters To Function' dialog in Jira. The 'Field to be checked for matching with the set of setting rules:' dropdown is set to 'Summary'. The 'Setting rules:' section is empty. The 'Target field to be set:' dropdown is open, showing a list of available fields. 'Assignee' is the third option in the list. The 'Available fields:' section is also visible, showing a list of field codes and their corresponding field names.

Add Parameters To Function

Add required parameters to the Function.

Field to be checked for matching with the set of setting rules: Summary

Setting rules:
Put only one rule per line.
Rule format: `{(regular_expression)}value`
Regular expression syntax

Write only one rule per line. The rules will be processed in order. Once a rule is matched by the field under evaluation, its associated value will be parsed and copied to selected target field, and the rest of the rules won't be processed.
If selected target field is of type number, date or date and time, the associated value should be a number or a mathematical/time formula. Other types like user, date, issue status, issue priority and issue resolution require values of corresponding suitable types.

Rule format: `{(regular_expression)}value`
Both, regular expression and value will be parsed like in post-function "Copy parsed text to a field", this way, by inclusion of field codes, you will be able to create dynamic regular expressions and assignable values.

Target field to be set: Assignee

Available fields:

Copy and paste field codes (e.g. `%{00001}` for issue Description) to the text fields

FIELD CODE

- `%{00000}`
- `%{00001}`
- `%{00003}`
- `%{00002}`
- `%{00004}`
- `%{00006}`

FIELD TYPE

- Summary
- Description
- Assignee
- Reporter
- Due date
- Priority
- Original estimate (minutes)
- Remaining estimate (minutes)
- Total time spent (minutes)
- Components
- Fixed versions
- Affected versions
- Environment
- Issue status
- Issue resolution
- Labels

I answer to your second question: you can make setting of target field dependent on more than one field by **setting an auxiliary field (for example, Ephemeral string 1) with composition of more than one field**, and configure the parameter "Field to be checked for matching with the set of setting rules" with this auxiliary field.

We want to set "Assignee" in create transition with a user based on the value of "Issue type" and "Components":

We use post-function [Copy parsed text to a field](#) to set field "Ephemeral string 1" with a composition of fields "Issue type" and "Components", using character '@' as separator:

Update parameters of the Copy a parsed text to a field Function for this transition.

Update parameters of the Copy a parsed text to a field Function for this transition.

Target field:

Ephemeral string 1

Text to be parsed and then copied to target field:

%{00014}@%{00094}

- **Compose a free text** inserting field codes that will be replaced by corresponding field values prior to be copied to target field.
- You can reference parent and child values of **cascading select fields** writing %{xxxxx.0} for parent value, and %{xxxxx.1} for child value.
- You can change **issue reporter, assignee, due date, issue status, priority, fixed versions, affected versions, time estimated and time spent** choosing the suitable target field and value to be assigned.
- To assign cascading selects, multi-selects, multi-checkboxes, components, labels, fixed versions and affected versions you should use comma or semicolon separated values.
- Additionally, fields of type "Select list", "Radio button", "Multi select", "Multicheck boxes" and "Versions" can be set through regular expressions: options that matches a regular expression can be set by writing **/regular_expression/**, and options that doesn't match a regular expression can be set by writing **!/regular_expression/**.
- You can also use this post-function to **cast a string into a number**.

Available fields:

Copy and paste field codes (e.g. %{00001} for issue Description) to the text fields that will be parsed. When parsed all the field codes in the text will be replaced with its value.

FIELD CODE	FIELD NAME	FIELD TYPE
%{00000}	Summary	Text
%{00001}	Description	Text
%{00003}	Assignee	JIRA User

We use post-function **Set a field as a function of other fields** to write the setting rules we want. In this case we write 4 rules for issue types "Bug" and "New Feature" and components "Component A" and "Component B". We write a 5th rule to cover rest of case (as an "else" in an "if" statement).

Update parameters of the Set a field from a set of rules based on regular expressions Function for this transition.

Update parameters of the Set a field from a set of rules based on regular expressions Function for this transition.

Field to be checked for matching with the set of setting rules:

Ephemeral string 1

Setting rules:

Put only one rule per line.
Rule format: **/regular_expression/value**
Regular expression syntax

(Bug@Component A)feynman
(Bug@Component B)einstein
(New Feature@Component A)kaku
(New Feature@Component B)newton
(.*)admin

Write **only one rule per line**. The rules will be processed in order. Once a rule is matched by the field under evaluation, its associated value will be parsed and copied to selected target field, and the rest of the rules won't be processed.
If selected target field is of type **number, date or date and time**, the associated value should be a **number** or a **mathematical/time formula**. Other types like **user, date, issue status, issue priority and issue resolution** require values of corresponding suitable types.

Rule format: **/regular_expression/value**

Both, **regular expression** and **value** will be **parsed** like in post-function "Copy parsed text to a field", this way, by inclusion of field codes, you will be able to create **dynamic regular expressions and assignable values**.

Target field to be set:

Assignee

Available fields:

Copy and paste field codes (e.g. %{00001} for issue Description) to the text fields that will be parsed. When parsed all the field codes in the text will be replaced with its value.

FIELD CODE	FIELD NAME	FIELD TYPE
%{00000}	Summary	Text
%{00001}	Description	Text
%{00003}	Assignee	JIRA User

As in **Components** field you can set more than one component at the same time, it is convenient to write rules to specify precedence of components. For example, if you want "Component A" to have precedence over "Component B", you should write this set of rules:

```
(Bug@.*Component A.*)feynman  
  
(Bug@Component B)einstein  
(New Feature@.*Component A.*)kaku  
(New Feature@Component B)newton  
(.*)admin
```

Once configured, transition looks like this:

All Validators (1) Post Functions (4)

Add a new post function to the unconditional result of the transition.

Creates the issue originally.

THEN

The following parsed text will be copied to **Ephemeral string 1**:

#Issue type#@##Components#

Edit | Move Up | Move Down | Delete

THEN

The field **Assignee** will be set according to the evaluation of **Ephemeral string 1** against the following set of rules:

(Bug@."Component A.")feynman

(Bug@Component B)einstein

(New Feature@."Component A.")kaku

(New Feature@Component B)newton

(.*)admin

Edit | Move Up | Move Down | Delete

THEN

Fire a **Issue Created** event that can be processed by the listeners.

Edit

Other examples of that function

Set a field as a function of other fields

[Page: Add watcher depending on security level](#)
[Page: Add watchers based on issue type](#)
[Page: Add watchers depending on the value of a custom field](#)
[Page: Assign issue based on the value of a Cascading Select custom field](#)
[Page: Assign issue to a specific user based on a specific custom field value](#)
[Page: Assign issue to current user if assignee is empty](#)
[Page: Assign issue to current user if the user is not member of a certain project role](#)
[Page: Change assignee based on a custom field](#)
[Page: Change parent's status depending on sub-task's summary](#)
[Page: Changing issue priority depending on issue description](#)
[Page: Compose dynamic text by inserting field values in a text template](#)
[Page: Copy "Due date" into a date type custom field in a linked issue if it's greater than current issue's "Due date"](#)
[Page: Limit the number of hours a user can log per day](#)
[Page: Make parent issue progress through its workflow](#)
[Page: Rise priority if due date is less than 3 weeks away](#)
[Page: Set "Due date" depending on the value of other fields, in case it's uninitialized](#)
[Page: Set "Due date" to a specific day of next week no matter of date of creation this week](#)
[Page: Set "Due date" to current date at issue creation if not initialized](#)
[Page: Set a custom field "Urgency" depending on a combined value of issue's priority and "Impact" custom field](#)
[Page: Set a date based on current date](#)
[Page: Set a field based on reporter's email](#)
[Page: Set a watcher at ticket creation depending on custom field's value](#)
[Page: Set assignee depending on issue type](#)
[Page: Set security level based on groups and project roles the reporter or creator are in](#)
[Page: Set security level depending on reporter or creator](#)
[Page: Set the assignee based on a condition](#)
[Page: Set the value of a field of type "User Picker" depending on other field's value](#)
[Page: Set watchers depending on the value of a custom field](#)
[Page: Setting a custom field \(User Picker\) based on the value of another custom field \(Text Field\)](#)
[Page: Setting a field's default value depending on another field](#)
[Page: Setting the priority depending on the multiplication of custom fields](#)
[Page: Transition an issue automatically depending on the value of a field](#)
[Page: Unassign an issue when assigned to project leader](#)
[Page: Update checkboxes custom field if a file has been attached during a transition](#)

Related Usage Examples

- [Creating a Jira Service Desk internal comment](#)
 - [example](#)
 - [post-function](#)
- [Limit the number of hours a user can log per day](#)
 - [example](#)
 - [validator](#)
 - [post-function](#)
 - [work-log](#)
- [Using project properties to calculate custom sequence numbers](#)
 - [example](#)
 - [post-function](#)
 - [calculated-field](#)
 - [project-properties](#)
- [Set a date based on current date](#)
 - [example](#)
 - [post-function](#)
- [Setting the priority depending on the multiplication of custom fields](#)
 - [example](#)
 - [calculated-field](#)
 - [post-function](#)
- [Parse Email addresses to watchers list](#)
 - [example](#)
 - [post-function](#)
- [Set the assignee based on a condition](#)
 - [example](#)
 - [post-function](#)
- [Create a dynamic set of sub-tasks based on checkbox selection with unique summaries](#)
 - [example](#)
 - [post-function](#)
 - [custom-field](#)
 - [sub-task](#)
- [Create a static set of sub-tasks with unique summaries](#)
 - [example](#)
 - [post-function](#)
- [Triage Jira Service Desk email requests \(Move issues\)](#)
 - [example](#)
 - [post-function](#)
 - [move](#)
 - [transition-issue](#)
- [Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress" \(Transition issues\)](#)
 - [example](#)
 - [post-function](#)
 - [transition](#)
- [Transition sub-tasks when parent is transitioned](#)

Copy parsed text to a field

Page: Add all assignees of certain sub-task types to a "Multi-User Picker" custom field

Page: Add and remove a single or a set of items from multi valued fields

Page: Add current user to comment

Page: Add or remove request participants

Page: Add watchers from a part of the issue summary: "Summary_text - watcher1, watcher2, watcher3, ..."

Page: Assign issue based on the value of a Cascading Select custom field

Page: Assign issue to last user who executed a certain transition in the workflow

Page: Automatically close resolved sub-tasks when parent issue is closed

Page: Automatically reopen parent issue when one of its sub-tasks is reopened

Page: Calculate the time elapsed between 2 transition executions

Page: Close parent issue when all sub-tasks are closed

Page: Combine the values of several Multi-User picker fields

Page: Compose a parsed text including the "full name" or a user selected in a User Picker custom field

Page: Compose dynamic text by inserting field values in a text template

Page: Copy issue labels to a custom field

Page: Copy the value of a user property into a user picker

Page: Create a comment in sub-tasks when parent transitions

Page: Execute transition in epic

Page: Getting the number of selected values in a custom field of type Multi Select

Page: Limit the number of hours a user can log per day

Page: Make a sub-task's status match parent issue's current status on creation

Page: Make parent issue progress through its workflow

Page: Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress"

Page: Moving story to "Ready for QA" once all its sub-tasks are in "Ready for QA" status

Page: Parse Email addresses to watchers list

Page: Parsing text from last comment and appending it to issue's summary

Page: Remove versions selected in a version picker custom field

Page: Replace certain issue link types with different ones

Page: Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status

Page: Set a Select or Multi-Select field using regular expression to express the values to be assigned

Page: Set assignee depending on issue type

Page: Set field depending on time passed since issue creation

Page: Set priority for issues that have been in a certain status for longer than 24 hours

Page: Set security level based on groups and project roles the reporter or creator are in

Page: Transition linked issues in currently active sprint

Page: Transition only a sub-task among several ones

Page: Transition parent issue only when certain issue sub-task types are done

Page: Update Cascading Select custom field with a value of the field in parent issue

Page: Update checkboxes custom field if a file has been attached during a transition

Page: Validation on issue attachments

Page: Validation on MIME types of issue attachments

Page: Writing a comment to blocked issues when blocking issues are resolved

- example
 - post-function
 - sub-task
 - transition
 - outdated
- Transition only a sub-task among several ones
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Moving sub-tasks to "Open" status when parent issue moves to "In Progress"
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Moving story to "Ready for QA" once all its sub-tasks are in "Ready for QA" status
 - example
 - post-function
 - sub-task
 - transition
 - outdated