Assign to project role

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

Find the new documentation at:

Assign to project role

On this page

Purpose

Assigns an issue to a user in a project role. In case there are more than one user with the project role, you can set a user as default for a project role in a project. You can use this functionality with every post-function in the plugin that allows you to write on any field of type User or Multi User.

There are other implementations available for doing the same, but they usually require a user property for each project and role, and in case you use a big number of projects and project roles, the configuration may require a lot of effort to configure and usually is prone to errors.

Usable Features

You can use any of the following post-functions:

- **Assign to Project Role**, with the following 12 assignment modes:
  - **default user** for project role.
  - **default user** for project role, **except if current assignee is already in project role.**
  - **last user** in project role who has had the issue assigned.
  - **last user** in project role who has had the issue assigned, or lacking that to **default user** for project role.
  - **previous user** in project role who has had the issue assigned. (available since version 2.2.35)
  - **previous user** in project role who has had the issue assigned, or lacking that to **default user** for project role. (available since version 2.2.35)
  - **random user** among those in project role, **except if current assignee is already in project role.**
  - **random user** among those in project role **different from current assignee.**
  - **least busy** user in project role, i.e., user with **fewer non-closed assigned issues** (non-closed issues = empty resolution).
  - **least busy** user in project role, **except if current assignee is already in project role.**
  - **next user** in selected group and project role according to **round-robin** algorithm. (available since version 2.2.33)
  - **next user** in selected group and project role according to **round-robin** algorithm, **except if current assignee is already in project role.** (available since version 2.2.33)

- **Copy parsed text to a field**: useful to assign **parent issue** by project role, or to set custom fields of types **user picker** and **multi-user picker** by project role.
- **Set a field as a function of other fields**: to assign issues to different project roles depending on the value of other virtual or custom fields values.
- **Write field on linked issues or sub-tasks**: to assign by project role sub-tasks, sibling sub-tasks, linked issues or transitively linked issues.
- **Update issue fields**: to assign by project role any issue returned by a JQL query or issue list expression.

The most easy and direct way to assign an issue by project role is using post-function **Assign to Project Role**, but you can also simply copy the **name of a project role** into virtual field **Assignee**. This way you can use post-functions **Write field on linked issues or sub-tasks**, **Update issue fields** an d **Set a field as a function of other fields**, to assign any issue, or to make conditional assignments.

Actually by copying the name of a project role you can set **any field of type User** and **Multi User** besides **Assignee**.

When you have more than one user playing the same project role in a project, you can set a user as default for a project role within a project.

Example 1: Setting default user for a project role within a project

There are two ways to set the default user for a project role within a project: by **project property** and by **user property**.

- **Project Properties**: adding in project's **Description** a pair key-value called **Project Properties** (a string with format **{property_name=property_value}**) that will ne used to set the default user for a project role:
    - **{project_role_name=user_name}**
      Example: **`{Developers=albert.einstein}`**

    - **{projectRolexxxxx=user_name}**
      Example: **`{projectRole10100=richard.feynman}`**, where **10100** is the ID of the **project role**.

- **User property**: adding a property to the user you want to be the default for a project role.
    - **key** = project_role_name
      value = regular_expression, to be matched by the **project key** or by the project **category name**.
      Example: **`key=Developers and value=CRM|TRB|JWKT`**, sets the user with the property as default user for project role Developers in projects with keys CRM, TRB and JWKT.

    - **key** = **projectRole**xxxxx, where **xxxxx** is the ID of the **project role**.
      value = regular_expression, to be matched by the **project key** or by the project **category name**.
      Example: **`key=projectRole10100 and value=JAVA Projects`**, sets the user with the property as default user for project role with ID **10100** in projects with category **JAVA Projects**.

      Example: **`key=projectRole10200 and value=[ABCD]...|..CR`**, sets the user with the property as default user for project role with ID **10100** in projects **with project keys of 4 characters with A, B, C OR D as first character, or ending by CR**.

Independent from this property the user must be in the project role of the selected project, otherwise the function assign to project role will not work.

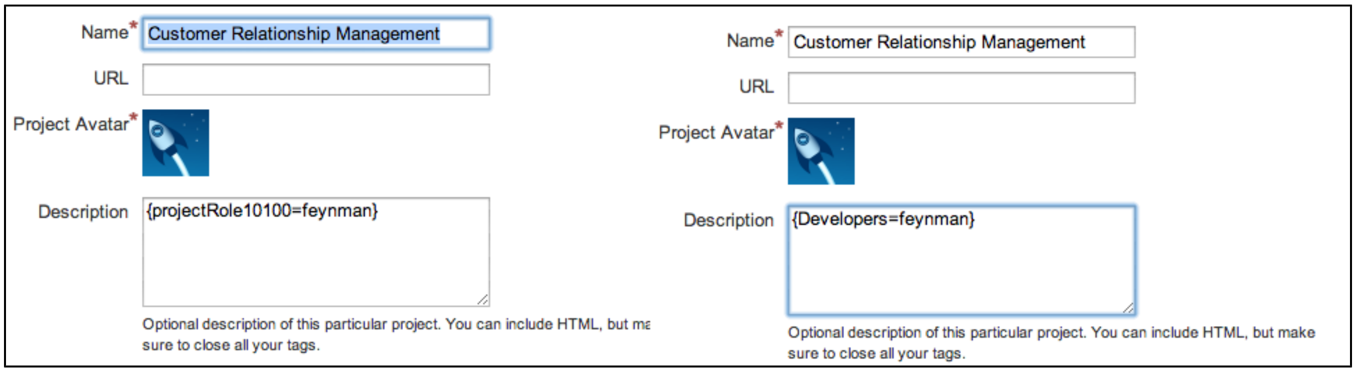
Using **projectRole**xxxxx instead of **project_role_name** as property name has the advantage that you can rename the project role without having to update the project property.

In case you set project role's default user using both, **project properties** and **user property**, then default user set by **project properties** will be used instead of the one set by **user property**.

We want to set user richard.feynman as the default user for project role Developers (being **10100** the **project role** ID) in a project with key CRM.

Using project property to set default user for a role

We can add in project description the property **{Developers=richard.feynman}**, or alternatively we ca use the property **{projectProperty10100=richard.feynman}**.

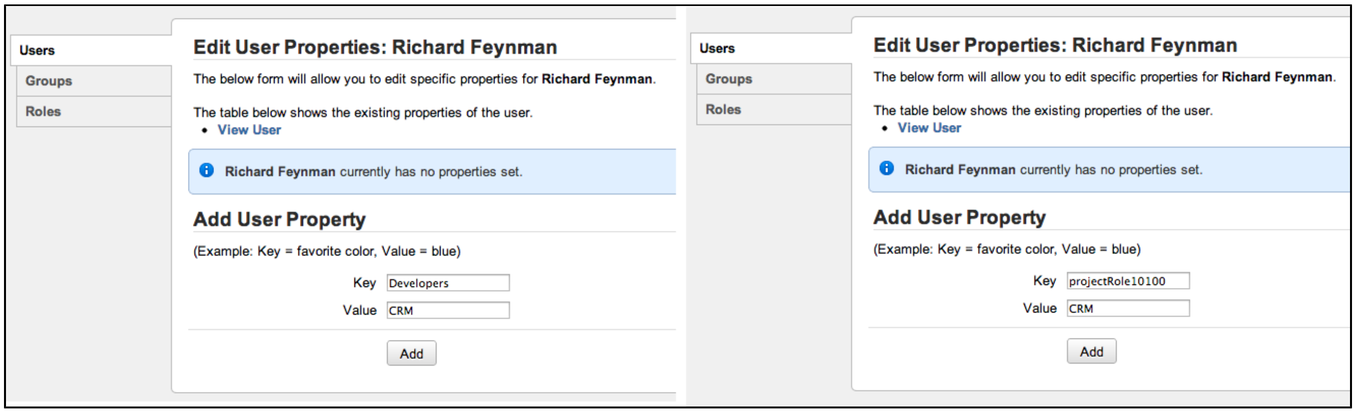


Name*	Customer Relationship Management
URL	
Project Avatar*	
Description	{projectRole10100=feynman}
	Optional description of this particular project. You can include HTML, but ma sure to close all your tags.

Name*	Customer Relationship Management
URL	
Project Avatar*	
Description	{Developers=feynman}
	Optional description of this particular project. You can include HTML, but make sure to close all your tags.

Using user property to set default user for a role

We add a user property to the user we want to be the default user for the project role:



Edit User Properties: Richard Feynman

The below form will allow you to edit specific properties for **Richard Feynman**.

The table below shows the existing properties of the user.

- View User

Richard Feynman currently has no properties set.

Add User Property

(Example: Key = favorite color, Value = blue)

Key	Developers
Value	CRM

Edit User Properties: Richard Feynman

The below form will allow you to edit specific properties for **Richard Feynman**.

The table below shows the existing properties of the user.

- View User

Richard Feynman currently has no properties set.

Add User Property

(Example: Key = favorite color, Value = blue)

Key	projectRole10100
Value	CRM

Since Version 2.2.33

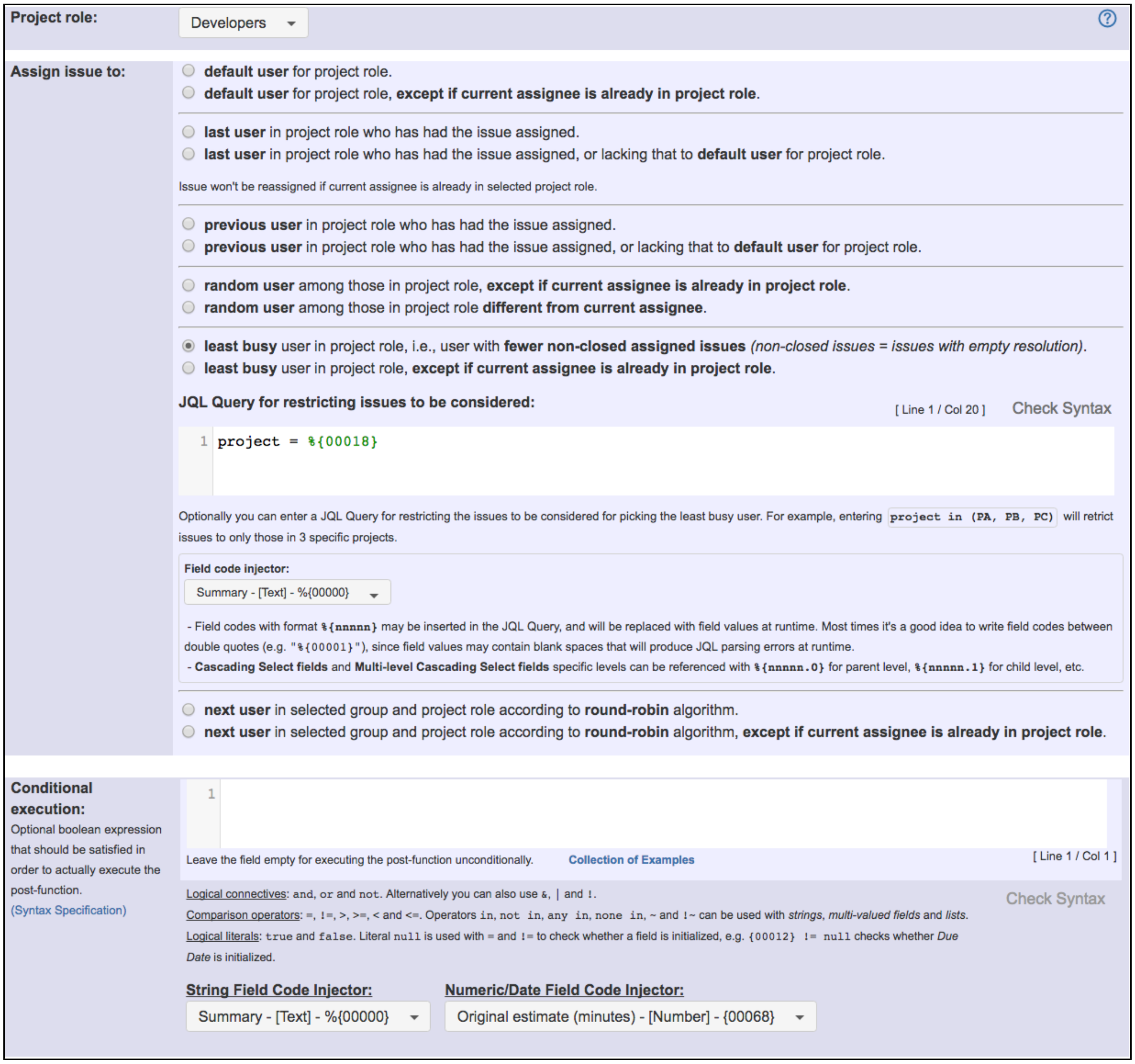
Post-function **Assign to Project Role** has 2 options for assigning issues to the least loaded user in a project role:

- **least busy** user in project role, i.e., user with **fewer non-closed assigned issues**
- **least busy** user in project role, **except if current assignee is already in project role.**

An issue is considered non-closed if it has unset **Resolution**.

Example 2: Load balancing: Assign to the least busy user in a project role

Since version **2.2.33**, it's possible to insert a JQL Query for restricting the issues to be considered when calculating the least loaded user. This way you can assign the issue to the user with fewer non-closed issues in the project, like in the example:



Project role: Developers

Assign issue to:

- ○ **default user** for project role.
- ○ **default user** for project role, **except if current assignee is already in project role**.

- ○ **last user** in project role who has had the issue assigned.
- ○ **last user** in project role who has had the issue assigned, or lacking that to **default user** for project role.

Issue won't be reassigned if current assignee is already in selected project role.

- ○ **previous user** in project role who has had the issue assigned.
- ○ **previous user** in project role who has had the issue assigned, or lacking that to **default user** for project role.

- ○ **random user** among those in project role, **except if current assignee is already in project role**.
- ○ **random user** among those in project role **different from current assignee**.

- ● **least busy** user in project role, i.e., user with **fewer non-closed assigned issues** *(non-closed issues = issues with empty resolution)*.
- ○ **least busy** user in project role, **except if current assignee is already in project role**.

JQL Query for restricting issues to be considered: [Line 1 / Col 20] Check Syntax

```
project = %{00018}
```

Optionally you can enter a JQL Query for restricting the issues to be considered for picking the least busy user. For example, entering `project in (PA, PB, PC)` will retrict issues to only those in 3 specific projects.

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.

- ○ **next user** in selected group and project role according to **round-robin** algorithm.
- ○ **next user** in selected group and project role according to **round-robin** algorithm, **except if current assignee is already in project role**.

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]

Check Syntax

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: Summary - [Text] - %{00000}

Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

There are also 3 parser functions in order to select the least loaded user in a project role. This functions can be used for assigning issues in sub-tasks, linked issues, JQL selected issues or newly created issues using post-function **Create issues and sub-tasks**:

FUNCTION	RETURNED VALUE
leastBusyUserInRole(string **projectRoleName**) : string Available since version 2.2.8	Returns the name of the active user playing project role with name **projectRoleName** in current issue's project, and has the lower number of issues with resolution empty assigned; or **null** if there isn't any user in the project role. Parameter **projectRoleName** can be a comma separated list of project role names, returning the least busy users among the project roles. Example: **`leastBusyUserInRole("Developers")`** returns the user playing role Developers in current project with the least number of unresolved issues in all the JIRA instance assigned.
leastBusyUserInRole(string **projectRoleName**, string **projectKey**) : string Available since version 2.2.8	Equivalent to the previous function but with extra argument **projectKey** for selecting the project argument **projectRoleName** refers to. Example: **`leastBusyUserInRole("Developers", "CRM")`** returns the user playing role Developers in project with key CRM with the least number of unresolved issues in all the JIRA instance assigned.

leastBusyUserInRole(string **projectRoleName**, string **projectKey**, string **jqlQuery**) : string Available since version 2.2.33	Equivalent to the previous function but with extra argument **jqlQuery**, used for restricting the issues to be considered to pick the least busy user. Example: **`leastBusyUserInRole("Developers", %{00018}, "project = " + %{00018})`** returns the user playing role Developers in current project, with the least number of unresolved issues in current project assigned. Note that **`%{00018}`** is field code for **Project key**.

Since version 2.2.33

Since version **2.2.33**, post-function **Assign to Project Role** has 2 options for assigning issues by turns using round-robin algorithm:

- **next user** in selected group and project role according to **round-robin** algorithm.
- **next user** in selected group and project role according to **round-robin** algorithm, **except if current assignee is already in project role.**

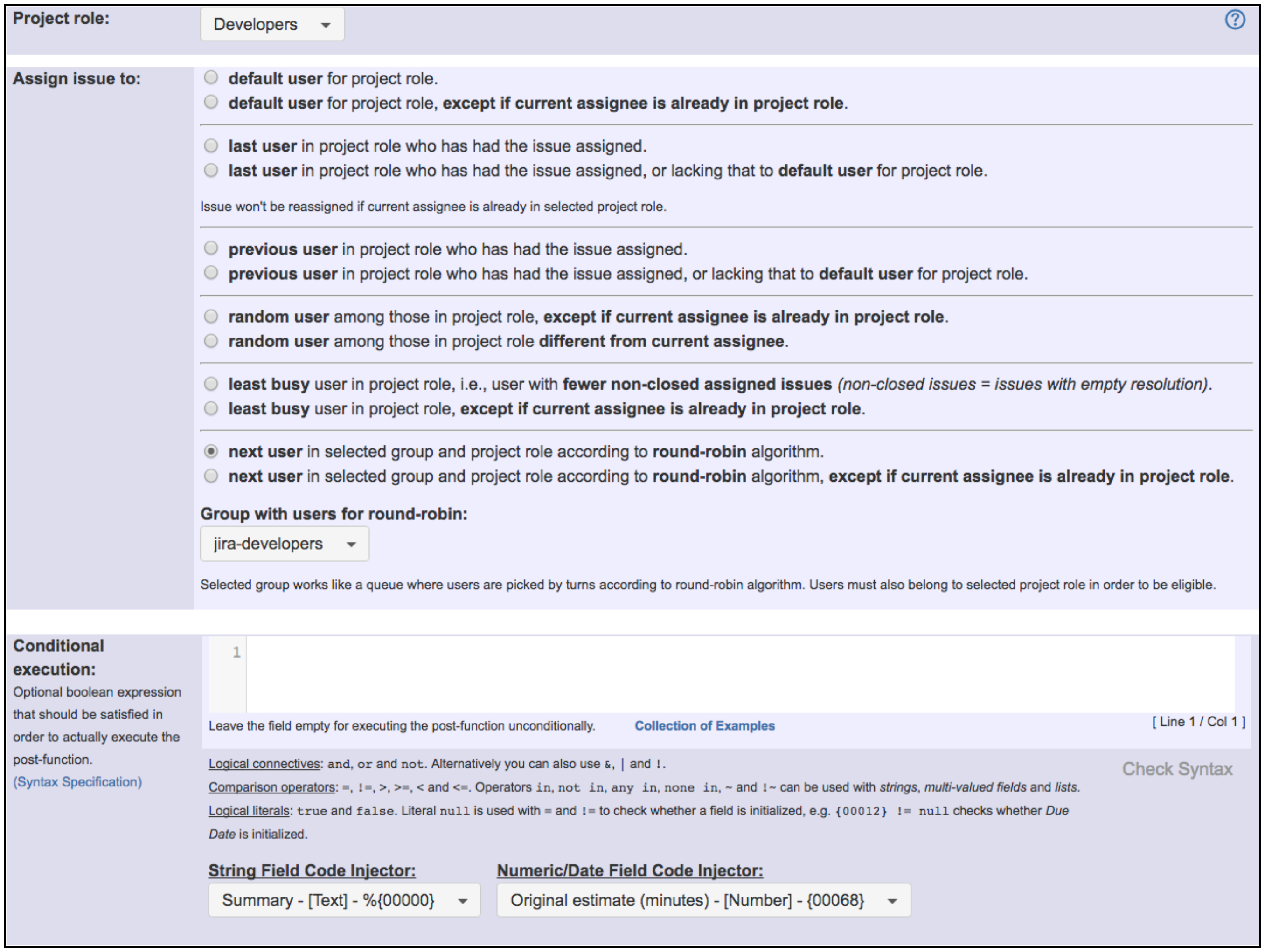
Round-Robin Queue

This kind of assignment requires to select a **group**, which in combination with the selected **project role**, define a **round-robin queue**. Each time post-function **Assign to Project Role** is executed in any workflow with the same configuration (i.e., same group and project role), the issue will be assigned to the next user in the round-robin queue.

The **round-robin queue** consists of all the users in the selected group an project role at the same time.

Example 3: Assign to users in project role by round-robin

Assigning the issue to users in project role Developers and group jira-developers by round-robin:



Project role: Developers

Assign issue to:

- ○ **default user** for project role.
- ○ **default user** for project role, **except if current assignee is already in project role**.
- ○ **last user** in project role who has had the issue assigned.
- ○ **last user** in project role who has had the issue assigned, or lacking that to **default user** for project role.

Issue won't be reassigned if current assignee is already in selected project role.

- ○ **previous user** in project role who has had the issue assigned.
- ○ **previous user** in project role who has had the issue assigned, or lacking that to **default user** for project role.
- ○ **random user** among those in project role, **except if current assignee is already in project role**.
- ○ **random user** among those in project role **different from current assignee**.
- ○ **least busy** user in project role, i.e., user with **fewer non-closed assigned issues** *(non-closed issues = issues with empty resolution)*.
- ○ **least busy** user in project role, **except if current assignee is already in project role**.
- [✓ **next user** in selected group and project role according to **round-robin** algorithm.]
- ○ **next user** in selected group and project role according to **round-robin** algorithm, **except if current assignee is already in project role**.

Group with users for round-robin: jira-developers

Selected group works like a queue where users are picked by turns according to round-robin algorithm. Users must also belong to selected project role in order to be eligible.

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.

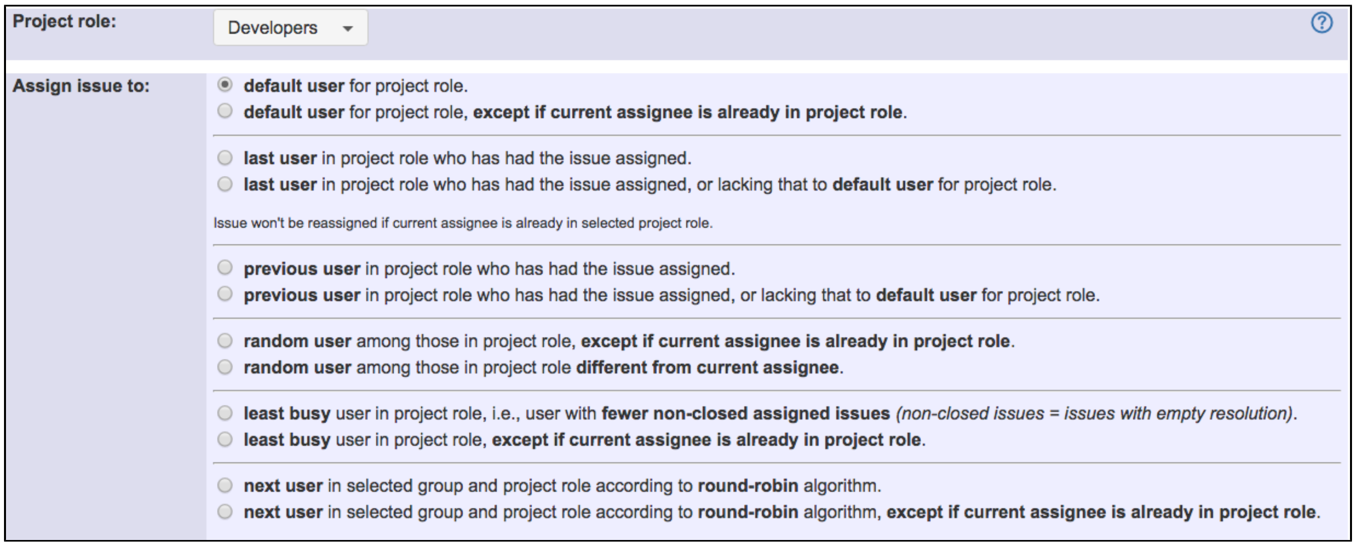
String Field Code Injector: Summary - [Text] - %{00000}

Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

There are also 1 parser function to select users by round-robin. This function can be used for assigning issues in sub-tasks, linked issues, JQL selected issues or newly created issues using post-function **Create issues and sub-tasks**

FUNCTION	RETURNED VALUE
nextUserInGroup(string **groupName**, string **queueName**) : string Available since version 2.2.33	returns the name of the next active user in group with name **groupName**, for a round-robin queue with name **queueName**. The string **queueName** is an arbitrary name. The queue is automatically created the first time a queue is used in a function call. Each time the function is called on the same pair of arguments **(group, queue)**, a different user in the group is returned. The queue can be used in different transitions of the same or different workflows within the same JIRA instance. Example: **`nextUserInGroup("jira-developers", "code-review-queue")`** returns the username of the next user in group **jira-developers** for round-robin queue **code-review-queue**. Each time the function is called with the same pair of arguments, a different username is returned.

Example 4: Use post-function assign to project role to assign current issue to default user set for project role "Developers"



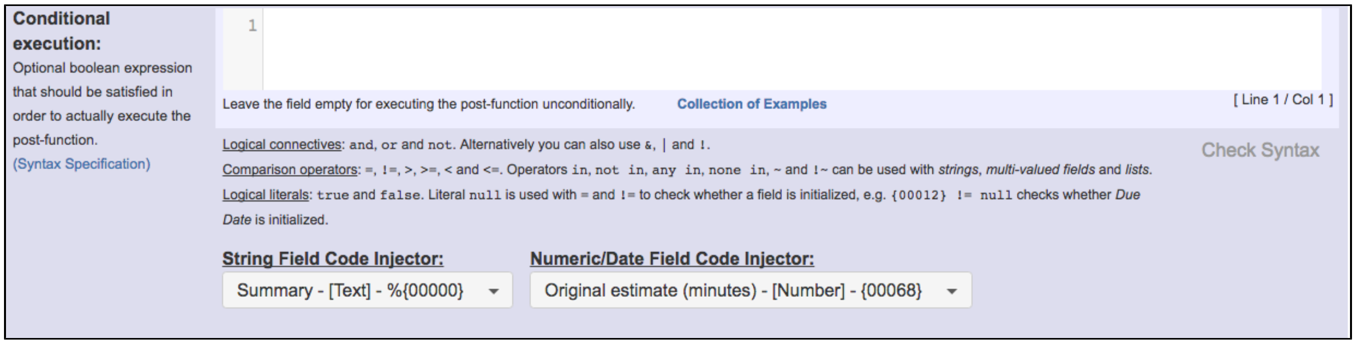
Project role: Developers

Assign issue to:

- [●] **default user** for project role.
- ○ **default user** for project role, **except if current assignee is already in project role**.
- ○ **last user** in project role who has had the issue assigned.
- ○ **last user** in project role who has had the issue assigned, or lacking that to **default user** for project role.

Issue won't be reassigned if current assignee is already in selected project role.

- ○ **previous user** in project role who has had the issue assigned.
- ○ **previous user** in project role who has had the issue assigned, or lacking that to **default user** for project role.
- ○ **random user** among those in project role, **except if current assignee is already in project role**.
- ○ **random user** among those in project role **different from current assignee**.
- ○ **least busy** user in project role, i.e., user with **fewer non-closed assigned issues** *(non-closed issues = issues with empty resolution)*.
- ○ **least busy** user in project role, **except if current assignee is already in project role**.
- ○ **next user** in selected group and project role according to **round-robin** algorithm.
- ○ **next user** in selected group and project role according to **round-robin** algorithm, **except if current assignee is already in project role**.



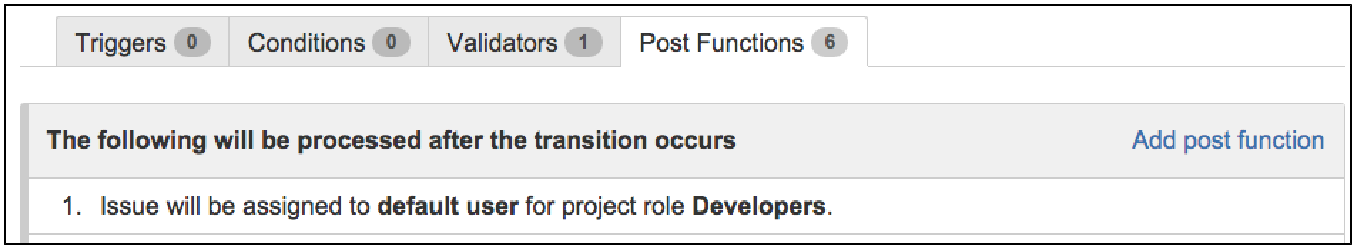
Conditional execution:
Optional boolean expression that should be satisfied in order to actually execute the post-function.
(Syntax Specification)

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: Summary - [Text] - %{00000}
Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068}

Once configured transition's post-function tab looks like this:

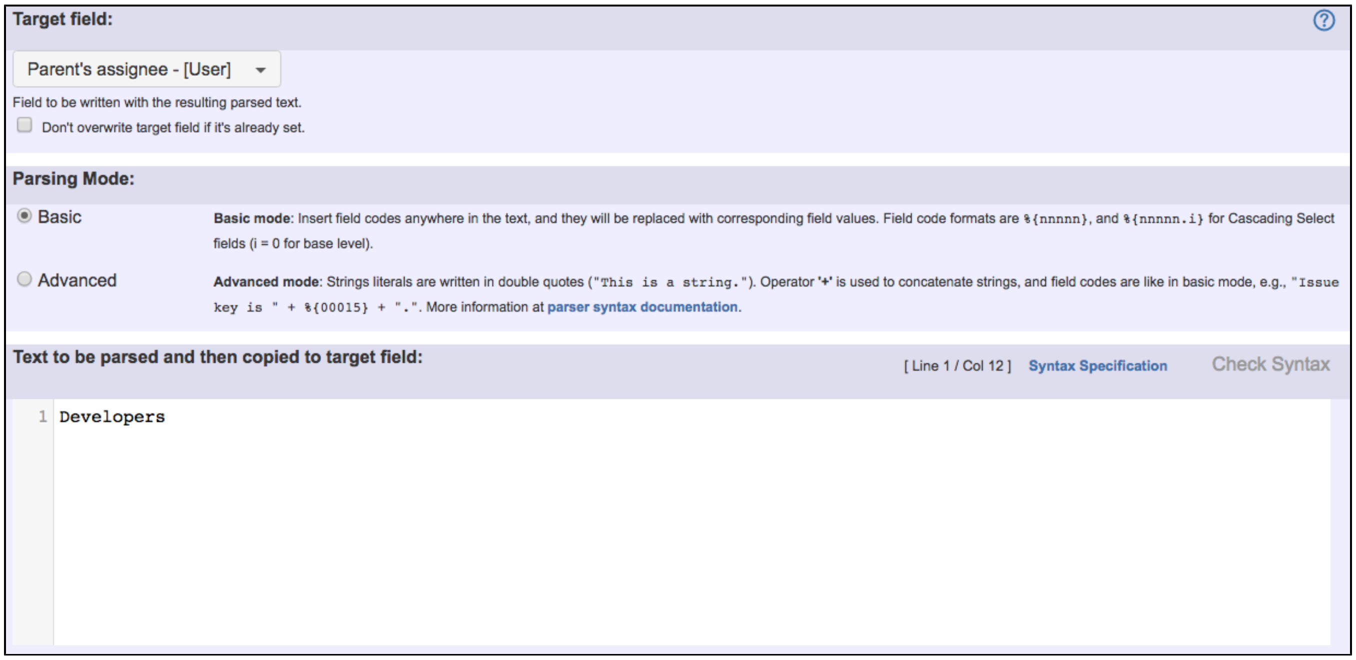


The following will be processed after the transition occurs

1. Issue will be assigned to **default user** for project role **Developers**.

Other examples

Example 5: Use post-function **Copy parsed text to a field** to assign parent issue to project role "Developers"



Target field:

Parent's assignee - [User]

Field to be written with the resulting parsed text.

☐ Don't overwrite target field if it's already set.

Parsing Mode:

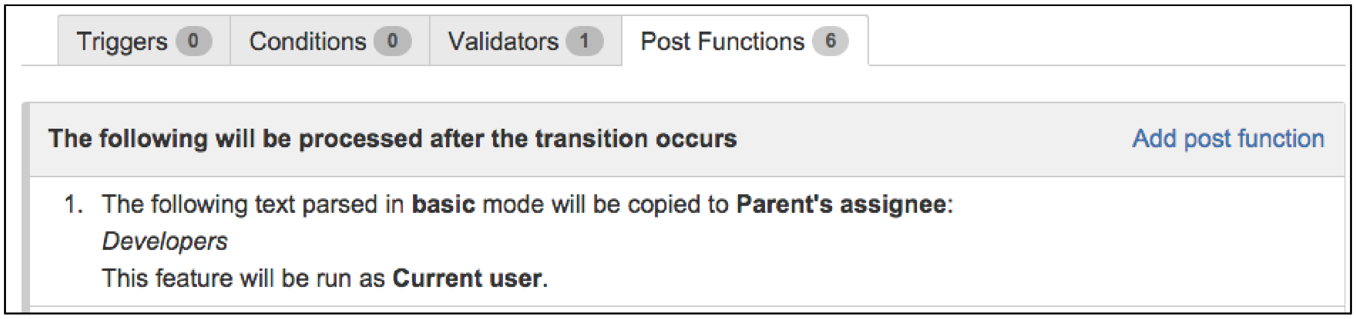
[✓ Basic] — **Basic mode:** Insert field codes anywhere in the text, and they will be replaced with corresponding field values. Field code formats are `%{nnnnn}`, and `%{nnnnn.i}` for Cascading Select fields (i = 0 for base level).

Advanced — **Advanced mode:** Strings literals are written in double quotes (`"This is a string."`). Operator '+' is used to concatenate strings, and field codes are like in basic mode, e.g., `"Issue key is " + %{00015} + "."`. More information at parser syntax documentation.

Text to be parsed and then copied to target field: [Line 1 / Col 12] Syntax Specification Check Syntax

```
Developers
```

Once configured transition's post-function tab looks like this:

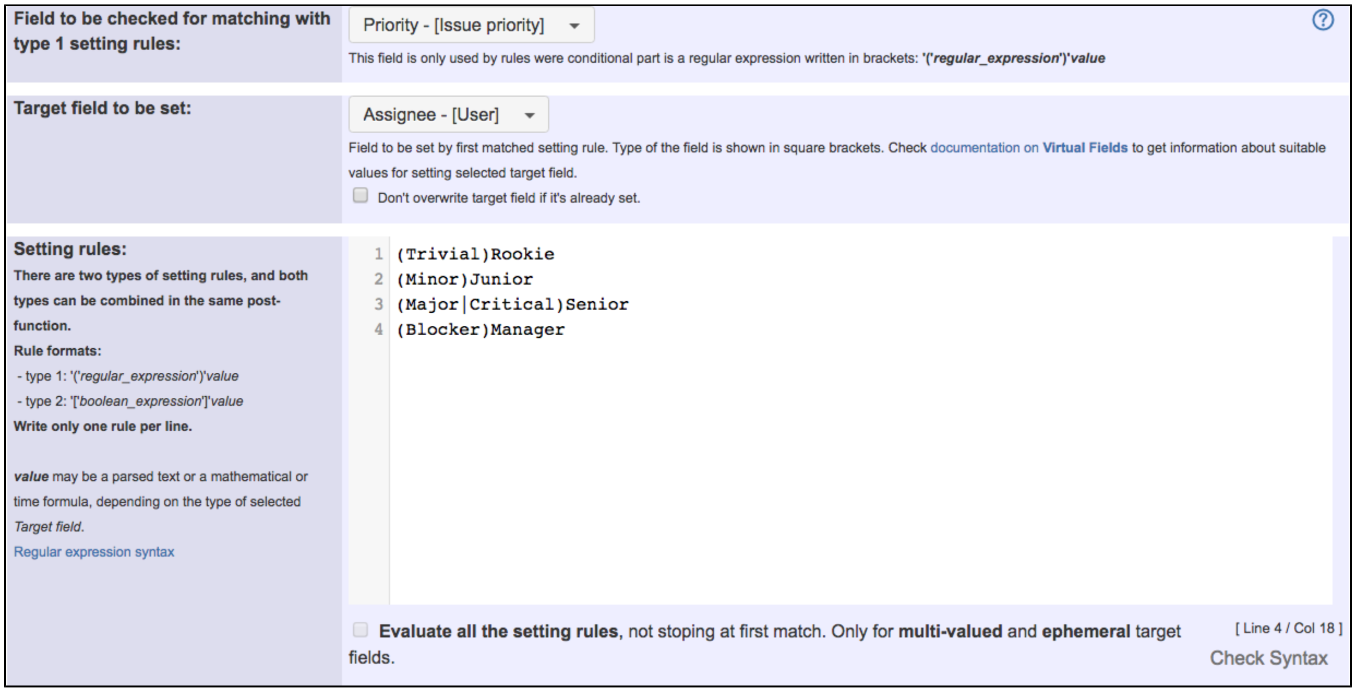


The following will be processed after the transition occurs

1. The following text parsed in **basic** mode will be copied to **Parent's assignee**:
   Developers
   This feature will be run as **Current user**.

Example 6: Assign higher priority issues to more experienced teams and lower priority issues to less experienced teams

We use post-function **Set a field as a function of other fields** to set field "**Assignee**" with a certain project role depending on issue priority. "**Rookie**", "**Junior**", "**Senior**" and "**Manager**" are **project role names** the issue will be assigned to depending on issue priority:



Field to be checked for matching with type 1 setting rules: Priority - [Issue priority]

This field is only used by rules were conditional part is a regular expression written in brackets: '('*regular_expression*')'*value*

Target field to be set: Assignee - [User]

Field to be set by first matched setting rule. Type of the field is shown in square brackets. Check documentation on **Virtual Fields** to get information about suitable values for setting selected target field.

☐ Don't overwrite target field if it's already set.

Setting rules:

There are two types of setting rules, and both types can be combined in the same post-function.

Rule formats:

- type 1: '('*regular_expression*')'*value*
- type 2: '['*boolean_expression*']'*value*

Write only one rule per line.

value may be a parsed text or a mathematical or time formula, depending on the type of selected *Target field*.

Regular expression syntax

```
(Trivial)Rookie
(Minor)Junior
(Major|Critical)Senior
(Blocker)Manager
```

☐ **Evaluate all the setting rules**, not stoping at first match. Only for **multi-valued** and **ephemeral** target fields.

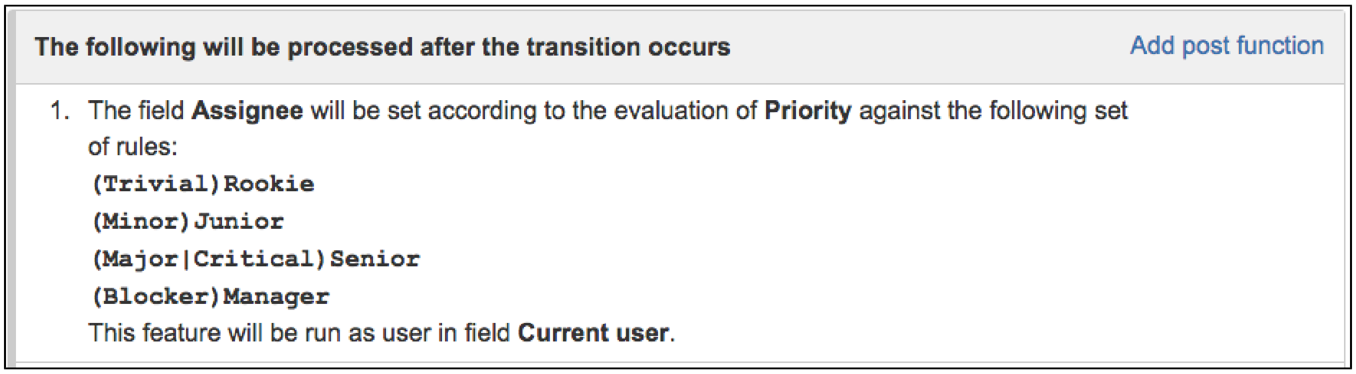
[Line 4 / Col 18]

Check Syntax

Setting rules are:

```
(Trivial)Rookie
(Minor)Junior
(Major|Critical)Senior
(Blocker)Manager
```

Once configured transition's post-function tab looks like this:

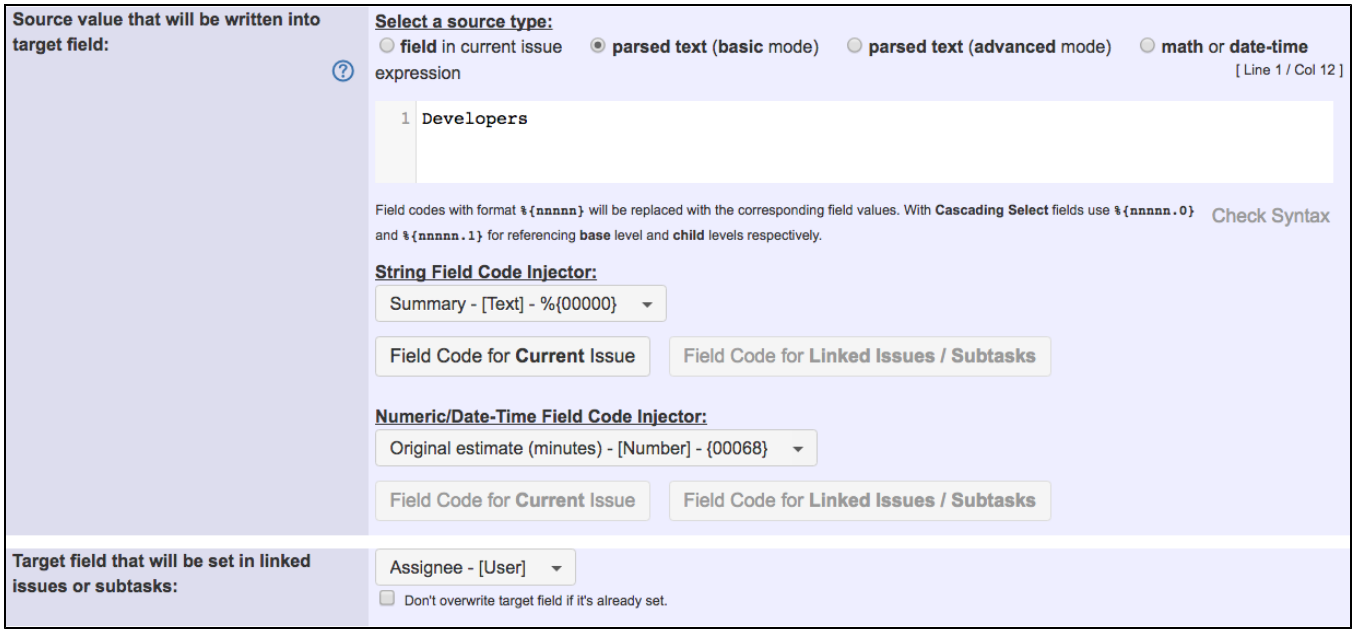


The following will be processed after the transition occurs

Add post function

1. The field **Assignee** will be set according to the evaluation of **Priority** against the following set of rules:
   ```
   (Trivial)Rookie
   (Minor)Junior
   (Major|Critical)Senior
   (Blocker)Manager
   ```
   This feature will be run as user in field **Current user**.

Example 7: Use post-function **Write field on linked issues or sub-tasks** to assign sub-tasks to project role "Developers" regardless of its type or status



Source value that will be written into target field:

Select a source type:
○ **field** in current issue [✓ ● **parsed text** (**basic** mode)] ○ **parsed text** (**advanced** mode) ○ **math** or **date-time** expression

[Line 1 / Col 12]

```
Developers
```

Field codes with format `%{nnnnn}` will be replaced with the corresponding field values. With **Cascading Select** fields use `%{nnnnn.0}` and `%{nnnnn.1}` for referencing **base** level and **child** levels respectively.

Check Syntax

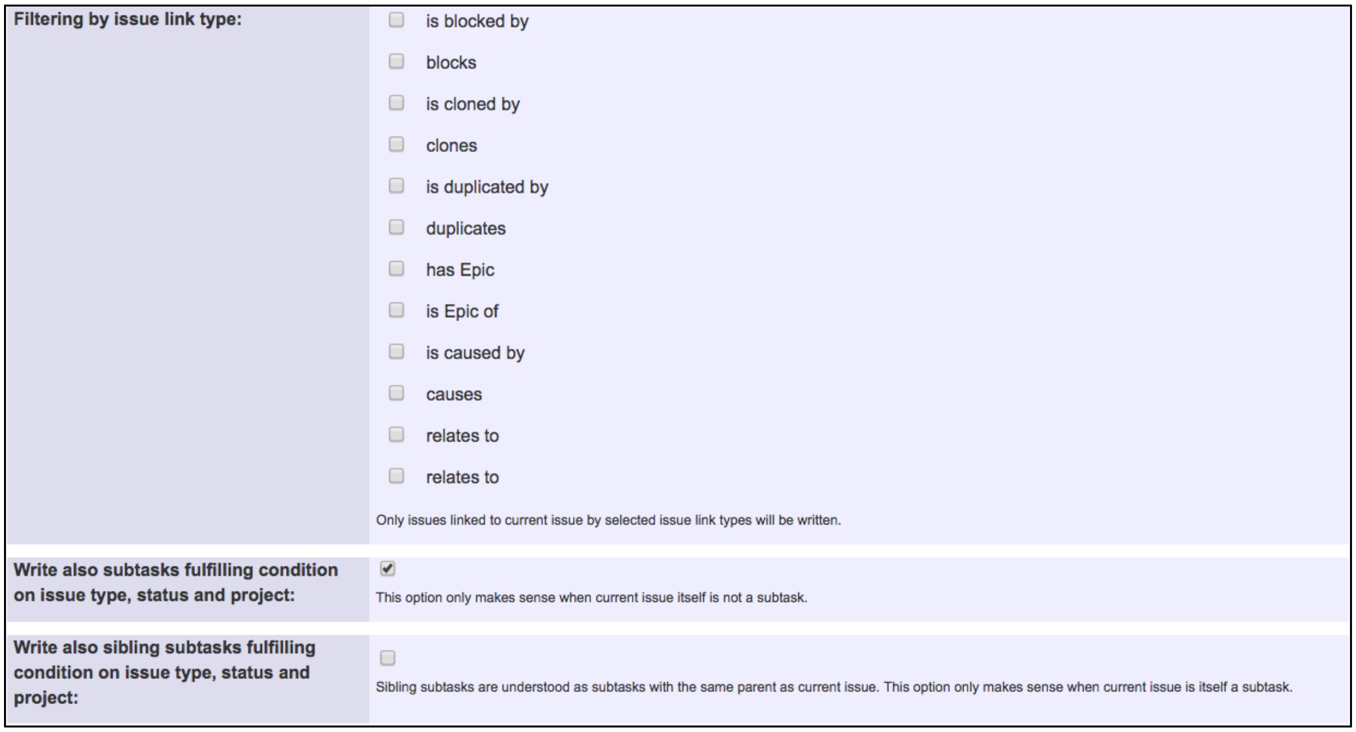
String Field Code Injector:
Summary - [Text] - %{00000}

Field Code for **Current** Issue | Field Code for **Linked Issues / Subtasks**

Numeric/Date-Time Field Code Injector:
Original estimate (minutes) - [Number] - {00068}

Field Code for **Current** Issue | Field Code for **Linked Issues / Subtasks**

Target field that will be set in linked issues or subtasks:
Assignee - [User]
☐ Don't overwrite target field if it's already set.



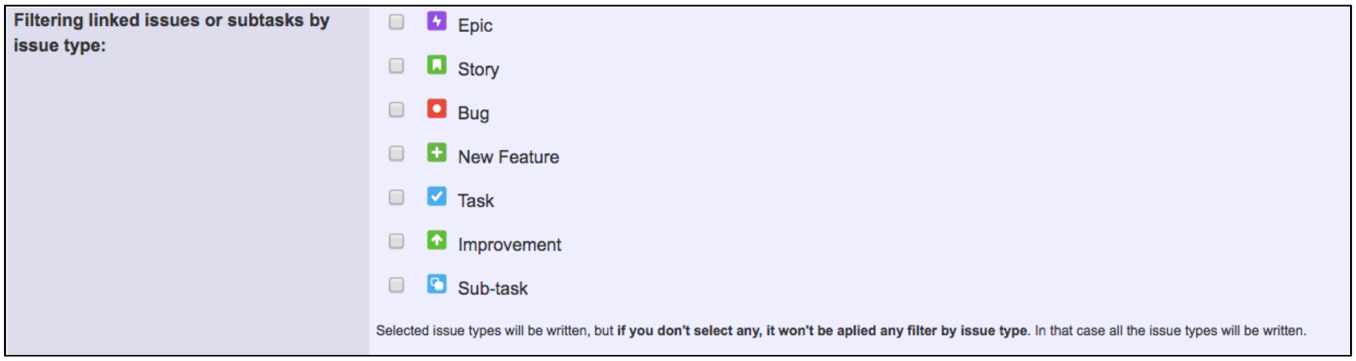
Filtering by issue link type:

- ☐ is blocked by
- ☐ blocks
- ☐ is cloned by
- ☐ clones
- ☐ is duplicated by
- ☐ duplicates
- ☐ has Epic
- ☐ is Epic of
- ☐ is caused by
- ☐ causes
- ☐ relates to
- ☐ relates to

Only issues linked to current issue by selected issue link types will be written.

Write also subtasks fulfilling condition on issue type, status and project:
☑
This option only makes sense when current issue itself is not a subtask.

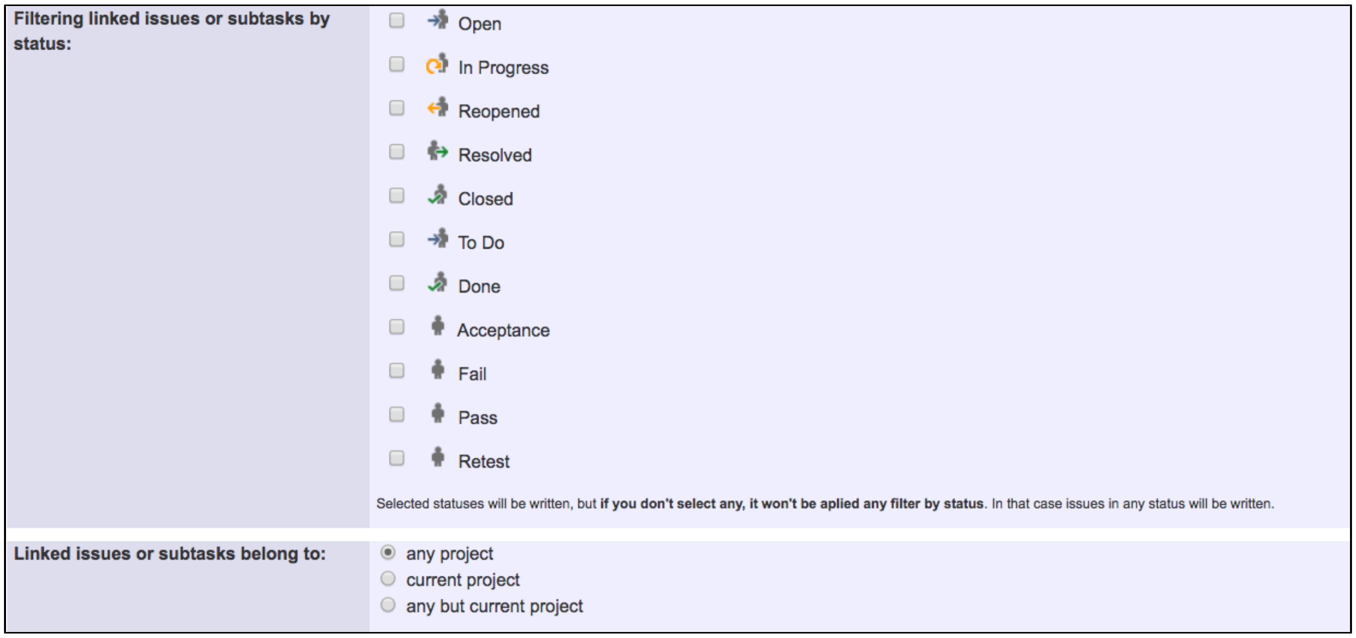
Write also sibling subtasks fulfilling condition on issue type, status and project:
☐
Sibling subtasks are understood as subtasks with the same parent as current issue. This option only makes sense when current issue is itself a subtask.



Filtering linked issues or subtasks by issue type:

- ☐ Epic
- ☐ Story
- ☐ Bug
- ☐ New Feature
- ☐ Task
- ☐ Improvement
- ☐ Sub-task

Selected issue types will be written, but **if you don't select any, it won't be aplied any filter by issue type**. In that case all the issue types will be written.



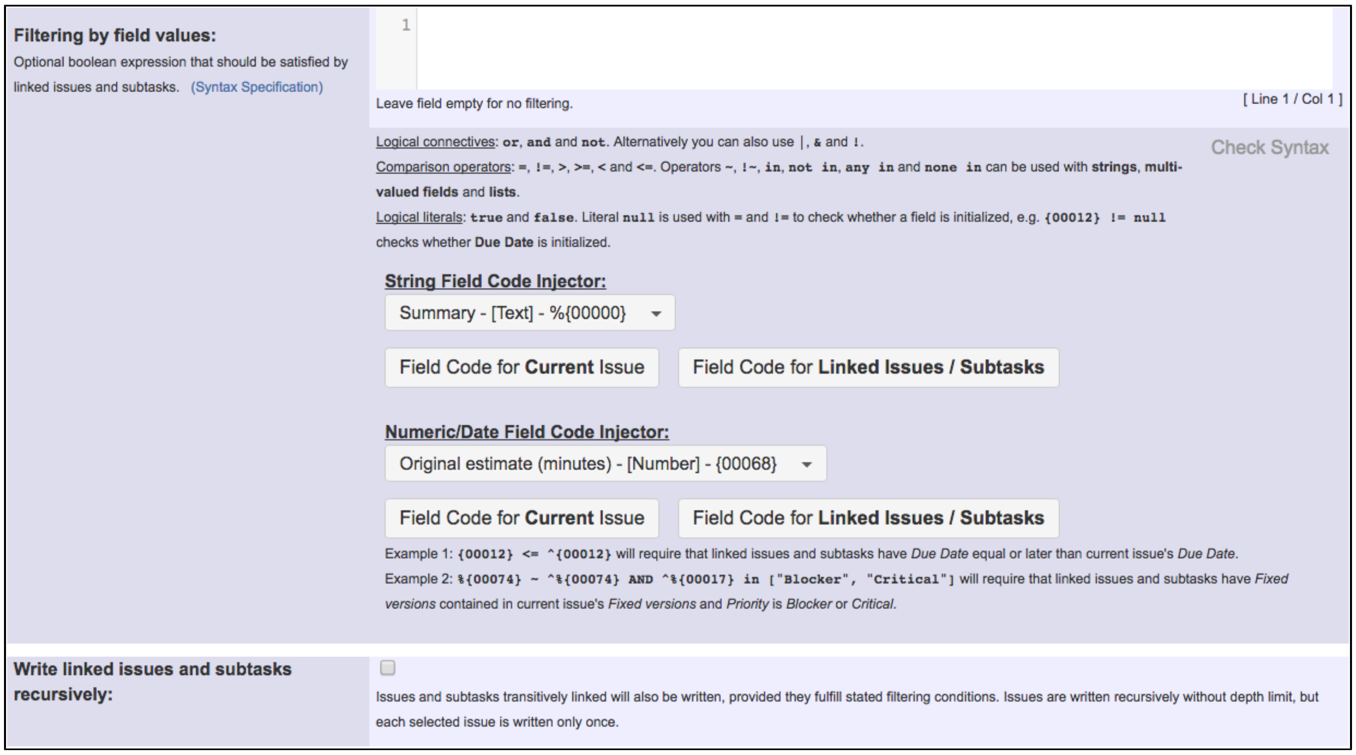
Filtering linked issues or subtasks by status:

- ☐ Open
- ☐ In Progress
- ☐ Reopened
- ☐ Resolved
- ☐ Closed
- ☐ To Do
- ☐ Done
- ☐ Acceptance
- ☐ Fail
- ☐ Pass
- ☐ Retest

Selected statuses will be written, but **if you don't select any, it won't be aplied any filter by status**. In that case issues in any status will be written.

Linked issues or subtasks belong to:

- ● any project
- ○ current project
- ○ any but current project



Filtering by field values:
Optional boolean expression that should be satisfied by linked issues and subtasks. (Syntax Specification)

Leave field empty for no filtering. [Line 1 / Col 1]

Check Syntax

Logical connectives: `or`, `and` and `not`. Alternatively you can also use `|`, `&` and `!`.
Comparison operators: `=`, `!=`, `>`, `>=`, `<` and `<=`. Operators `~`, `!~`, `in`, `not in`, `any in` and `none in` 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:
Summary - [Text] - %{00000}

Field Code for **Current** Issue | Field Code for **Linked Issues / Subtasks**

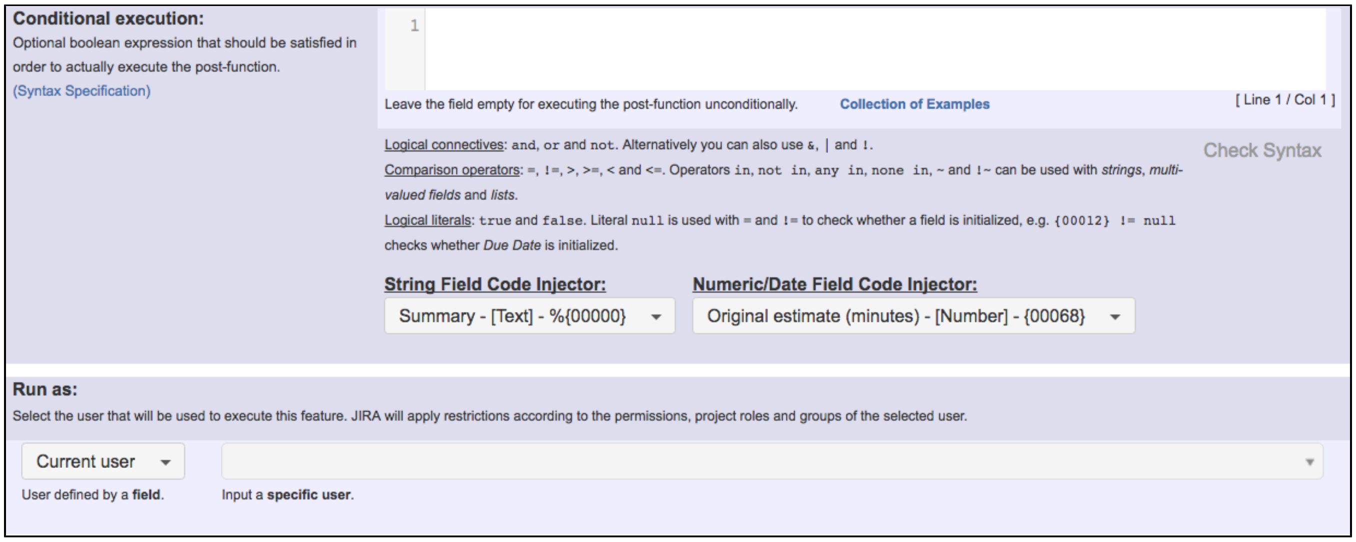
Numeric/Date Field Code Injector:
Original estimate (minutes) - [Number] - {00068}

Field Code for **Current** Issue | Field Code for **Linked Issues / Subtasks**

Example 1: `{00012} <= ^{00012}` will require that linked issues and subtasks have *Due Date* equal or later than current issue's *Due Date*.
Example 2: `%{00074} ~ ^%{00074} AND ^%{00017} in ["Blocker", "Critical"]` will require that linked issues and subtasks have *Fixed versions* contained in current issue's *Fixed versions* and *Priority* is *Blocker* or *Critical*.

Write linked issues and subtasks recursively: ☐

Issues and subtasks transitively linked will also be written, provided they fulfill stated filtering conditions. Issues are written recursively without depth limit, but each selected issue is written only once.


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.

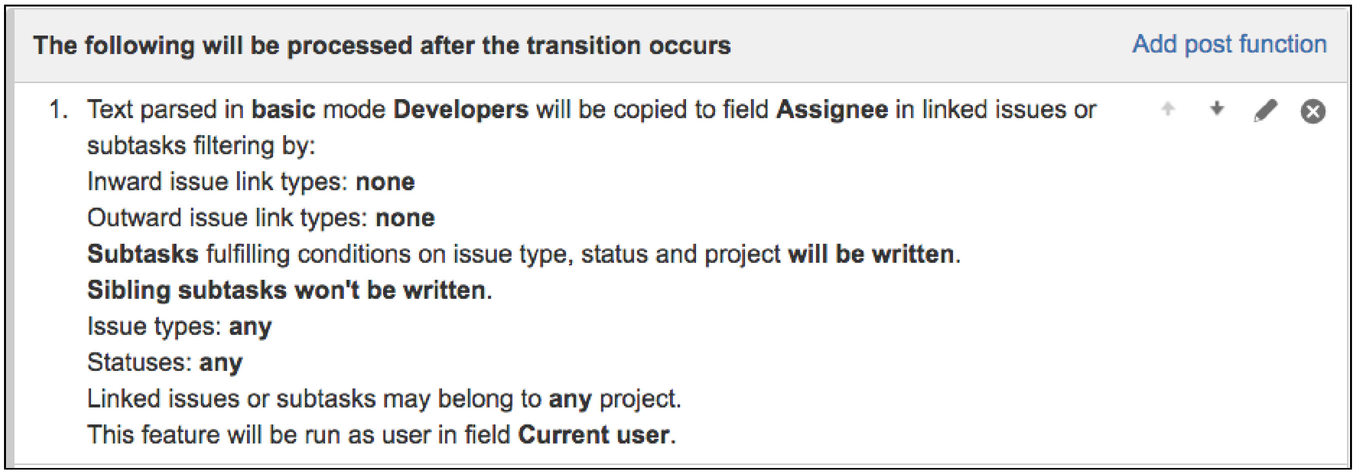
Check Syntax

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, transition's post-function tab looks like this:


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

1. Text parsed in **basic** mode **Developers** will be copied to field **Assignee** in linked issues or subtasks filtering by:
Inward issue link types: **none**
Outward issue link types: **none**
Subtasks fulfilling conditions on issue type, status and project **will be written**.
Sibling subtasks won't be written.
Issue types: **any**
Statuses: **any**
Linked issues or subtasks may belong to **any** project.
This feature will be run as user in field **Current user**.

Usage Examples

Page: Assign issue based on the value of a Cascading Select custom field
Page: Assign new issues to a different project role depending on field value in current issue

Related Features

- **Copy parsed text to a field**
- **Set a field as a function of other fields**
- **Write field on linked issues or sub-tasks**
- **Update issue fields**: