

# JWT Release Notes 2.2.33

[2017-03-08] Released [Jira Workflow Toolbox 2.2.33](#)

## New features

- New parser functions:

Function	Returned value
<b>leastBusyUserInRole</b> (string <b>projectRoleName</b> , string <b>projectKey</b> , string <b>jqlQuery</b> ) : string	Equivalent to the previous function but with extra argument <b>jqlQuery</b> , used for restricting the issues to be considered to pick the least busy user. Example: <code>leastBusyUserInRole("Developers", "%{00018}", "project = " + "%{00018}")</code> returns the user playing role <b>Developers</b> in current project, with the least number of unresolved issues in current project assigned. Note that <code>%{00018}</code> is field code for <b>Project key</b> .
<b>nextUserInGroup</b> (string <b>groupName</b> , string <b>queueName</b> ) : string	returns the name of the next active user in group with name <b>groupName</b> , for a round-robin queue with name <b>queueName</b> . The string <b>queueName</b> 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 ( <b>group</b> , <b>queue</b> ), 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: <code>nextUserInGroup("jira-developers", "code-review-queue")</code> returns the username of the next user in group <b>jira-developers</b> for round-robin queue <b>code-review-queue</b> . Each time the function is called with the same pair of arguments, a different username is returned.

- Included **round-robin** algorithm as an option for [Assign to project role](#):

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 in the past.

☐ last user in project role who has had the issue assigned in the past, 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 = 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.  
[\(Syntax Specification\)](#)

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.

String Field Code Injector:

Numeric/Date Field Code Injector:

Summary - [Text] - %{00000}

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.

## Improvements

- New parameter **JQL Query** in post-function [Assign to project role](#) for restricting issues to be considered when assigning to the **least busy user** in a project role:

Project role:
Testers

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 in the past.  
☐ last user in project role who has had the issue assigned in the past, 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 = 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 25 ]
Check Syntax

```
1 project in (PA, PB, PC)
```

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 restrict 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)

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, some 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}

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.

- Performance improvement in functionality for selecting the least busy user in a project role, either using post-function **Assign to project role** or function `leastBusyUserInRole()`.

## Bug fixes

- Issue #499** - Post-function **Create issues and sub-tasks** doesn't check whether selected issue type is valid for selected project
- Issue #534** - Virtual field **Target status** doesn't work on "Create Issue" transitions
- Issue #536** - Problems with `lastAssigneeInRole()` function and the parser's documentation