Condition and validation based on JQL query

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

Find the new documentation at:

Condition based on JQL query

Validation based on JQL query

On this page

Purpose

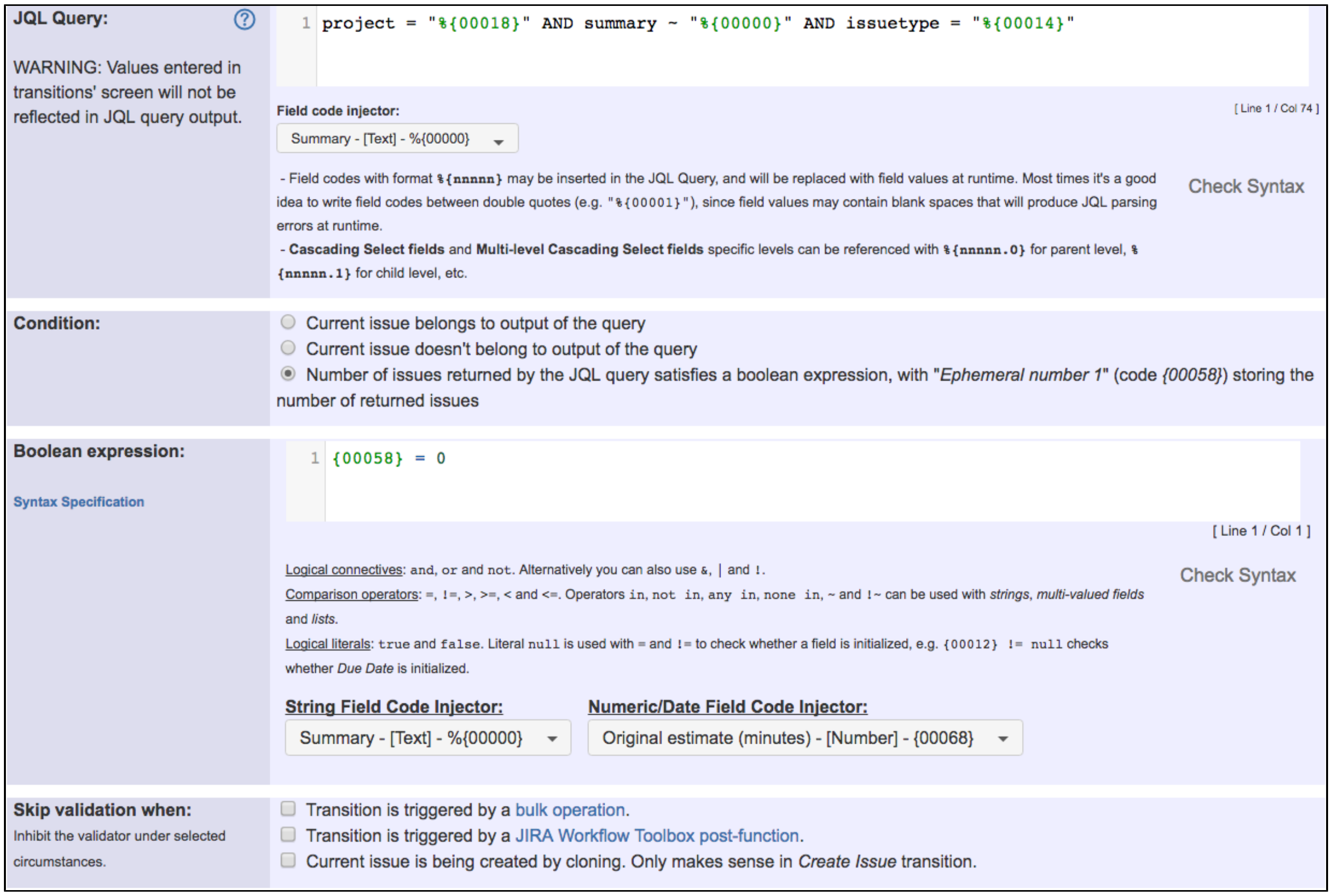
Condition based on JQL query and **Validation based on JQL query** are used for **hiding** or **blocking** a transition depending on the result of a JQL query.

Field codes in format **%{nnnnn}** can be inserted in the JQL query. They will be replaced by the corresponding field values at JQL execution time. Most of the times you will require to introduce one or more field codes in your JQL query, since you usually want to make some kind of reference to current issue **(%{00015}), current user (%{00020}), current date (%{00057}), assignee (%{00003})**, etc.

Example: Prevent creation of same issue types with identical summary

Condition based on JQL query and **Validation based on JQL query** have 3 configuration parameters. In the case of the validator you also have a forth parameter called "**Message to show when validation fails**" which is a custom message to show the user to inform him the cause that prevents him from executing the transition.

In the following screenshot it's shown the way to config a validator in transition "**Create Issue**", in order to prevent issue creation whenever there already is an issue with the **same summary**, and **same issue type**, in the **same project**:



JQL Query:

WARNING: Values entered in transitions' screen will not be reflected in JQL query output.

```
project = "%{00018}" AND summary ~ "%{00000}" AND issuetype = "%{00014}"
```

[Line 1 / Col 74]

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

Condition:

- ○ Current issue belongs to output of the query
- ○ Current issue doesn't belong to output of the query
- ● Number of issues returned by the JQL query satisfies a boolean expression, with "*Ephemeral number 1*" (code *{00058}*) storing the number of returned issues

Boolean expression:

Syntax Specification

```
{00058} = 0
```

[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.

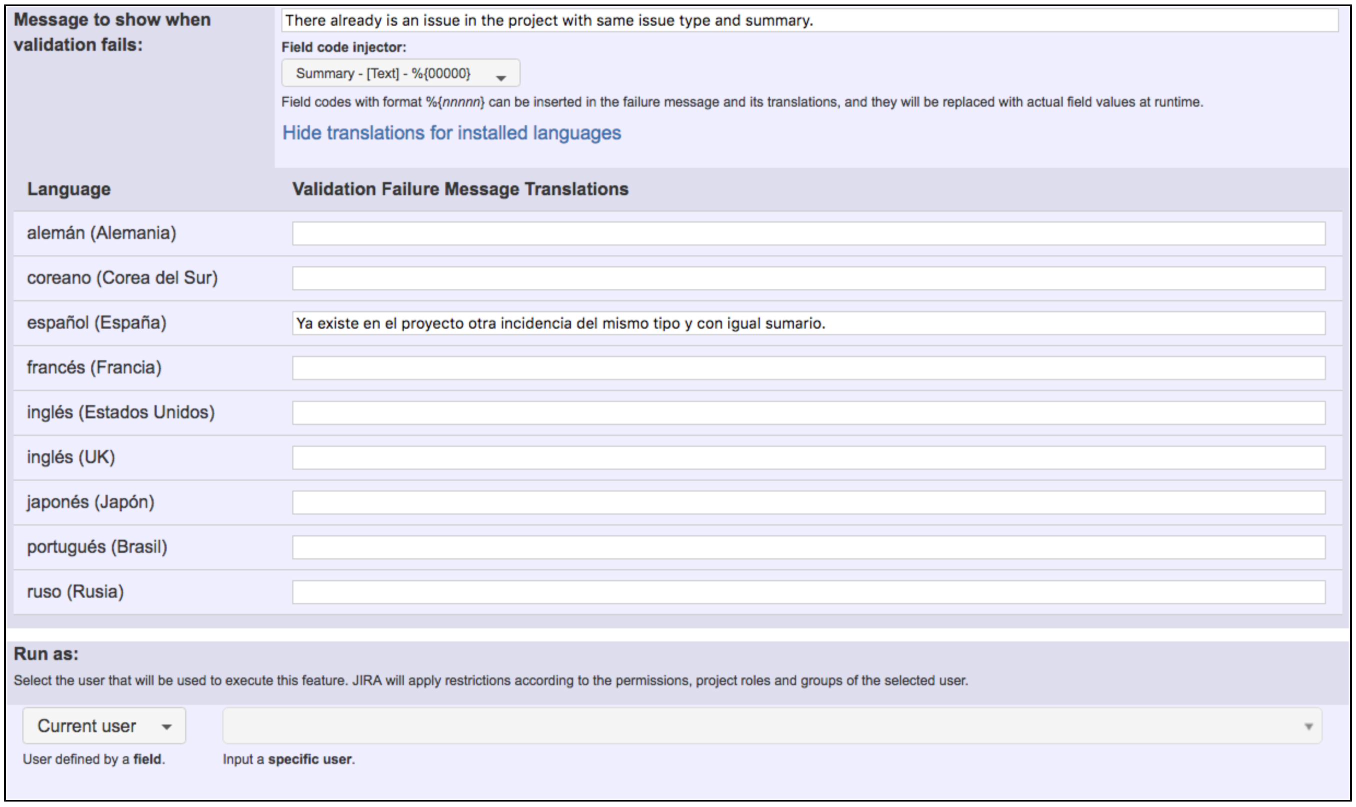
Check Syntax

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

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

Skip validation when:
Inhibit the validator under selected circumstances.

- ☐ Transition is triggered by a bulk operation.
- ☐ Transition is triggered by a JIRA Workflow Toolbox post-function.
- ☐ Current issue is being created by cloning. Only makes sense in *Create Issue* transition.



Message to show when validation fails: There already is an issue in the project with same issue type and summary.

Field code injector: Summary - [Text] - %{00000}

Field codes with format %{*nnnnn*} can be inserted in the failure message and its translations, and they will be replaced with actual field values at runtime.

Hide translations for installed languages

Language	Validation Failure Message Translations
alemán (Alemania)	
coreano (Corea del Sur)	
español (España)	Ya existe en el proyecto otra incidencia del mismo tipo y con igual sumario.
francés (Francia)	
inglés (Estados Unidos)	
inglés (UK)	
japonés (Japón)	
portugués (Brasil)	
ruso (Rusia)	

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:

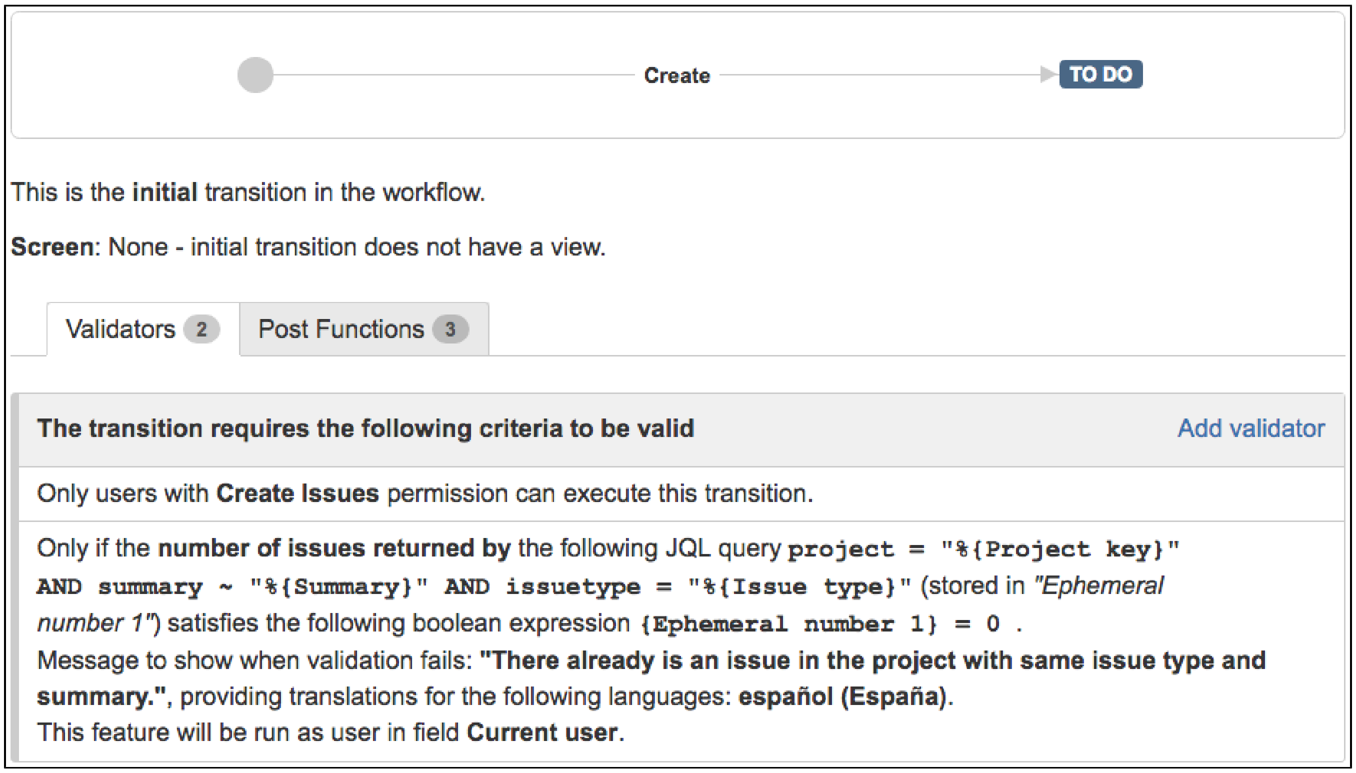
- **%{00018}** is field code for "**Project key**"
- **%{00000}** is field code for "**Summary**"
- **%{00014}** is field code for "**Issue type**"
- **{00058}** is code for numeric value of "**Ephemeral number 1**"

This might not work for issues where the summary contains special characters since the default search will be performed as a **fuzzy search**.

You can perform an exact search, however, if you use the following syntax **\"%{00000}\"**

```
project = "%{00018}" AND issuetype = "%{00014}" AND summary ~ "\"%{00000}\""
```

Once configured, transition looks like this:



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

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

The transition requires the following criteria to be valid

Only users with **Create Issues** permission can execute this transition.

Only if the **number of issues returned by** the following JQL query **`project = "%{Project key}" AND summary ~ "%{Summary}" AND issuetype = "%{Issue type}"`** (stored in *"Ephemeral number 1"*) satisfies the following boolean expression **`{Ephemeral number 1} = 0`** .
Message to show when validation fails: **"There already is an issue in the project with same issue type and summary."**, providing translations for the following languages: **español (España)**.
This feature will be run as user in field **Current user**.

Configuration Parameters

Parameter "JQL Query"

This parameter is a JQL query which can contain field codes in format **%{nnnnn}**. This query will be run when the validator is invoked by transition execution, but previously all field codes will be replaced with its corresponding value. In most cases these field codes are used to refer to current issue, current user, current project, current date and time, etc.
As a result of JQL query execution we get a set of issues, and we will make our condition or validator depend in a certain way of them.

When you write your JQL for selecting the issues you want to write, 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**

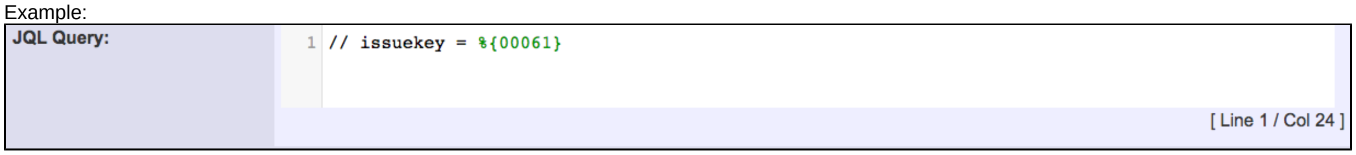
separated list of values, and we want to use it with JQL operator **IN**. In those cases we will not write the field code between quotes, since we want the content of the field to be processed as a **list of values**, not as a single string value.

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.

Disabling JQL Syntax Pre-Checking:

When we enter our JQL query, a syntax pre-checking is carried out in order to verify that it's correctly written. But when we insert field codes in our JQL query, the definitive form of the query that will be executed is unknown, since it depends on the actual values of the fields in runtime. In these cases the syntax pre-checking is done with speculative values given to the fields, and it might happen that fake syntax errors are reported. In order to inhibit the JQL syntax pre-checking you should enter `//` at the beginning of the line. Those characters will be removed in the actual JQL query that will be executed.

Example:



JQL Query:

```
// issuekey = %{00061}
```

[Line 1 / Col 24]

Parameter "Condition"

This parameter sets the way we want to make our condition or validator depend of the result of the JQL query:

- **current issue belongs to output of the query**: if current issue is among issues returned by the JQL query.
- **current issue doesn't belong to output of the query**: if current issue is not among issues returned by the JQL query.
- **number of issues returned by the JQL query...**: depending on the number of issues returned by the JQL query. This dependence is set in the next parameter "**Mathematical expression**".

Parameter "Mathematical expression"

This is a boolean expression for evaluating the number of issues returned by the JQL query. The number of issues returned is stored in virtual field **"Ep hemeral number 1" (field code {00058})**.

Syntax of these expressions is the same as used for **Boolean condition and validator with math. date-time or text-string terms**.

Additional parameters:

- **Run as**: JIRA user post-function is going to be executed as. This parameter can be set to a **fixed user** (e.g. "john.nash"), or to a **user field** (e.g. "Reporter", "Assignee", etc). This parameter is particularly important in this feature since JQL query will return issues according to the browse permission this user has in the different project of the instance of JIRA.

- **Message to show when validation fails**: This parameter is only available in the **validator** form of this feature. It's a **custom message** to explain the user the cause that is preventing him from executing the transition.
  Translations* to every language installed in the JIRA instance can be optionally entered.
  In **Condition based on JQL query** the transition is simply hidden whenever condition isn't satisfied, so this parameter is not present.

Usage Examples

Page: Make linked issues, sub-tasks and JQL selected issues progress through its workflows
Page: Prevent issue creation if another issue with same field value already exists

Related Features

- **Block or hide a transition for an issue depending on its issue links**
- **Condition and validation on sub-tasks**
- **Boolean condition and validator with math. date-time or text-string terms**