

Validation on sibling sub-tasks depending on issue type and status

On this page

- [Features used to implement the example](#)
- [Example: Validation on sibling sub-tasks depending on issue type and status](#)
- [Other variation of the usage example](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Boolean validator with math, date-time or text-string terms](#)

Example: Validation on sibling sub-tasks depending on issue type and status

I have one Issue with two sub-tasks, say **sub-task A (of issue type A)** and **sub-task B (of issue type B)**.

I would like to add a validation on a transition of **sub-task A** that allows that transition to be executed only if the the value of the **Status Field** of **sub-task B** is for example **"Open" OR "In Progress" OR "Reopened"**. For any other value of the **sub-task B's Status Field**, the transition of **sub-task A** will be forbidden.

That validation can be implemented with [Jira Workflow Toolbox 2.1.20](#) or higher, using [Boolean validator with math, date-time or text-string terms](#). I will show how to implement two variants of the validation. You can choose the one that better fits your use case:

I suppose that parent issue has at least one sub-task of type **"issue type B"**, and we want to validate that at least one of them is in the status **"Open"**, **"In Progress"** or **"Reopened"**. To implement that validation we use the validator with the following configuration:



```
1 count(filterByStatus(filterByIssueType(subtasks(#{00041}), "Issue Type B"), "Open, In Progress, Reopened")) > 0
```

Logical connectives: and, or 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.

Check Syntax

NUMERICAL AND DATE-TIME TERMS

Numeric and Date-Time field values: insert field codes with format {nnnnn}.

Original estimate (minutes) - [Number] - {00068} ▾

Insert Numeric Value

Valid date-time literal formats: yyyy/MM/dd [hh:mm] or yyyy-MM-dd [hh:mm]. Time literals use format: hh:mm.

There is a set of **mathematical functions** and **time macros and functions** available to be used in your expression.

TEXT-STRING TERMS

Text-String field values: insert field codes with format %{nnnnn} or %{nnnnn.i} for referencing levels in cascading select fields (i = 0 for base level).

Summary - [Text] - %{00000} ▾

Insert String Value

String literals: written in **double quotes**, e.g., "This is a string literal."

String concatenation: use operator '+' to concatenate string values, e.g., "The summary of issue with key " + %{00015} + " is \" + %{00000} + "\"."

Escape character: character \ is used with characters "", \, \n, \r, \t, \f and \b to invoke an alternative interpretation.

There is a set of **string functions** available to be used in your expression.

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:

At least one sibling subtask of type "Issue Type B" must be in status "Open", "In Progress" or "Reopened".

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.

[Set translations for installed languages](#)

Boolean expression used is: `count(filterByStatus(filterByIssueType(subtasks(#{00041}), "Issue Type B"), "Open, In Progress, Reopened")) > 0`

Note that:

- **{00041}** is code for virtual field **"Parent's issue key"**
- **"Issue type B"** is the name of a issue type

Once configured, the transition will look like this:

Triggers 0 Conditions 0 Validators 1 Post Functions 9

The transition requires the following criteria to be valid [Add validator](#)

Only if the following boolean expression is true: **`count(filterByStatus(filterByIssueType(subtasks(%
{Parent's issue key}), "Issue Type B"), "Open, In Progress, Reopened")) > 0`**

Message to show when validation fails: **At least one sibling subtask of type "Issue Type B" must be in statuses "Open", "In Progress" or Reopened".**

Other variation of the usage example

I suppose that parent issue may have or may not have sub-tasks of type "Issue type B". We want to validate that every sub-task of type "Issue type B" is in the status "Open", "In Progress" or "Reopened".

To implement that validation we use [Boolean validator with math, date-time or text-string terms](#) with the following configuration:



```
1 count(filterByStatus(filterByIssueType(subtasks({00041}), "Issue Type B"), "Open, In Progress, Reopened")) =
count(filterByIssueType(subtasks({00041}), "Issue Type B"))
```

Logical connectives: and, or 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**.

[Check Syntax](#)

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.

NUMERICAL AND DATE-TIME TERMS

Numeric and Date-Time field values: insert field codes with format {nnnnn}.

Original estimate (minutes) - [Number] - {00068} ▾

Insert Numeric Value

Valid date-time literal formats: yyyy/MM/dd [hh:mm] or yyyy-MM-dd [hh:mm]. Time literals use format: hh:mm.

There is a set of **mathematical functions** and **time macros and functions** available to be used in your expression.

TEXT-STRING TERMS

Text-String field values: insert field codes with format %(nnnnn) or %(nnnnn.i) for referencing levels in cascading select fields (*i* = 0 for base level).

Summary - [Text] - %{00000} ▾

Insert String Value

String literals: written in **double quotes**, e.g., "This is a string literal."

String concatenation: use operator '+' to concatenate string values, e.g., "The summary of issue with key " + %{00015} + " is \" + %{00000} + "\"."

Escape character: character \ is used with characters "", \, 'n', 'r', 't', 'f' and 'b' to invoke an alternative interpretation.

There is a set of **string functions** available to be used in your expression.

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:

All sibling subtasks of type "Issue Type B" must be in status "Open", "In Progress" or "Reopened".

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.

[Set translations for installed languages](#)

Boolean expression used is: `count(filterByStatus(filterByIssueType(subtasks({00041}), "Issue Type B"), "Open, In Progress, Reopened")) = count(filterByIssueType(subtasks({00041}), "Issue Type B"))`

Note that:

- **{00041}** is code for virtual field **"Parent's issue key"**
- **"Issue type B"** is the name of a issue type

Once configured, the transition will look like this:

Triggers 0

Conditions 0

Validators 1

Post Functions 9

The transition requires the following criteria to be valid

Add validator

Only if the following boolean expression is true: `count(filterByStatus(filterByIssueType(subtasks(%{Parent's issue key}), "Issue Type B"), "Open, In Progress, Reopened")) = count(filterByIssueType(subtasks(%{Parent's issue key}), "Issue Type B"))`

Message to show when validation fails: All sibling subtasks of type "Issue Type B" must be in statuses "Open", "In Progress" or Reopened".

Other examples of that function

Page: Block a transition until all sub-tasks have certain fields populated
Page: Block an epic's transition depending on linked issues status and due date
Page: Block or hide a transition for an issue depending on its issue links
Page: Block or unblock a transition after an issue rested a specific time in a status
Page: Block transition until all sub-tasks are in a specific status category
Page: Close parent issue when all sub-tasks are closed
Page: Enforce a field (Select List) to be set when another field (Radio Button) has a certain value (works with any kind of field type)
Page: Ensure that all issues linked with a certain issue link type have "Due Date" field set
Page: If field A is populated then, field B must also be populated
Page: Limit issue creation per role and issue type
Page: Limit the number of hours a user can log per day
Page: Limit valid dates for work logs
Page: Make "Time Spent" field required when there is no time logged in the issue
Page: Make a custom field mandatory when priority is "Critical" or "Blocker" and issue type is "Incident"
Page: Make attachment mandatory depending on the value of certain custom field
Page: Make different fields mandatory depending on the value of a Select List custom field
Page: Make linked issues, sub-tasks and JQL selected issues progress through its workflows
Page: Make parent issue progress through its workflow
Page: Prevent issue creation if another issue with same field value already exists
Page: Reject duplicated file names in attachments
Page: Require at least one sub-task in status "Resolved" or "Closed" when "Testing required" is selected in Check-Box custom field
Page: Require issue link when resolving as duplicate
Page: Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status
Page: Restrict sub-task type creation depending on parent issue status
Page: Restrict sub-task type creation depending on parent issue type
Page: Set a condition in a global transition which only applies in a certain status
Page: Validate a custom field "Story Points" has been given a value in Fibonacci sequence
Page: Validate compatible values selection among dependent custom fields
Page: Validate only issue links created in transition screen
Page: Validate that multi-user picker custom field A does not contain any user in multi-user picker custom field B
Page: Validation and condition based on time expressions
Page: Validation based on the value of a date type project property
Page: Validation on issue attachments
Page: Validation on MIME types of issue attachments
Page: Validation on sibling sub-tasks depending on issue type and status
Page: Validation on the value of a Cascading Select field

Related Usage Examples

- Validation on sibling sub-tasks depending on issue type and status
 - example
 - validator
 - sub-task
 - transition
- Block a transition until all sub-tasks have certain fields populated
 - example
 - condition
 - validator
 - sub-task
 - transition
- Transition sub-tasks when parent is transitioned
 - 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
- Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress"
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Automatically close resolved sub-tasks when parent issue is closed
 - example
 - post-function
 - sub-task
 - transition
 - outdated
- Change parent's status depending on sub-task's summary
 - example
 - post-function
 - sub-task

- transition
 - outdated
- Close parent issue when all sub-tasks are closed
 - example
 - condition
 - validator
 - post-function
 - sub-task
 - transition
- Proceed with a task only when all sub-tasks are completed
 - example
 - condition
 - validator
 - sub-task
 - transition
- Prevent transitioning when there is a blocking issue
 - example
 - validator
 - issue-links
 - sub-task
 - transition
- Transition parent issue only when certain issue sub-task types are done
 - example
 - validator
 - sub-task
 - transition
- Enforce certain type of sub-tasks to be "Resolved" to allow executing a transition
 - example
 - validator
 - sub-task
 - transition
- Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status
 - example
 - validator
 - post-function
 - sub-task
 - transition