

Make parent issue progress through its workflow

On this page

- [Features used to implement the example](#)
- [Example: Reopen parent issue when a sub-task is reopened](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Copy parsed text to a field](#)

A sub-task can execute a post-function in order to move its parent issue through the workflow. For example, we can [close a parent issue when all sub-tasks are closed](#).

How to make parent issue progress through its workflow?

A sub-task can make its parent issue transition using 2 different sets of virtual fields:

- **"Parent's issue status"** and **"Parent's issue status (delayed writing)"**: if we write the **name of a status** (e.g., **Open**, **In Progress**, **Closed**, ...) into any of these 2 virtual fields, the plugin will look for transitions in parent's workflow going from parent's current status to a status with the same name as the one we have written. If a suitable transition is found and its conditions and validations are satisfied, the transition will be executed, and parent issue will be moved to entered status.
- **"Execute transition on parent issue"** and **"Execute transition on parent issue (delayed execution)"**: if we write the **name of a transition** (e.g., **Start Progress**, **Close Issue**, ...) into any of these 2 virtual fields, the plugin will look for a transition in parent's workflow departing from parent's current status with the entered name. If that transition exists and its conditions and validations are satisfied, the transition will be executed.

In order to transition parent issue, a sub-task typically uses post-function [Copy parsed text to a field](#) with the following configuration:

- Target field: any of the 4 virtual fields above described.
- Parsing mode: **basic**
- Text to be parsed...: the **name of a status** or the **name of a transition**, depending of the selected target field.

Also [Set a field as a function of other fields](#) can be used.

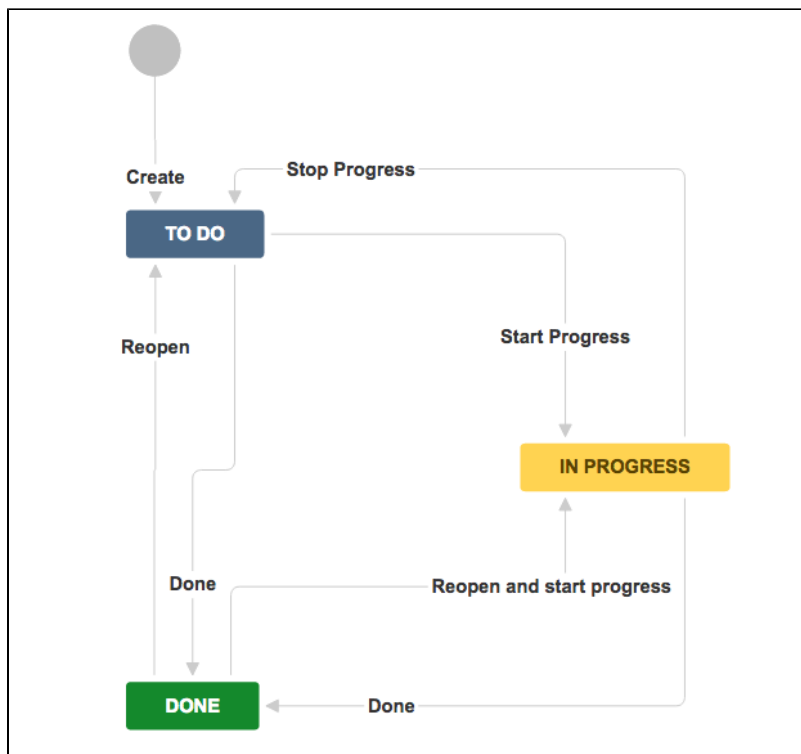
Difference between "Parent's issue status" and "Parent's issue status (delayed writing)" or "Execute transition on parent issue" and "Execute transition on parent issue (delayed execution)"

Virtual fields **"Parent's issue status"** and **"Execute transition on parent issue"** try to transition parent issue **as soon as they are written**, while fields **"Execute transition on parent issue (delayed execution)"** and **"Execute transition on parent issue (delayed execution)"** **wait for sub-task's current transition to end execution**, before transitioning parent issue.

When a sub-task uses **"Execute transition on parent issue (delayed execution)"** or **"Execute transition on parent issue (delayed execution)"** fields for transitioning its parent issue, parent's execution attempt won't be carried out until current sub-task's transition has finished, and thus sub-task is in the new status. For this reason, these fields are used when parent issue is **blocked by a condition or validator depending on sub-tasks statuses**, typically by a conditions or validators like "Sub-Task Blocking Condition", [Condition and validation on sub-tasks](#) and [Boolean condition and validator with math. date-time or text-string terms](#), etc.

Example: Reopen parent issue when a sub-task is reopened

In this example we show how to automatically reopen an issue in status **"Done"** when any of its sub-tasks is reopened. Let's assume the following workflow:



We will insert post-function [Copy parsed text to a field](#) into transitions **"Reopen"** and **"Reopen and start progress"** with the following configuration:

Target field:

Parent's issue status - [Issue status]

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 13]

Syntax Specification

Check Syntax

1 In Progress

Summary - [Text] - %{00000}

Insert String Value

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

Insert Numeric Value

- **Compose dynamic text** by inserting field codes (`%{nnnnn}`) that will be replaced with corresponding field values prior to be copied to target field.

- You can reference parent and child values of **cascading select fields** writing `%{nnnnn.0}` for parent value, and `%{nnnnn.1}` for child value. Use greater indexes to read **multi-level cascading select** custom fields.

- You can change **reporter**, **assignee**, **due date**, **issue status**, **priority**, **resolution**, **labels**, **components**, **fixed versions**, **affected versions**, **original estimate**, **estimated**, **time spent**, and **security level** by 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 box**, **Multi user**, **Multi groups**, **Components** 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)/`.

- **Setting and unsetting individual values** in multi-valued fields, leaving the rest untouched, can be achieved simply by inserting a character '+' or '-' preceding the value or the list of values. You can insert more than one '+' / '-' character in a sole setting operation.

- Fields **Attachments (only new attachments will be added)** and **Attachments (all current attachments will be replaced)** expect one or more issue keys whose attachments will be copied to current issue.

- You can also use this post-function to **cast a string into a number**.

Conditional execution:

Optional boolean expression that should be satisfied in order to actually execute the post-function.

(Syntax Specification)

1 `%{00042} = "Done"`

Leave the field empty for executing the post-function unconditionally.

Collection of Examples

[Line 1 / Col 18]

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:

Parent's issue status - [Issue status] - %{00042}

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.

Note that:

- `%{00042}` is field code for **Parent's issue status**
- We haven't written the name of the statuses in upper case, but like **In Progress** and **Done**, because Jira UI typically shows the name of statuses in upper case, while the actual names are not. You can check up the actual names at Jira Administration screens

Once configured, transitions **"Reopen"** and **"Reopen and start progress"** will look like this:

DONE
Reopen
TO DO

Screen: None - it will happen instantly

Triggers 0
Conditions 1
Validators 0
Post Functions 7

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

- The following text parsed in **basic** mode will be copied to **Parent's issue status**:
In Progress
Post-function will only be executed if the following boolean expression is satisfied: `%{Parent's issue status} = "Done"`
This feature will be run as user in field **Current user**.
- The **Resolution** of the issue will be **cleared**.
- Set issue status to the linked status of the destination workflow step.
- Add a comment to an issue if one is entered during a transition.
- Update change history for an issue and store the issue in the database.
- Re-index an issue to keep indexes in sync with the database.
- Fire a **Generic Event** event that can be processed by the listeners.

Other examples of that function

[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](#)

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](#)

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

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