# Update any Issue Field Post-Function

**JSU for Jira Server/Data Center**

This is the documentation of JSU for Jira Server/Data Center. If you are using JSU on Jira Cloud, you can find the documentation here.

## Description

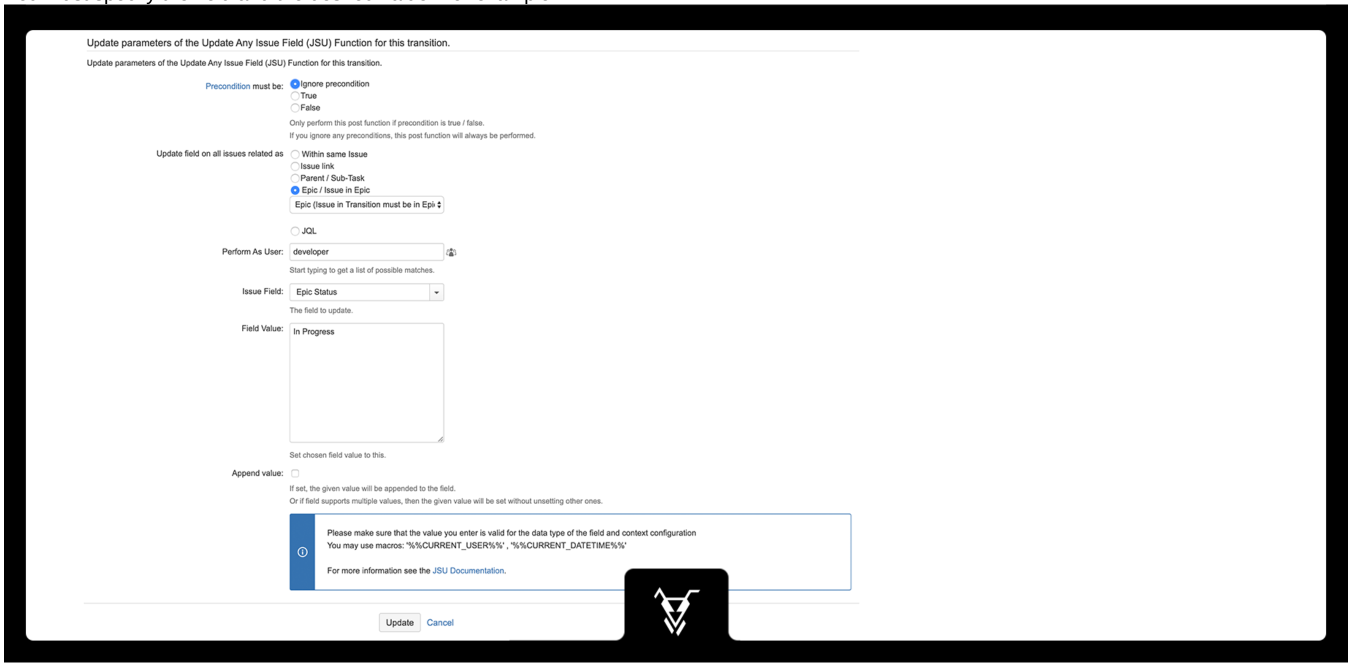
The 'Update any Issue Field' post-function updates any field to a specified value, after a transition has been completed. This can be a system- or a custom field.

The field can be on the issue in transition(within same issue) or on a related issue, like a sub-task, a linked issue or an issue within an Epic (during the transition on the Epic).

In addition to setting values to fields, the 'Update any Issue Field' post-function can also add comments.

## Configuration

You must specify the field and the desired value. For example:



Update parameters of the Update Any Issue Field (JSU) Function for this transition.

Update parameters of the Update Any Issue Field (JSU) Function for this transition.

Precondition must be: [✓ Ignore precondition] True False

Only perform this post function if precondition is true / false.
If you ignore any preconditions, this post function will always be performed.

Update field on all issues related as: Within same Issue, Issue link, Parent / Sub-Task, [✓ Epic / Issue in Epic], JQL

Epic (Issue in Transition must be in Epi

Perform As User: developer

Start typing to get a list of possible matches.

Issue Field: Epic Status

The field to update.

Field Value: In Progress

Set chosen field value to this.

Append value: ☐

If set, the given value will be appended to the field.
Or if field supports multiple values, then the given value will be set without unsetting other ones.

Please make sure that the value you enter is valid for the data type of the field and context configuration
You may use macros: '%%CURRENT_USER%%' , '%%CURRENT_DATETIME%%'

For more information see the JSU Documentation.

## Precondition

There are several ways in which preconditions can be evaluated in the context of a post function:

- Ignore precondition (By default every precondition is ignored. It means that post function will be always performed)
- True (Precondition must be true to execute a post function)
- False (Precondition must be false to execute a post function)

You can find more generic description of precondition here

We use the Update any Issue Field post function like this in the 'Start Progress' transition of our Story issue types. If the Story is part of an Epic, the Epic Status will be set to 'In Progress'.

In the past users forgot to change the Epic Status in time and it was left behind as 'To Do'. Using JSU this does no more happen.

## Update field on all issues related as

The field can be on the issue in transition(within same issue) or on a related issue, like a sub-task, a linked issue or an issue within an Epic (during the transition on the Epic).

See Related Issues for more explanation on this topic.

## Perform As User

If you don't specify anything here, the transition on the related will be performed under the same user, who triggered this post function on the origin issue. Thus that user must have the necessary permissions on the related issue.

However, in some restrictive setup that user might not be allowed to do so on the related issue. He might not even see the project of the related issue!

With 'Perform As User' you can specify a different user account which owns the necessary permissions. Usually this user account is assumed to be only technical (impersonation), with broad permissions, but not used to log into Jira by in life persons.

In combination with the 'User is in Any Users' Condition, you can hide a transition from all other users than the 'Perform As User' user. For further example go here.

## Field Value

Please make sure that the value you enter is valid for the datatype of the selected field. Also verify, that the context configuration for the project using this workflow will allow to modify the selected field.
⚠ Otherwise, the transition may fail at execution time.

Typically you will use text or numbers as value.

### Cascading Select fields

For Cascading Select fields, you may either use the value of the option you would like to set, or it's id. In either case, no need to specify the parent option. For example:

- Vehicles
    - Car
    - Train
    - Bus
- Buildings
    - House
    - Skyscraper

Using **Vehicles** as the parameter for **Field Value** would set the field content to that very option. Same if you would choose **Train**. Or, assuming **10701** is the ID of **House**, then the option **House** will be set.

### Special macros

If you use
`%%CURRENT_USER%%`
as the field value (exactly this, nothing more), the user who triggered the post function will be set as value.

If you use
`%%CURRENT_DATETIME%%`
as the field value (exactly this, nothing more), the current date and time will be set as value.

If you use
`%%ADD_CURRENT_USER%%`
as the field value (exactly this, nothing more), the user who triggered the post function will be appended to the existing field content.

**Obsolete since 1.4.10: %%ADD_CURRENT_USER%%**

Please use the option 'Append value' combined with the macro '%%CURRENT_USER%%' instead.

## Position of the Post Function

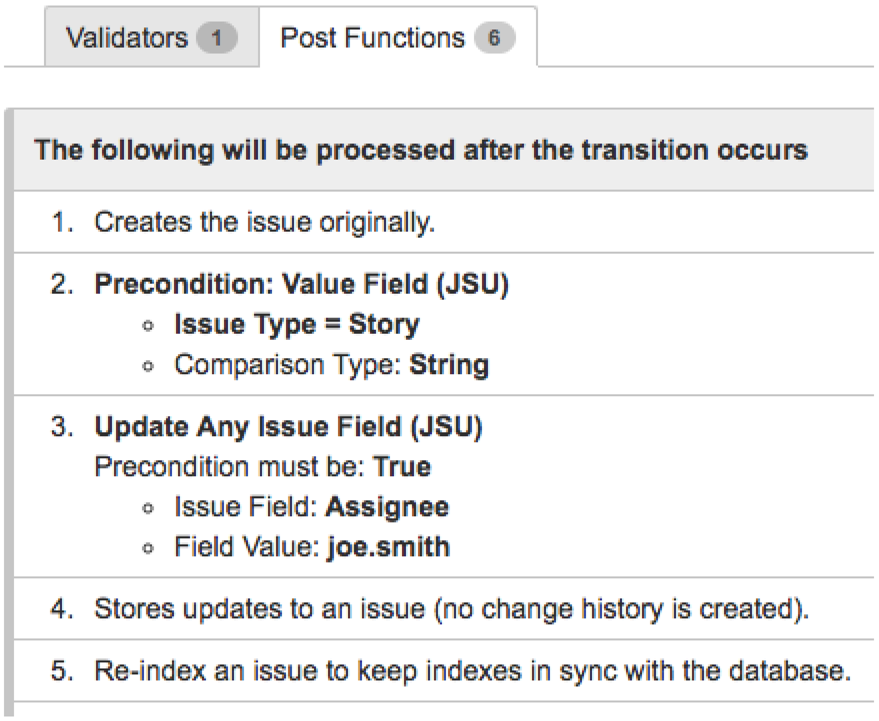
It is important to place the post function in the correct order of other post functions.

### Create Transition

ℹ The 'Create' transition is the very first transition, which does not yet has a source status (only destination status - usually Open, but could also be another).

Instead of using the "Update any Issue Field" post function in the Create transition, you might consider to just configure a default value for that field.

If you are using the "Update any Issue Field" post function in the Create transition, you must put it after the "Creates the issue originally." but before the "Re-index an issue to keep indexes in sync with the database." post function. Depending on the field type, you also need to add the "Store Issue" post function after the "Update Any Issue Field".


Validators 1 | Post Functions 6

**The following will be processed after the transition occurs**

1. Creates the issue originally.
2. **Precondition: Value Field (JSU)**
   - **Issue Type = Story**
   - Comparison Type: **String**
3. **Update Any Issue Field (JSU)**
   Precondition must be: **True**
   - Issue Field: **Assignee**
   - Field Value: **joe.smith**
4. Stores updates to an issue (no change history is created).
5. Re-index an issue to keep indexes in sync with the database.

Note that at step 4 the "Store Issue" post function is needed, which you have to add manually in the workflow configuration.

In our sample we also use a Workflow Precondition before the "Update any Issue Field" post function. This just makes it a more interesting 'real-life' example.

### Any other Transition (not Create)

Put the "Update any Issue Field" post function anywhere before the "Update change history for an issue and store the issue in the database." post function

## Example

See the above example of a Story changing the Epic Status of its Epic.

Or whenever a parent issue is set to 'In Progress' the assignee of all its sub-tasks could be set to the current user (the one changing the status on the parent).

Another example:
A developer has fixed a Bug. He proceeds in the Jira workflow to the status 'Resolved' (this might be triggered from his code pushed to Bitbucket). The 'Update any Issue Field' now adds a label 'testing-required'.
This would be a very light weight solution. There are also cases when you need a more complex solution. Have a look at Testing and Fixing Bugs.

For more information on how to configure a post-function in JIRA, see the JIRA documentation.

## Supported Field Types

In its different modules (especially those for workflows), the JSU app supports many different field types. System fields, as well as custom fields.

However you should be aware, that not all field types are supported. Also not in all combinations. We think we cover the most important field types and still are continuously adding and improving which and how different field types are supported. But the one you need, might just not (yet) work. Some custom fields of other third party app might never get supported.

For that reason you should always test anything you do with the JSU app with fields. Before you buy a license for JSU, try it with a free evaluation license, if it works for you.