Update field based on rules

This very powerful post function can be used to **update** a **single issue field** according to a set of **rules**.

You can use this function to construct multiple **"if then" rules.**

Once you have defined the **source field**, a **rule** and a **value** you have constructed the entire **if clause**.

The next step is to define, which **value** the **target field** will be updated to, if the values match. This is the **then** part of your rule.

If used in the create transition, they need to be placed **after** the “Create issue initially” post function

Configuration

Source field

Select the field that will be analyzed by the defined rules. The combination of this **source field** and the **source value**, as specified in each rule, builds the **if** part.

Even though this parameter is **mandatory**, the selection is **irrelevant** if you choose to base your rule on a **Logical expression**.

Target field

Select the field to be updated according to the rules.

The rules will be processed in order. The **first** rule **matching** the given **expression** will update the target field.

Rules

Define the **rules** for updating the **target field** and the corresponding **target field values**.

Once a rule is **matched**, its associated value will be parsed or calculated and copied to selected target field, and the rest of the rules will **not** be **processed**.

If the selected target field is of type number, date, date and time, the associated value should be a number or a mathematical/time formula. Learn more about Numbers and working with Dates, times and time zones.

Other data types like users, issue status, issue priority and issue resolution require values of corresponding suitable types.

Rules can be written as line separated text in **Expert mode** or using the easy way in **Table view.**

Table view

You can choose between the following options:

Option	Description
Text comparison	Choose this option if you want to compare the **value** of the **specified source field** with another text value.

Logical expression	Choose this option if you want to construct a logical rule using the Logical mode. Once set up, you have to define, which **value** the **target field** will be updated to if the expression returns **`true`**.
Regular expression	Enter a valid regular expression and optionally use field codes to to specify your rule. This expression will then be evaluated against the selected **source field**.

Examples:

IF			THEN		Output
Source field	**Mode**	**Source field value**	**Target field**	**Target field value**	**Description**
Priority	Text comparison	High	Assignee	a.agrant	If the issue **priority** is **High**, the issue will be assigned to **a.grant.**
Priority	Text comparison	Low	Assignee	d.jones	If the issue **priority** is **Low**, the issue will be assigned to **d.jones.**

Expert mode

You can write three types of rules as in the following options:

Option	Description
Text comparison	Add the prefix **<l>** if you want to compare the **value** of the **specified source field** with another text value. The text value should be put in round brackets followed by the target value. Example: `l(<comparableText)targetValue`
Logical expression	Add the check expression in square brackets if you want to construct a logical rule using the Logical mode. It should be followed then with the **value** the **target field** will be updated to if the expression returns **`true`**. `[<logical expression>]targetValue`
Regular expression	Enter a valid regular expression in brackets and optionally use field codes to specify your rule. This expression will then be evaluated against the selected **source field**. If you want to ignore case, then add the prefix **<i>**. `<optional>i(<regEx>)targetValue`

The prefix **<a>** is used to indicate advanced parsing mode for the target value in the rule.

Examples:

Refer to the old version of the postfunction for more examples.

Logical expressions are not analyzing the source field. Whereas text comparison and regular expression rules are checked against the **source field value**, logical expressions are **independent** of the **source field value**. Instead you will have to provide a logical expression using the Logical mode.

Alternatively you could use the Update or copy field values post function and set values using multiple conditional operators.

Additional options

Option	Description
Write protection	Check the **write protection** option if you want to prevent a field to be updated, if it already has a value. If checked, only empty fields will be set.
Evaluate all rules	Per default, JWT will **stop evaluating** the rules when the **first** rule matches (e.g. a source field value was identified according to a rule). If you check this option, JWT will evaluate **all** rules, potentially executing a different rule. The order in which the rules are specified matters in this case as JWT will go through each rule from **top to bottom**. Note that it is necessary to **add a plus "+" sign** at the beginning of every expression in order **not to overwrite the previous values** if a rule is matched. In that manner, the value of every rule will be added to the existing values. Only applicable for multi-valued or temporary target fields.
Update issue immediately	Choose to **update issues immediately** if you want Jira to fire a separate "Issue updated" event for any field change. This will also result in a dedicated **issue history** entry.

Conditional execution

You can **optionally** specify a logical expression to define the circumstances (or conditions) under which the post function should be executed.

The result of the logical expression must return a boolean value of either:

- `true` the post function will be executed
- `false` the post function will **not** be executed

Using the **conditional operator,** even complex or multi-layered conditions can be constructed.

Make sure to learn more about defining logical expressions and browse through the various **examples** here: Logical mode

Run as

Select which **user** will be used to execute this post function. By default this parameter is set to the **current user**. You can also use field codes to run the function as a dynamic user (e.g. current assignee).

Make sure that the user running the post function has all the **relevant permissions** to perform the actions defined in the configuration (e.g. "Update Issues")!

If you want to keep track the actions being performed automatically, we suggest to create a **dedicated JWT account**, granted all relevant **permissions**, and use it in the Run as parameter to identify which changes have been made with JWT.

Use case and examples

Use case	JWT feature	Workflow function	Parser functions	Label

Set reporter as assignee if a User Picker field is empty		Update field based on rules		
Add user to field depending on selected options		Update field based on rules		
Set assignee depending on issue type		Update field based on rules	issueType()	
Set issue security level depending on reporter		Update field based on rules	issueSecurityLevel()	STAFF PICK
Set assignee based on priority		Update field based on rules	priority()	

If you still have questions, feel free to refer to our support team.