Logical validator

The logical validator is one of the **most powerful** and **versatile** validators that can be used in JWT since it uses the **full potential** of the JWT expression parser functions.

This validator evaluates a logical expression that returns `true` or `false`. The transition will only be possible, if the result is `true`.

Configuration

Expression

Since you only have a single parameter, an **expression**, you need to familiarize yourself with the Logical mode, which explains how to write logical expressions.

Validation options

The validation can be **skipped** under certain circumstances Select **one** or **multiple scenarios** in which you want to **skip** the **validation** to ensure the correct execution of certain operations.

Option	Description
Skip validation for **JWT post functions**	If the transition is triggered by a **JWT post function**, e.g. through Transition issue, the validation will be skipped.
Skip validation for **bulk operations**	If the transition is triggered by a **bulk operation**, the validation will be skipped.
Skip validation for **clone operations**	If the create transition is triggered by a **clone operation**, the validation will be skipped.
Skip validation for **mail handlers**	If the create transition is triggered by a **mail handler**, the validation will be skipped.

Error message

You can **optionally** define a custom error message in case the validator **fails**, which will be shown to the user trying to execute the transition.

You have the following options:

Location

Define **where** the error message should be displayed. By default, the message will be shown at the **top** of the transition screen, or in case there is no transition screen, as a **popup**.

Alternatively, you can locate the message below any other field. This option **only makes sense**, if there is a **transition screen** defined for the specific transition.

Due to **limitations** in **Jira Service Management**, the location parameter will be **ignored** on Jira Service Management related screens. The location parameter **only** works for the **Create Issue** transition screen if **JSD version 4.10** or higher is being used.

Message

Define the content of the error message in Basic text mode or Advanced text mode.

Learn more about the possibilities of the JWT expression editor.

In comparison to e.g. our calculated custom fields it is **not possible to display HTML** here. Displaying links to specific issues, for example, is not possible inside the error message.

Translations

After clicking on the **Add translation** button you can **optionally** translate the error message to other languages.

The language in which the error message will be displayed depends on the **language setting** of the individual **users**.

If you want to use this functionality in a **condition** instead, have a look at the Logical condition.

Use cases and examples

Use case	JWT feature	Workflow function	Parser functions	Label
Prevent a transition depending of the number of components	🔍 🔓	Logical validator Logical condition	numberOfSelectedItems()	
Prevent a closed issue from being reopened after resting 1 week closed	🔍 🔓	Logical validator Logical condition	isInRole()	
Prevent issue creation if description contains less than 100 characters	🔓	Logical validator	replaceAll() length()	
Evaluate Assets objects	🔍 🔓	Logical validator Logical condition	findPattern() findReplaceAll() replace All() toStringList()	

Block a transition until all sub-tasks have certain fields populated	🔍 🔓	Logical validator
		Logical condition
			count()
			filterByPredicate()
			subtasks()
				STAFF PICK

Validate a Select List (cascading) custom field	🔍	Logical validator Logical condition		
Prevent external users from creating issues	🔍	Logical validator	matches() toString() textOnStringList() toStringList() userDisplayName()	

Validate only issue links created in transition screen	🔍	Logical validator	count()
			transitionLinkedIssues()
			filterByProject()
			filterByIssueType()	

Hide transition to a previous status	🔓	Logical condition		
Prevent setting due dates outside business hours	🔍	Logical validator Logical condition	withinCalendar()	STAFF PICK
Set a condition in a global transition which only applies in a certain status	🔓	Logical condition		
Prevent the creation of sub-tasks unless the parent issue is in a certain status	🔓	Logical validator		
Halt a transition if an element is not contained in a list	🔍 🔓	Logical validator Logical condition	toStringList()	

Validation based on the value of a date type project property	🔍	Logical validator
		Logical condition
			stringToDate()
			projectProperty()	

Restrict issue creation per issue type and project role	🔍	Logical validator	isInRole()	

Reject duplicated file names in attachments	🔍 🔓	Logical validator
		Logical condition
			count()
			toStringList()
			distinct()	

Prevent issue creation with the same field value (Boolean)	🔍	Logical validator
		Validation based on JQL query
			count()
			issuesFromJQL()	

Ensure that all issues linked with a certain issue link type have "Due Date" field set	🔍 🔓	Logical validator
		Logical condition
			count()
			linkedIssues()
			fieldValue()	

Block an Epic's transition until all the issues under that Epic are resolved	⚙	Logical validator
		Logical condition
			count()
			issuesUnderEpic()
			filterByResolution()	

Block an Epic's transition depending on linked issues status and due date	🔍 🔓	Logical validator
		Logical condition
			count()
			filterByPredicate()
			linkedIssues()	

Make "Time spent" field required	🔍	Logical validator		
Ensure that an issue has at least one attachment	🔍	Logical validator Logical condition		STAFF PICK

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