Migrate 'Condition based on cascading select list value'

The Condition based on cascading select list value of JWT DC is not yet available in JWT Cloud but can be implemented easily using a Jira expression condition .

Since JWT DC field codes are not available in conditions or validators in JWT Cloud, they have to be replaced by their Jira expression equivalent.

Migration steps

1 Add a Jira expression condition.

2 Build a matching Jira expression by migrating the parameters of the Condition based on cascading select list value

The following table shows how to obtain the individual components of the resulting Jira expression and describes how to assemble these parts into a Jira expression.

Migration details

JWT DC	JWT DC option	JWT Cloud	Notes
Field		Select the corresponding cascading select field from the Field injector. Example `issue?.customfield_10044`	The list of available field codes differs between JWT DC and JWT Cloud.
Option level	Parent	Add `?.value` to the field code selected in the **Field** parameter. Example `issue?.customfield_10044?.value`	
	Child	Add `?child?.value` to the field code selected in the **Field** parameter. Example `issue?.customfield_10044?.child?.value`	
Comparison operator	is equal	Add == to the expression built so far. Example `issue?.customfield_10044?.child?.value =`	

	isn't equal	Add `!=` to the expression built so far. Example `issue?.customfield_10044?.child?.value !=`	
	contains	Add `.includes()` to the expression built so far. Example `issue?.customfield_10044?.child?.value.includes()`	
	doesn't contain	Add `.includes()` to the expression built so far and put the negation operator ! in the beginning of the expression. Example `!issue?.customfield_10044?.child?.value.includes()`	
	starts with	Add `. indexOf() == 0` to the expression built so far. Example `issue?.customfield_10044?.child?.value.indexOf() == 0`	
	doesn't start with	Add `. indexOf() != 0` to the expression built so far. Example `issue?.customfield_10044?.child?.value.indexOf() != 0`	
	ends with	Add `.match("$") != null` to the expression built so far. Example `issue?.customfield_10044?.child?.value.match("$") != null`	
	doesn't end with	Add `.match("$") == null` to the expression built so far. Example `issue?.customfield_10044?.child?.value.match("$") == null`	

Comparison value		If field codes are used within the comparison value, they have to be replaced with the corresponding Jira expression field codes (using the field code injector) and concatenated with the remaining text using "+". All other texts have to be quoted (enclosed by ""). This value is then put either as right-hand operator or as parameter in the ()-part of the current expression (depending on the **comparison operator** which is used - in case of "ends with"/"doesn't end with" right before the "$"). Examples (see table below)	

Comparison operator	Comparison value	Jira expression
starts with	"label"	`issue?.customfield_10044?.value.indexOf("label") == 0`
contains	First %{issue.description}	`issue?.customfield_10044? .value .includes("First"+issue.description.plaintext)`
doesn't equal	3	`issue?.customfield_10044?.value != "3"`

Examples

JWT DC parameter values	Jira expression
Parameter: Field = %{issue.cf10003}; Option level = Parent; Comparison operator = =; Comparison value = %{issue.summary}	`issue?.customfield_10003?.value == issue.summary`
Parameter: Field = %{issue.cf10042}; Option level = Child; Comparison operator = starts with; Comparison value = First	`issue?.customfield_10042?.child?value.indexOf("First") == 0`
Parameter: Field = %{issue.cf10042}; Option level = Child; Comparison operator = doesn't contain; Comparison value = Child value of %{issue.key}	`!issue?.customfield_10042?.child?value.includes("Child value of "+issue.key)`

Parameter	Value
Field	%{issue.cf10003}
Option level	Parent
Comparison operator	=
Comparison value	%{issue.summary}

```
issue?.customfield_10003?.value == issue.summary
```

Parameter	Value
Field	%{issue.cf10042}
Option level	Child
Comparison operator	starts with
Comparison value	First

```
issue?.customfield_10042?.child?value.indexOf("First") == 0
```

Parameter	Value
Field	%{issue.cf10042}
Option level	Child
Comparison operator	doesn't contain
Comparison value	Child value of %{issue.key}

```
!issue?.customfield_10042?.child?value.includes("Child value of 
"+issue.key)
```

Parameter	Value
Field	%{issue.cf10042}
Option level	Child
Comparison operator	doesn't end with
Comparison value	%{issue.key} value

```
issue?.customfield_10044?.child?.value.match(issue.key+" value$") ==
null
```

Due to the different architecture, it may happen that the condition gets too complex. This is the case when many fields are checked. The condition cannot be saved, and a corresponding error message will be displayed. If that's the case, the condition has to be split up into two or more.

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