


Migrate logical expressions

Logical expressions in JWT Cloud are built the same as in JWT DC. In addition to the basic differences in [field codes](#) and [expression parser functions](#), there are a few other minor differences that are explained on this page.

Multi-valued fields

In JWT DC so-called multi-valued fields are treated as lists (and not as simple text) when using the following logical operators :

 JWT DC operator	Meaning
~	contains
!~	does not contain
in	is contained in
not in	is not contained in
any in	any element is in
none in	no single element is in
~~	contains (case ignoring)
!~~	does not contain (case ignoring)
in~	is contained in (case ignoring)
not in~	is not contained in (case ignoring)
any in~	any element is in (case ignoring)
none in~	no single element is in (case ignoring)

That means that in JWT DC it is possible to write expressions like

```
%{issue.components} any in %{parent.components}
```

In JWT Cloud, this notion is **not** available. An expression dealing with such a multi-valued field has to be migrated by explicitly converting the field code value using the function [toStringList](#) (or [toNumberList](#), respectively).

The following fields are considered multi-valued fields:

- [Affects version/s](#)
- [Attachments](#)
- [Available target statuses](#)
- [Available transitions](#)
- [Component/s](#)
- [Component/s leads](#)
- [Fix version/s](#)
- [Labels](#)
- [Keys of linked issues](#)
- [Keys of sub-tasks](#)
- [Watchers](#)
- Custom fields
 - Select List (Multiple choice)
 - Checkboxes
 - Labels
 - User picker (multiple users)
 - Group picker (multiple groups)
 - Version picker (multiple versions)


In order to migrate multi-valued fields in logical expressions, apply the [toStringList\(\)](#) to the respective field.

Examples

 JWT DC expression	 JWT Cloud expression
<code>{issue.versions} none in {issue.fixVersions}</code>	<code>toStringList({issue.versions}) none in toStringList({issue.fixVersions})</code>
<code>{issue.labels} any in ["Backend", "Frontend"]</code>	<code>toStringList({issue.labels}) any in ["Backend", "Frontend"]</code>



Case ignoring operators

In JWT DC, some case-ignoring operators are available (for details please see the respective part in the description of JWT DC's [Logical mode](#)). They are **not** available in JWT Cloud.

 JWT DC operator	Meaning
<code>=~</code>	equal to
<code>!=~</code>	not equal to
<code>~~</code>	contains
<code>!~~</code>	does not contain
<code>in~</code>	is contained in
<code>not in~</code>	is not contained in
<code>any in~</code>	any element is in
<code>none in~</code>	no single element is in

As a work around you can use the functions [toUpperCase\(\)](#) / [toLowerCase\(\)](#) for both operands and use the respective case sensitive operator. If an operand is a list, it first has to be converted to a text first using [toString\(\)](#).

Examples

 JWT DC expression	 JWT Cloud expression
<code>"HELLO" =~ "Hello"</code>	<code>toUpperCase("HELLO") = toUpperCase("Hello")</code>

```
%{issue.labels} ~~  
[ "Backend", "Database" ]
```

```
toListStringList(toLowerCase(%issue.labels})) ~ toListStringList(toLowerCase  
(toString([ "Backend", "Database" ])))
```

In this case, we are dealing with a multi-valued field and thus have to convert it to a text list using [toListStringList\(\)](#) before comparing.

If you still have questions, feel free to refer to our [support](#) team.