Operators

General Information

The expression parser accepts the most common operators. The operators listed below are available for the following **data types**:

- **Numbers**
- **Strings**
- **Issue lists**
- **Number lists**
- **String lists**

Operators = and != are also available for type BOOLEAN

Case-sensitive operators

Operator	Meaning	Examples (all examples return `true`)

`=`	equal to	`1 = 1`
		`"HELLO" = toUpperCase("Hello")`
		`%{...description} = {...timeoriginalestimate}` , auto-casting numeric field **{...originalEstimate}** to Text-String.
		`%{...originalEstimate} = toString({... originalEstimate})` , explicit casting of numeric field **{... originalEstimate}** to Text-String.
		`true = true`
		`%{...cf10001} = null` , for checking whether field with code **%{... cf10001}** is not initialized.
		`[1, 2, 3] = [1, 2, 3]` , when used with lists elements existence and its order are evaluated.
		`["blue", "red", "green"] = ["blue", "red", "green"]`

`!=`	not equal to	`0 != 1`
		`"HELLO" != "Hello"`
		`%{...description} != "Hello"`
		`true != false`
		`{...cf10010} != null` , for checking whether the numeric field with code **{...cf10010}** is initialized.
		`[1, 2, 3] != [1, 3, 2]` , when used with lists elements existence and its order are evaluated.
		`["blue", "red", "green"] != ["blue", "green", "red"]`

`<`	lower than	`1 < 2`
		`"abc" < "bbc"`
		`"abc" < "abcd"`

`>`	greater than	`2 > 1`
		`"bbc" > "abc"`
		`"abcd" > "abc"`

`<=`	less than or equal to	-
`>=`	greater than or equal to	-

`~`	contains	`"Hello world!" ~ "world"` , checks whether a string contains a substring.
		`%{...componentLeads} ~ %{...currentUser}` , checks whether "**Component leaders**" contains "**Current user**".
		`linkedIssues() ~ subtasks()` , checks whether all sub-tasks are also linked to current issue.
		`[1, 2, 3, 2, 2, 4] ~ [2, 1, 2]` , when used with lists cardinalities must match.
		`["blue", "red", "green", "red", "white", "red"] ~ ["red", "green", "red"]`
		`(["green", "red"] ~ ["red", "green", "red"]) = false`

`!~`	doesn't contain	`"world" !~ "Hello world!"`
		`%{...fixVersions} !~ %{...versions}` , checks whether "**Fix version/s**" doesn't contain all versions in "**Affects version/s**".
		`fieldValue(%{...reporter}, linkedIssues()) !~ fieldValue(%{...reporter}, subtasks())` , checks whether linked issues reporters don't include all sub-tasks reporters.
		`[1, 2, 3, 2, 2, 4] !~ [2, 1, 1, 4]` , when used with lists cardinalities must match.
		`["blue", "red", "green", "red", "red"] !~ ["red", "green", "green", "red"]`

`in`	is contained in	`"world" in "Hello world!"` , to check whether a substring is contained in a string.
		`%{...currentUser} in %{...componentLeads}` , checks whether "**Current user**" is contained in "**Component leaders**".
		`subtasks() in linkedIssues()` , checks whether all sub-tasks are also linked to current issue.
		`[1, 1, 2] in [2, 1, 1, 1, 4]` , cardinality must match.
		`["blue", "red", "red"] in ["red", "green", "blue", "red", "red"]` , cardinality must match.
		`2 in [1, 2, 3]`
		`"blue" in ["red, "blue", "white"]`

`not in`	isn't contained in	`"Hello world!" not in "world"`
		`%{...versions} not in %{...fixVersions}` , checks whether not all versions in "**Affects version/s**" are contained in "**Fix version/s**".
		`fieldValue(%{...reporter}, subtasks()) not in fieldValue(%{...reporter}, linkedIssues())` , checks whether not all sub-tasks reporters are included in linked issues reporters.
		`[1, 1, 2, 2] not in [2, 1, 1, 1, 4]` , cardinality must match.
		`["blue", "red", "red", "blue"] not in ["red", "blue", "red", "red"]` , cardinality must match.
		`5 not in [1, 2, 3, 3, 4]`
		`"orange" not in ["blue", "red", "white"]`

`any in`	some element is in	`%{...versions} any in %{...fixVersions}` , checks whether any version in "**Affects version/s**" is contained in "**Fix version/s**".
		`fieldValue(%{...reporter}, subtasks()) any in fieldValue(%{...reporter}, linkedIssues())` , checks whether any sub-task's reporter is present among linked issues reporters.
		`[1, 3] any in [3, 4, 5]`
		`["blue", "white"] any in ["black", "white", "green"]`

`none in`	no single element is in	`%{...versions} none in %{...fixVersions}` , checks whether there isn't a single version "**Affects version/s**" in "**Fix version/s**".
		`fieldValue(%{...reporter}, subtasks()) none in fieldValue(%{...reporter}, linkedIssues())` , checks whether there isn't a single sub-task reporter among linked issues reporters.
		`[1, 2] none in [3, 4, 5]`
		`["blue", "red"] none in ["black", "white", "green"]`

Case-ignoring Operators

The following comparison operators are applicable to STRING and STRING [] **data types**.

All operators ignore the case of the characters.

Operator	Meaning	Examples (all examples return `true`)

`=~`	equal to	`"HELLO" =~ "Hello"`
		`"up" =~ "UP"`
		`["blue", "red", "green"] =~ ["Blue", "RED", "Green"]`

`!=~`	not equal to	`" HELLO" !=~ "Hello"`
		`"up" !=~ "down"`
		`("up" !=~ "UP") = false`
		`["blue", "red"] !=~ ["Blue", "green"]`
		`["blue", "red"] !=~ ["Red", "BLUE"]`
		`(["blue", "red", "green"] !=~ ["Blue", "RED", "Green"]) = false`

`~~`	contains	**`"Hello World!" ~~ "world"`** , checks whether a string contains a substring. **`"A small step for a man" ~~ "STEP"`** , checks whether a string contains a substring. **`["one", "two", "three"] ~~ ["TWO", "One"]`** , checks whether a string list contains all the elements of another string list.
`!~~`	doesn't contain	**`"Hello World!" !~~ "bye"`** , checks whether a string doesn't contain a substring. **`"A small step for a man" !~~ "big"`** , checks whether a string doesn't contain a substring. **`["one", "two", "three"] !~~ ["Four"]`** , checks whether a string list doesn't contain one element of another string list. **`(["one", "two", "three"] !~~ ["TWO"]) = false`**
`in~`	is contained in	**`"world" in~ "Hello World!"`** , checks whether a substring is contained in another string. **`"STEP" in~ "A small step for a man"`** , checks whether a substring is contained in another string. **`["TWO", "One"] in~ ["one", "two", "three"]`** , checks whether all the elements of a string list are contained in another string list.
`not in~`	isn't contained in	**`"bye" not in~ "Hello World!"`** , checks whether a substring is not contained in another string. **`"big" not in~ "A small step for a man"`** , checks whether a substring is not contained in another string. **`["Four"] not in~ ["one", "two", "three"]`** , checks whether any of the elements of a string list are not contained in another string list. **`(["TWO"] not in~ ["one", "two", "three"]) = false`**
`any in~`	some element is in	**`["blue", "violet"] any in~ ["Blue", "Red", "Green"]`** **`["Five", "One"] any in~ ["FOUR", "FIVE", "SIX"]`**
`none in~`	no single element is in	**`["Orange"] any in~ ["red", "blue", "green"]`** **`(["orange"] any in~ ["Red", "Orange"]) = false`**

Operators and applicable data types

Below you find a comprehensive matrix of all operators and applicable data types.

Comparison Operator	BOOLEAN	NUMBER	STRING	NUMBER []	STRING []	ISSUE
`=`	X	X	X	X	X	X
`!=`	X	X	X	X	X	X
`<`	-	X	X	-	-	-
`>`	-	X	X	-	-	-
`<=`	-	X	X	-	-	-
`>=`	-	X	X	-	-	-
`~`	-	-	X	X	X	X
`!~`	-	-	X	X	X	X
`in`	-	-	X	X	X	X
`not in`	-	-	X	X	X	X
`any in`	-	-	-	X	X	X
`none in`	-	-	-	X	X	X
`=~`	-	-	X	-	X	-
`!=~`	-	-	X	-	X	-
`~~`	-	-	X	-	X	-
`!~~`	-	-	X	-	X	-
`in~`	-	-	X	-	X	-
`not in~`	-	-	X	-	X	-
`any in~`	-	-	-	-	X	-
`none in~`	-	-	-	-	X	-

Remember	Example
Operators **`~, !~, in`** and **`not in`** can be used for checking a single element **(number or string)** against a **number list** or a **string list**	• **`1 in [1, 2, 3]`** • **`["blue", "red"] ~ "blue" .`**
Operators **`~, !~, in`** and **`not in`** when used with a **string** a re useful to look for substrings in another string.	• **`"I love coding" ~ "love"`** • **`"I don't like Mondays" !~ "Fridays"`** • **`"love" in "I love coding"`** • **`"Fridays" not in "I don't like Mondays".`**
Operators **`~, !~, in`** and **`not in`** respect cardinality, i.e., container list must have at least the same number of elements as contained list.	• **`[1, 1] in [1, 1, 1]`** • **`[1, 1] not in [1, 2, 3] .`**
Operators **`=`** and **`!=`** , when used for comparing lists, require to have the **same elements**, with the **same cardinality** and the **same order**.	• **`[1, 2, 3] = [1, 2, 3]`** • **`[4, 5, 6] != [4, 6, 5] .`**
Operators **`<, >, <=`** and **`>=`** work according to lexicographical order when comparing strings.	

A reference of all data types can be found here.