Issue list mode

The **issue list** parsing mode, just like the JQL mode, lets you define a target **selection** of **issues**. It is being used in Post functions and Automation actions, that allow you to **create** or **update multiple issues**.

The output has to be a **valid issue list** as defined in the Data types section.

All JWT expression parser functions, that return an issue list can be used. Additionally, issue lists can be constructed using functions like issueKeysToIssueList().

Example expressions

Parser expression	Description
`subtasks()`	This example returns a **list** of all **sub-tasks** of the current issue. To achieve this, the following functions are used: • subtasks()
`issueKeysToIssueList("CRM-12, HT-254")`	This example returns an issue list of issues with keys **CRM-12** and **HT-254**: **["CRM-12", "HT-254"]** To achieve this, the following functions are used: • issueKeysToIssueList()
`filterByFieldValue(subtasks(), {issue.dueDate}, =, null)`	This example returns an issue list with all **sub-tasks** that don't have a **due date** set. To achieve this, the following functions are used: • filterByFieldValue() • subtasks()
`getIssuesFromProjects("CRM, HT")`	This example returns an **issue list** of all issues in project **CRM** and **HT**. To achieve this, the following functions are used: • getIssuesFromProjects()

Make sure to read all about working with Lists as they come with many extremely useful JWT expression parser functions.

List functions can also be used in the Advanced text mode. The difference is, that the output will be a **flat text** instead of an **issue list**.

The main advantage over the advanced text mode is, that you can use the returned **elements** of the issue list as **Seeds**.

JWT offers individual operators that can be used when working with Lists.

Available operators

Function	Short description	Output
APPEND	**Combines** the elements of two **lists**.	LIST
UNION	Returns **distinct elements** of two lists.	LIST
INTERSECT	Returns **common elements** of two lists.	LIST
EXCEPT	**Removes** certain elements from a list.	LIST

Order of operations

If you use multiple operators in a single expression, they will follow a certain order in which they are processed or a precedence.

OPERATORS	PRECEDENCE	ASSOCIATIVITY
INTERSECT	1 (highest)	Left-to-right
APPEND, EXCEPT, UNION	2 (lowest)	Left-to-right

- When using the list operators, you have to make sure that both lists that you compare are of the **same type**.
- All operators are **case insensitive**, i.e., they can also be written in lower case: **append, union, `intersect`** and **`except`**.
- There are **four equivalent functions** available for each type of list, and their behavior is **exactly equivalent** to that of its corresponding operator.
  - append()
  - except()
  - intersect()
  - union()
- This way, you can choose to use **operators** or **functions** according to your preference. Although operators yield shorter expressions and with fewer parentheses, the usage of functions produces a more functional consistent syntax.

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