Text list mode

The **text list** parsing mode is exclusively being used in the Create issue post function and enables you to create issues based on multi-value custom fields, components and many more.

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

All JWT expression parser functions, that return a text list can be used. Additionally, text lists can be composed using the toStringList() function.

Example expressions

Parser expression	Description
`["Jira","Workflow","Toolbox"]`	This examples illustrates how to compose a **custom text list**.
`fieldValue(%{issue.assignee}, subtasks())`	This examples returns a list of all **assignees** of the current **issues's sub-tasks**. The list may contain duplicate user names. To achieve this, the following functions were used: • fieldValue() • subtasks()
`distinct(fieldValue(%{issue.assignee}, subtasks()))`	This examples returns a list of all **distinct assignees** of the current **issues's sub-tasks**. The list only contains **unique** user names. To achieve this, the following functions were used: • distinct() • fieldValue() • subtasks()

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 a **text list**.

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.