Lists

On this page

JWT is capable of processing various types of lists, including **text**, **number** and **issue lists**. This page contains valuable information about working with lists.

The **list** **data type** is an ordered list of elements. Those elements have a certain data type (text or number).

It is possible to:

- access individual elements (e.g. using the function nthElement()),
- create lists out of virtual fields (e.g. using toStringList()) or
- use the list functions presented on this page to work with lists.

Fixed values

A **list** can be written in literal as a **comma separated list of texts in double quotes written inside brackets.**

Expression	Description
`[] #All lists are enclosed in brackets.`	An **empty** list.
`["Blue", "Green", "Yellow", "Orange", "Red"]`	A **text** list with five elements.
`[1,2,3]`	A **number** list with three elements.
`[1, {issue.subtasks.count}, {issue.links.count}]`	A **number** list with three elements using field codes and JWT expression parser functions. To achieve this, the following virtual field codes: • Number of sub-tasks • Number of linked issues
`toStringList(%{issue.components})`	A text list with all **components** of the **current issue**. Fields codes of multi-valued fields, such as components, labels, etc. by default return a text with a **comma separated list of values**. If you want to individually process these elements, you need to convert the text into a text list using toStringList(). Learn more about Converting data types.

Reading values from issue lists

Most of the times, lists are being defined or returned using field codes or JWT expression parser functions. In order to read or retrieve values from an issue list, functions such as fieldValue() must be used.

Expression	Description
`fieldValue(%{issue.assignee}, subtasks())`	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()))`	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()
`fieldValue(%{issue.priority}, linkedIssues ("blocks, is cloned by"))`	A list of priorities of those issues linked to current issue through issue link types "**blocks**" and "**is cloned by**". To achieve this, the following functions were used: • fieldValue() • linkedIssues()
`distinct(fieldValue(%{issue.components}, issuesUnderEpic()))`	A list with **distinct components** of all issues which are **linked** to **same** epic **a s the current issue**. To achieve this, the following functions were used: • distinct() • fieldValue() • issuesUnderEpic()

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.

List functions for multiple list types

The following list contains all the available functions that work with all kinds of lists; text, number and issue lists.

Function	Short description	Output	Label
append()	**Combines** the elements of two **lists.**	LIST	
count()	Returns the **number of elements** in a text, number or issue list.	NUMBER	STAFF PICK
distinct()	**Removes all duplicates** from a number, text, or issue list.	LIST	STAFF PICK
except()	**Removes** certain elements from a list.	LIST	
filterByCardinality()	Filters a given number, text, or issue list by the **number** of **occurence** of **elements**.	LIST	
filterByPredicate()	Filters a number, issue, or a text list by a given **logical expression**.	LIST	STAFF PICK
filterByValue()	**Filters** a number or text list using a **given comparison**.	LIST	STAFF PICK
first()	Returns the first element of a **number**, **text**, or **issue list**.	NUMBER TEXT ISSUE LIST	STAFF PICK
getMatchingValue()	Returns a custom reference value for a given or **text** or **number**.	NUMBER TEXT	
getRemoteLinks()	Returns a list of remote links.	TEXT LIST	
indexOf()	Returns the **index / position** of a **specific element** in a list.	NUMBER	
intersect()	Returns **common elements** of two lists.	LIST	
invertList()	Inverts the **order** of a given list.	LIST	
last()	Returns the last element of a **number**, **text**, or **issue list**.	NUMBER TEXT ISSUE LIST	

mathOnStringList()	Returns a **number list** with results of the given calculation performed for each text in the specified list.	NUMBER LIST
nthElement()	Returns the **nth element** of a number, string or issue list.	NUMBER TEXT ISSUE LIST
sort()	**Sorts** a given list in a specific order.	LIST
sublist()	Returns a **defined extract** of a given **list**.	LIST
textOnNumberList()	Returns a **text list** in a result of evaluating **text_expression** against each of the numeric values in argument **numbers**.	TEXT LIST
textOnNumberList()		
textOnStringList()	Matches elements of a text list against a **text expression**.	TEXT LIST
union()	Returns **distinct elements** of two lists.	LIST

Number list (only) functions

The following list contains all the available functions that work with **number lists only**.

Function	Short description	Output	Label
avg()	Calculates the **average** values of a given number list.	NUMBER	
mathOnNumberList()	Returns a **number list** with results of the given calculation performed for each number in the specified list.	NUMBER LIST	
max(list)	Returns the **highest value** in a number list.	NUMBER	
min(list)	Returns the **smallest value** in a number list.	NUMBER	
sum()	**Sums up** all values in a given number list.	NUMBER	STAFF PICK

Issue list (only) functions

The following list contains all the available functions that work with **issue lists only.**

Function	Short description	Output	Label
allIssuesUnder()	Returns issues located in **any level under** a given parent issue according to **Advanced Roadmaps hierarchy.**	ISSUE LIST	STAFF PICK
epic()	Returns **all epics** linked to specified issues.	ISSUE LIST	
fieldValue()	Returns a number or text list with **field values**.	NUMBER LIST TEXT LIST	STAFF PICK
filterByFieldValue()	**Filters** an **issue list** using a given comparison for field values.	ISSUE LIST	
filterByIssueType()	**Filters** a given issue list by **issue type**.	ISSUE LIST	

filterByProject()	**Filters** a given issue list by **project**.	ISSUE LIST	
filterByProjectCategory()	**Filters** a given issue list by **project category**.	ISSUE LIST	
filterByResolution()	**Filters** a given issue list by **resolution**.	ISSUE LIST	
filterByStatus()	**Filters** a given issue list by **issue status**.	ISSUE LIST	STAFF PICK
filterByStatusCategory()	**Filters** a given issue list by **status category.**	ISSUE LIST	
getIssuesFromProjects()	Returns **all issues** from specified projects.	ISSUE LIST	
issuesAbove()	Returns all issues located in **any level above** a given parent issue according to **Advanced Roadmaps hierarchy.**	ISSUE LIST	STAFF PICK
issuesFromJQL()	Returns a list of issues returned by a specified **JQL query**.	ISSUE LIST	STAFF PICK
issuesUnder()	Returns issues located in the level **just under** a given parent issue according to **Advanced Roadmaps hierarchy.**	ISSUE LIST	STAFF PICK
issuesUnderEpic()	Returns all **issues** linked to a given **epic**	ISSUE LIST	STAFF PICK
linkedIssues()	Returns a **list** of **issues linked**.	ISSUE LIST	STAFF PICK
mathOnIssueList()	Returns a **number list** with results of the given calculation performed for each issue in the specified list.	NUMBER LIST	
numberOfRemoteIssueLinks()	Returns the number of **issue links** to remote Jira instances	NUMBER	
parent()	Returns the **direct parent(s)** of the given issue(s) according to **Advanced Roadmaps hierarchy.**	ISSUE LIST	STAFF PICK
siblingIssues()	Returns all issues which are located **directly under a given issue's parent** according to **Advanced Roadmaps hierarchy.**	ISSUE LIST	STAFF PICK
siblingIssuesUnderEpic()	Returns all **sibling issues** linked to the same **epic**	ISSUE LIST	
siblingSubtasks()	Returns all **sibling sub-tasks**.	ISSUE LIST	
subtasks()	Returns **sub-tasks** of given issues.	ISSUE LIST	STAFF PICK
textOnIssueList()	Returns a **text list** in result of evaluating **textExpression** against each of the issues in argument **issues**.	TEXT LIST	
transitionLinkedIssues()	Returns a list of all **issues linked** during the **transition**.	ISSUE LIST	
transitivelyLinkedIssues()	Returns issues that are **directly** or **transitively linked** to the current issue.	ISSUE LIST	

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