

Examples of String List expressions

This page presents a collection of string list expressions valid for the [Expression parser 201 - All functions](#).

String list is one of the data types that can be returned by an expression in **Automation Toolbox for Jira**.

String List Literals

The syntax of a string list literal is a **comma separated list of strings in double quotes written inside brackets**. Examples:

Expression	Example of returned value	
[]	Empty list.	-
["red", "blue", "green"]	String list with 3 strings.	-
["Ada Lovelace", "John von Neuman", "Alan Turing"]	String list with 3 strings.	-

On this page

- [String List Literals](#)
- [Reading Fields from Issue Lists](#)
- [Reading Multi-valued Fields as String Lists](#)
- [Reading Multi-valued Fields from Issue Lists](#)
- [Parsing a String into a String List](#)

Reading Fields from Issue Lists

When we read field values from issues using field codes for string values (format `%{...nnnnn}`), the returned data type is a string list.

Expression	Example of returned value
<code>fieldValue(%{...assignee}, subtasks())</code>	List of usernames of the assignees of the current issue's subtasks. It may contain repeated usernames.
<code>distinct(fieldValue(%{...assignee}, subtasks()))</code>	List of usernames of the assignees of the current issue's subtasks. It will not contain repeated usernames, because of the usage of function <code>distinct()</code> .
<code>fieldValue(%{...priority}, linkedIssues("blocks, is cloned by"))</code>	List of priorities of those issues linked to current issue through issue link types "blocks" and "is cloned by" .

You will find very useful these examples of [Issue List expressions](#) in order to select the issues you need to read fields from.

Reading Multi-valued Fields as String Lists

Fields codes of multi-valued fields like Affected versions, Fixed versions, Components, Labels, Attachments, Watchers, Request Participants, Multi-Select List, Check List, Multi-User Picker, Multi-Group Picker and Multi-Version Picker custom fields, return a **string** with a **comma separated list of values**. We can convert this string into a string list by means of function `toStringList()`.

Expression	Example of returned value
<code>toStringList(%{...version})</code>	If Fixed versions field code (i.e., <code>%{...version}</code>) returned "1.1, 1.2, 1.3", the expression would return ["1.1", "1.2", "1.3"]
<code>toStringList(%{...components})</code>	String list with the components currently selected in field Components .

Reading Multi-valued Fields from Issue Lists

Expression	Example of returned value
<code>distinct(toStringList(tostring(fieldvalue(%{...components}, subtasks())),""))</code>	Returns a string list with all the components present in the sub-tasks of current issue. The list obtained doesn't contain duplicates, thanks to function <code>distinct()</code> .

On this page

- [String List Literals](#)
- [Reading Fields from Issue Lists](#)
- [Reading Multi-valued Fields as String Lists](#)
- [Reading Multi-valued Fields from Issue Lists](#)
- [Parsing a String into a String List](#)

Parsing a String into a String List

We can use the following functions in order to parse a string into a string list:

Expression	Example of returned value
<code>toStringList(string s, string separators) : string list</code>	INPUT: String with a list of tokens separated by one or more characters. OUTPUT: A string list with tokens in argument s separated by characters in argument separators . Example: <code>toStringList("red, orange, yellow; green; blue; purple", ",;")</code> returns the following string list: ["red", "orange", "yellow", "green", "blue", "purple"].
<code>findPattern(string s, string regexp) : string list</code>	Returns a string list with all substrings in argument s matching regular expression in string argument regexp . Example: <code>findPattern("Between 1900 and 2000 world population increase from 1.5 to 6.1 billions.", "\\d+(\\.\\d+)?")</code> returns ["1900", "2000", "1.5", "6.1"].
<code>findPatternIgnoreCase(string s, string regexp) : string list</code>	Returns a string list with all substrings in argument s matching regular expression in string argument regexp . Evaluation of the regular expression is carried out in ignoring case mode. Example: <code>findPatternIgnoreCase("Grass is Green and Sky is Blue.", "red green blue")</code> returns ["Green", "Blue"].