Texts

On this page

The JWT expression parser is very powerful when it come to composing texts.

This page contains a comprehensive overview of all related information.

Fixed values

- When used inside the function mode (**`%{...}`**), texts need to be written in **double quotes**, e.g., "This is a text"
- Operator **+** is used for **concatenating texts**. e.g., **`"This is" + " a text." = "This is a text."`** .
- The **Escape character** is "\" . This character can precede any of the following characters: "**, \, n, r, t, f** and **b** in order to invoke an alternative interpretation. e.g. **\n** for a linefeed.

If you want to introduce a **double quote** in a text expression you should precede it with escape character **`\`** as in **`"The man said: \" Hello!\"."`**, where we are using the escape character****to write the text **`Hello!`** in double quotes.

Variable values (field values)

Text field values can be inserted into expressions using field codes in the format **%{...somefield}**, or **%{...somefield.i}** for referencing concrete levels in cascading select fields (i = 0 for base level).

For checking if a field has a value you can use **`%{...somefield} = null`** or **`%{...somefield} != null`**.

For a concrete level in a **Cascading Select** or **Multi-Cascading Select** field, you should use **`%{...somefield.i} = null`** or **`%{...somefield.i} != null`**.

Any field type has a text value, so you can also use**%{...somefield}** to insert text values of fields of type **Number** (which includes Date and Date-Time values). Read more about Data types (JWT expressions).

Examples

Input	Output
`%{"Hello" + " " + "world" + "."}`	**Hello world.**
`%{trim(%{issue.summary})}`	**Summary** of an issue **without leading** and **trailing blanks**
`%{issue.description} \nLAST USER: %{toUpperCase(%{system.currentUser})}`	**Description** of an issue and a new line with string "**LAST USER:** " and the **name of current user** in upper case.

Available functions

Function	Short description	Output
findPattern()	Returns all **substrings** matching a given **regular expression**.	TEXT LIST

findPatternIgnoreCase()	Returns all **substrings** matching a given **regular expression**, **ignoring the case**.	TEXT LIST
findReplaceAll()	**Replaces all occurrences** of a given text with the given replacement.	TEXT
findReplaceAllIgnoreCase()	**Replaces all occurrences** of a given text with a given replacement, **ignoring the case**.	TEXT
findReplaceFirst()	**Replaces** the **first occurrence** of a given text with the given replacement.	TEXT
findReplaceFirstIgnoreCase()	**Replaces** the **first occurrence** of a given text with the given replacement, **ignoring the case**.	TEXT
getFromJSON()	Returns the result of a given JMESPath which is applied to a JSON	TEXT
issueType()	Returns the **name** of the **issue type** with a given **ID.**	TEXT
jiraExpression()	Returns the result of a **Jira expression.**	TEXT
length()	Returns the **length** any given text.	NUMBER
matches()	Checks, if the given **text matches** the given **regular expression**.	BOOLEAN
project()	Returns the **key** of the **project** with a given **ID.**	TEXT
replaceAll()	Replaces all substrings matching a **regular expression** with a given **replacement**.	TEXT
replaceFirst()	Replaces the **first substring** matching a given **regular expression** with a given replacement.	TEXT
resolution()	Returns the **name** of the **resolution** with a given **ID**.	TEXT
status()	Returns a **status name**.	TEXT
substring()	Returns a **specific part** of a text.	TEXT
toLowerCase()	Converts a given **text** with all its characters to **lower case**.	TEXT
toUpperCase()	Converts a given **text** with all characters to **upper case**.	TEXT
trim()	Removes **leading and trailing blanks** (white spaces and tabs) from a text.	TEXT

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