General mode

The **general** parsing mode is a powerful mode where you can write **free text**, insert **field codes** and use JWT expression parser functions anywhere in your text.

Field codes will be replaced at runtime with the corresponding field values of the issue currently being processed.

The general mode is generally the preferred mode if you want to **set** or **update** values (from fields to issue types) based on a parser expression. Due to the specific architecture of Jira Cloud, this is currently only available when configuring Post functions.

The general mode always returns a **text**. In case the output is used as an input for a **number** (especially date) parameter or as an **issue list**, JWT for Jira Cloud internally converts the data type.

Example expressions

Parser expression	Description
`This is the issue summary: %{issue.summary}`	This example might return: **This is the issue summary: JWT 3.0 wording update**
`%{"This is the issue summary: " + %{issue.summary} + ". The issue has been created at: " + dateToString({issue.created}, "Europe/Berlin", "de")}`	This example might return: **This is the issue summary: JWT 3.0 wording update. The issue has been created at: 05/März/20**

Whenever at least one JWT expression parser function or a field code should be used, they have to be enclosed by **`%{...}`**. When you are nesting functions, only one **`%{...}`** is necessary.

Within functions, **texts** have to be written in **quotation marks**.

Quotes (**"**) within a quoted text have to be escaped (e.g. "This is a \"quoted\" text" will result in **This is a "quoted" text**).

Additional examples

Parser expression	Description
`%{subtasks()}`	This example returns all sub-tasks of the current issue. To achieve this the following functions are used: • subtasks()
`%{count(subtasks())}`	This example returns the **number** of **all sub-tasks** of the current issue. To achieve this the following functions are used: • count() • subtasks()

`This is the issue summary: "%{issue.summary}" and that of the parent "%{parent.summary}". Issue %{parent.key} has %{count(subtasks(%{parent.key}))} sub-tasks.`	This example returns a text text which contains the current **summary**, the **summary** of the **parent** issue and the **number of sub-tasks** of the parent issue. To achieve this the following functions are used: • count() • subtasks()
`%{"This is the issue summary: \"" + %{issue.summary} + "\" and that of the parent \"" + %{parent.summary} +"\". Issue " + %{parent.key} + " has "+count(subtasks(%{parent.key})) +" sub-tasks."}`	Alternative implementation of the example above written as a **single function**.

The combination of the characters **`%{`** generally indicate the use of a field code or JWT expression parser functions. If you want to use this character combination in your free text, they need to be escaped by "\":

You want to use these characters as free text...	... you need to to write this
`%{`	**`\%\{`**

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