issueSelection()

This very powerful JQL function combines the **power of classic JQL and JWT**, by letting you refine your JQL subquery using a JWT logical expression.

The function works in the following way:

1. The **JQL subquery** will be evaluated first and return a number of issues
2. These issues will then be run against a **logical expression**
3. Issues where the expression returns **TRUE** will be returned by the overall JQL query

Example

```
issue in issueSelection('project = DESK', '%{issue.subtasks.count} >= 1')
```

1. The JQL subquery will return **all issues from the DESK project**
2. The logical expression will check whether these issues have **at least one sub-task**.
3. The JQL function will **only** return the **issues with sub-tasks**.

Syntax

```
issueSelection(subquery, logicalExpression) #Output: Issue list
```

Examples

Parser expression	Description
`issue in issueSelection('project = QA', '%{issue.assignee} = %{issue.reporter}')`	This example returns all issues within the QA project where the **reporter** is also the **current assignee**.
`issue in issueSelection('type = Story', '%{issue.subtasks.count} >= 1')`	This examples returns all stories that **have at least one sub-task**.
`issue in issueSelection('project = CRM', 'count(allComments()) > 10')`	This example returns all issues within the CRM project with **more than 10 comments**. To achieve this, the following functions are used: • count() • allComments()
`issue in issueSelection('category = Support', 'count(filterByResolution(subtasks(), "")) > 0')`	This example returns all issues from projects within the Support category with **unresolved sub-tasks**. To achieve this, the following functions are used: • count() • filterByResolution() • subtasks()

`issue in issueSelection('type = Task', 'isAClone()')`	This example returns all tasks that have been **created by cloning** an issue. To achieve this, the following functions are used: • isAClone()
`issue in issueSelection('project = KANBAN', 'isInRole(%{issue.assignee}, "Developers")')`	This example returns all issues within the KANBAN project that are **assigned to users in the Developers role**. To achieve this, the following functions are used: • isInRole()
`issue in issueSelection('type = Incident', 'lastFieldChangeTime(%{issue.priority}) > ({system.currentDateTime} - 60 * {MINUTE})')`	This example returns all Incidents with a **change of priority within the last 60 minutes**. To achieve this, the following functions are used: • lastFieldChangeTime()
`issue in issueSelection("type = Bug and resolution = Unresolved", 'matches(%{issue.versions}, ".*EAP.*")')`	This example returns all unresolved Bugs with a **'EAP'-labelled version**. • matches()
`issue in issueSelection("project = HR", '!isActive(%{issue.assignee})')`	This example returns all issues within the HR project that are **assigned to inactive users**. • isActive()
`issue in issueSelection("project = JWT", 'matches(%{issue.attachments.details}, "(.*application/pdf.*){1,}")')`	This example returns all issues within the JWT project that have **at least one PDF file attached**. • matches()

Additional information

Parameters used in this function

Parameter	Input (data type)	Description
`subquery`	TEXT	A **JQL query** to select the issues that should be further filtered by the logical expression. ⚠ The subquery **must not be empty** to avoid negative performance impacts. Always try to keep the number issues returned by the subquery as minimal as possible.
`logicalExpression`	TEXT	A logical expression that returns a boolean value. See additional examples and learn how to build logical expressions here.

Pro tip

Enclose the subquery with single quotes **''** instead of double quotes **""**. Otherwise, every time a **double-quotation** is used in the subquery, they must be jumped with a **slash** /.

Output

This function returns an ISSUE LIST

The result is **not what you expected**? The number of returned issues feels **too low**?

By default, the **maximum number** of issues that will be returned by the JQL subquery, and thus can be processed by the logical expression is **1 000**.

To ensure the performance of your entire Jira instance, we limit the execution of the JQL function in terms of **issue count** and **execution time**. Please refer to your admin if you need to change the configuration.

Use cases and examples

Use case	JQL function
List all issues with an inactive assignee and that were assigned by the current user	issueSelection()
List all issues with a specific value matching a custom text field	issueSelection()
List all bugs where the current user was mentioned in a comment	issueSelection()
Return all issues in the currently open sprint with unresolved sub-tasks	issueSelection()