Copy parsed text to a field

This function has been **renamed** with the **JWT 3.0** release.

Find the new documentation at:

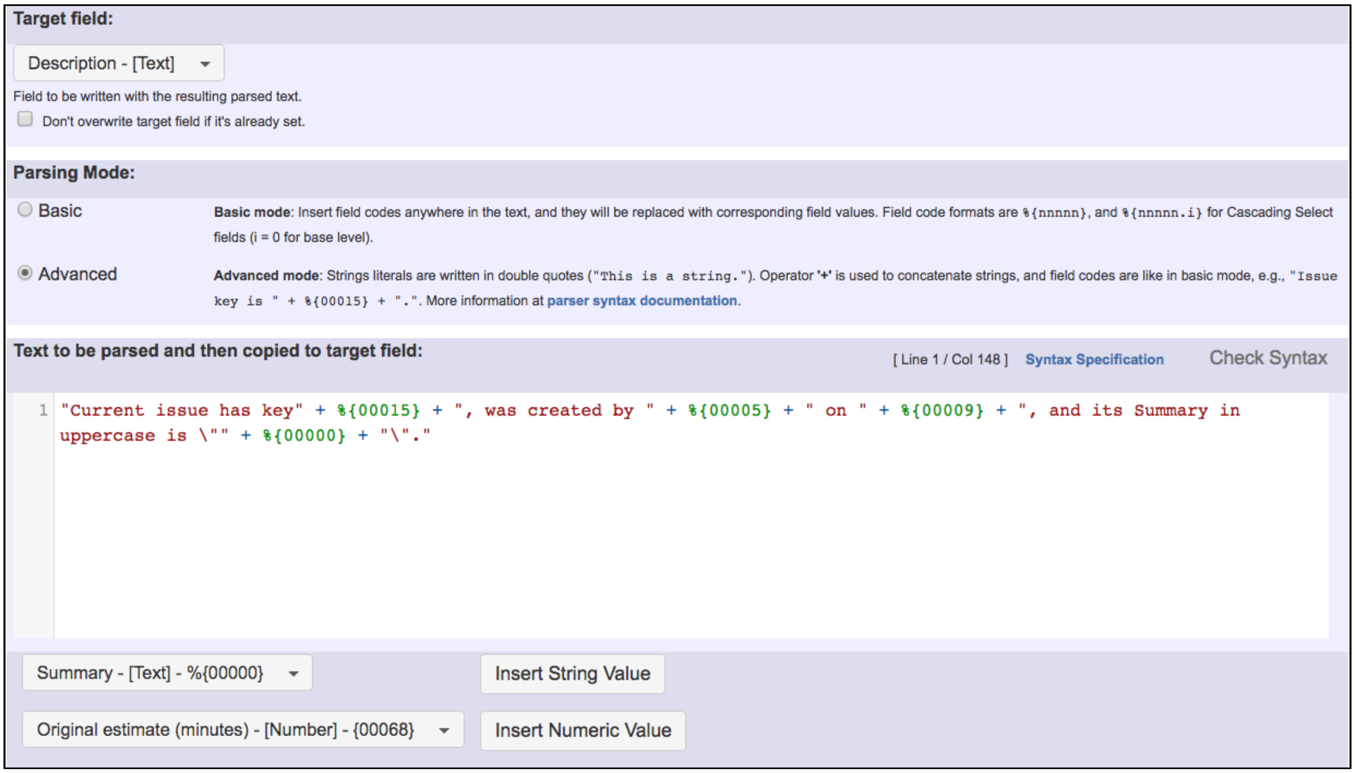
Copy a parsed text to a field

On this page

Purpose

Copy Parsed Text to a Field post-function is a general purpose tool for setting almost any kind of field (text, select, radio button, multi-select, checkboxes, cascading select, multi-cascading select, user, multi-user, dates, date-time, number, etc). Although, in the case of field types Number, Date, Date-Time and Priority (when using associated numeric value) it's usually preferable to use post-function **Mathematical and date-time expression calculator**, since you can enter complex math and time expressions.

Example 1: Compose dynamic text by inserting field values



Target field:

Description - [Text]

Field to be written with the resulting parsed text.

□ Don't overwrite target field if it's already set.

Parsing Mode:

○ Basic — **Basic mode**: Insert field codes anywhere in the text, and they will be replaced with corresponding field values. Field code formats are `%{nnnnn}`, and `%{nnnnn.i}` for Cascading Select fields (i = 0 for base level).

● Advanced — **Advanced mode**: Strings literals are written in double quotes (`"This is a string."`). Operator '+' is used to concatenate strings, and field codes are like in basic mode, e.g., `"Issue key is " + %{00015} + "."`. More information at parser syntax documentation.

Text to be parsed and then copied to target field:

```
"Current issue has key" + %{00015} + ", was created by " + %{00005} + " on " + %{00009} + ", and its Summary in uppercase is \"" + %{00000} + "\"."
```

Summary - [Text] - %{00000}
Insert String Value
Original estimate (minutes) - [Number] - {00068}
Insert Numeric Value

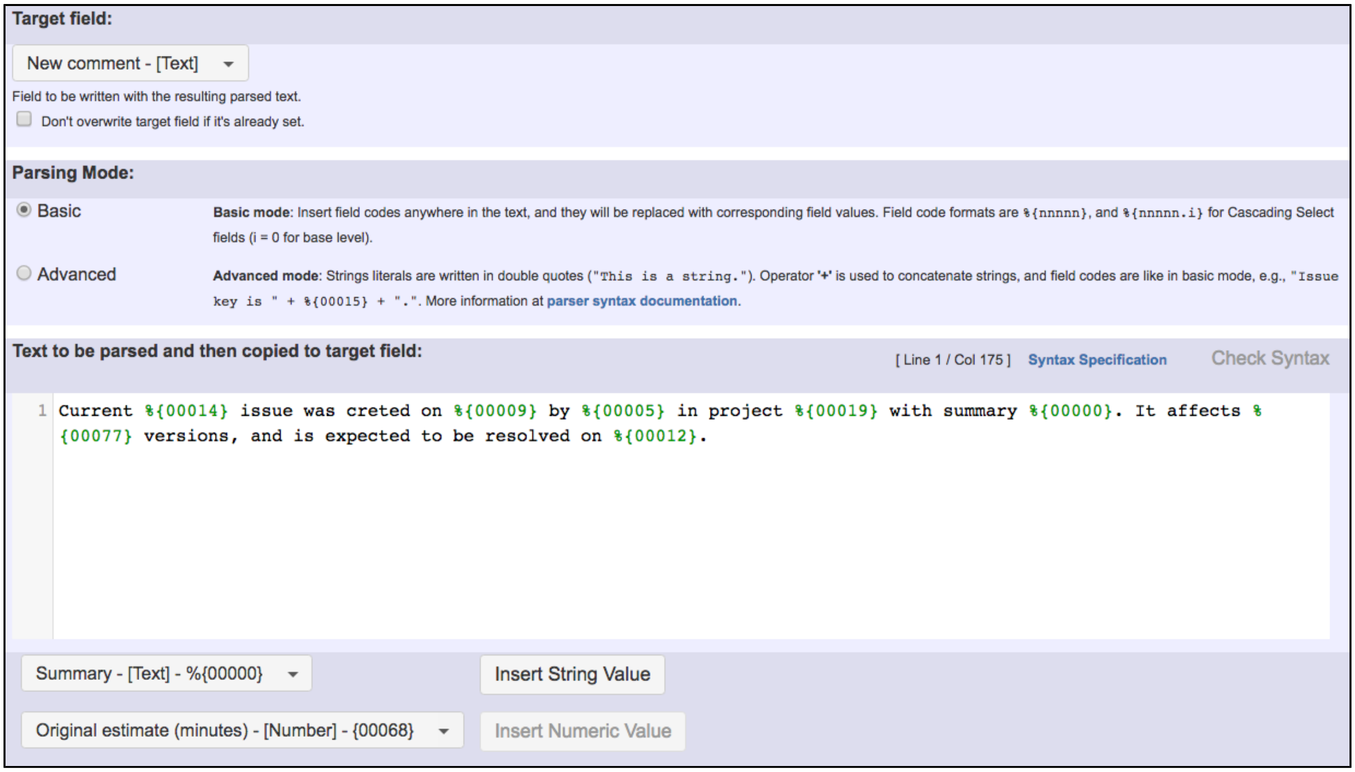
Text to be parsed in the example is: **`"Current issue has key" + %{00015} + ", was created by " + %{00005} + " on " + %{00009} + ", and its Summary in uppercase is \"" + %{00000} + "\"."`**

Note that:

- **`%{00015}`** is field code of "**Issue key**"
- **`%{00005}`** is field code of "**Reporter's full name**"
- **`%{00009}`** is field code of "**Date and time of creation**"
- **`%{00000}`** is field code of "**Summary**"

Example 2: Create a comment with custom text after issue creation

Composing dynamic text by inserting field codes with format **%{nnnnn}** among text string literals. These field codes will be replaced at transition execution with the values of the corresponding fields. The resulting text will be copied into Target field.



Target field:

New comment - [Text] ▾

Field to be written with the resulting parsed text.

☐ Don't overwrite target field if it's already set.

Parsing Mode:

● Basic — **Basic mode:** Insert field codes anywhere in the text, and they will be replaced with corresponding field values. Field code formats are `%{nnnnn}`, and `%{nnnnn.i}` for Cascading Select fields (i = 0 for base level).

○ Advanced — **Advanced mode:** Strings literals are written in double quotes (`"This is a string."`). Operator '**+**' is used to concatenate strings, and field codes are like in basic mode, e.g., `"Issue key is " + %{00015} + "."`. More information at **parser syntax documentation**.

Text to be parsed and then copied to target field:

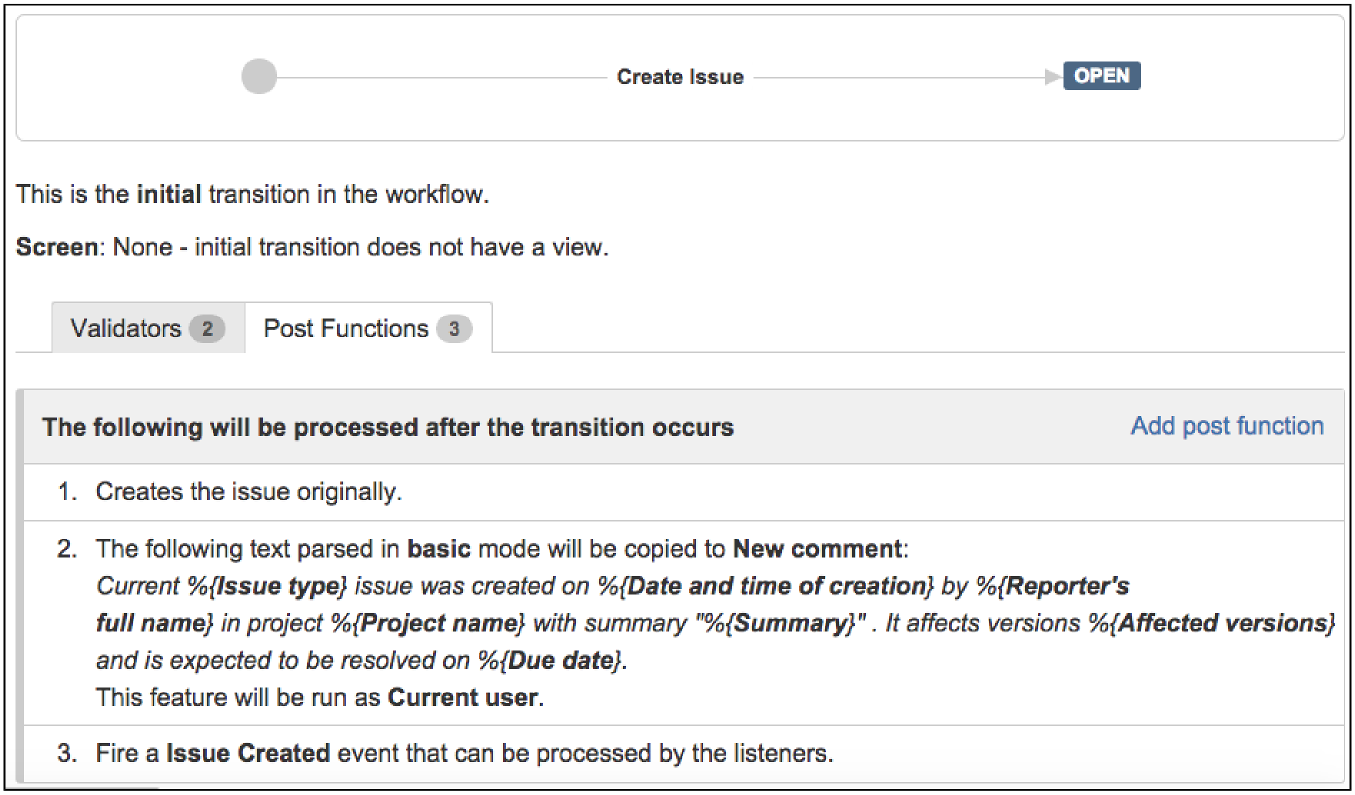
[Line 1 / Col 175] **Syntax Specification** Check Syntax

```
Current %{00014} issue was creted on %{00009} by %{00005} in project %{00019} with summary %{00000}. It affects %
{00077} versions, and is expected to be resolved on %{00012}.
```

Summary - [Text] - %{00000} ▾
Insert String Value
Original estimate (minutes) - [Number] - {00068} ▾
Insert Numeric Value

Text to be parsed is: **`Current %{00014} issue was created on %{00009} by %{00005} in project %{00019} with summary %{00000}. It affects %{00077} versions, and is expected to be resolved on %{00012}.`**

Once configured, transition will look like this:



This is the **initial** transition in the workflow.

Screen: None - initial transition does not have a view.

The following will be processed after the transition occurs

1. Creates the issue originally.
2. The following text parsed in **basic** mode will be copied to **New comment**:
   *Current %{**Issue type**} issue was created on %{**Date and time of creation**} by %{**Reporter's full name**} in project %{**Project name**} with summary "%{**Summary**}" . It affects versions %{**Affected versions**} and is expected to be resolved on %{**Due date**}.*
   This feature will be run as **Current user**.
3. Fire a **Issue Created** event that can be processed by the listeners.

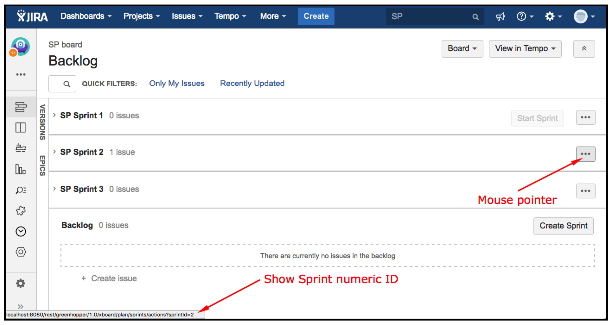
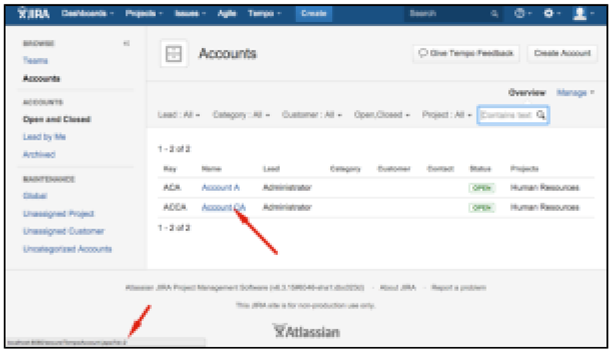
Example 3: Set or update the value of any type of custom field or virtual field

This is a very versatile feature that allows you to set the value of any custom field or issue feature in almost any way you may require.

Setting custom fields

Field type	Expected values and format when setting the field by a post-function	Examples
Check boxes	Comma separated list of options to be selected. Admits + and - prefixes to add and remove single items.	- blue, red, white - + blue - -red, white
Radio Buttons	Name of the choice to be selected.	
Select List (single choice)	Name of the choice to be selected.	
Select List (multiple choices)	Comma separated list of options to be selected. Admits + and - prefixes to add and remove single items.	- blue, red, white - + blue - -red, white

Select List (multi-level cascading)	**Multi-Level Cascading Select** app supported: comma separated list of values for each level of the cascade.	America, USA, California, San Francisco
Text Field (single line)	Text value (any other type will be cast to text)	
Text Field (multi-line)	Text value (any other type will be cast to text)	
Labels	space separated list of labels. Admits + and - prefixes to add and remove single items.	web crm question + web crm - question
Group Picker (single group)	**Name** of a group.	Administrators
Group Picker (multiple groups)	Comma separated list of **group names**. Admits + and - prefixes to add and remove single items.	Administrators, Developers, Users + Administrators - Developers, Users
User Picker (single user)	Name of a **user name** (not full name) or **name of a project role**. For **project roles** please make sure to **set up a default user for a project role (Example 1)** before. Supported apps: **Issue Alternative Assignee**	john
User Picker (multiple users)	Comma separated list of **user names** (not full names), **name of a group** (every user in the group will be set), or **name of a project role** (every user in the project role will be set). Admits + and - prefixes to add and remove single items.	john, mary Developers + john - mary, Administrators
Version Picker (single version)	Version name.	1.0
Version Picker (multiple versions)	Comma separated list of version names. Admits + and - prefixes to add and remove single items.	1.0, 1.1, 2.0 + 1.0, 1.1 - 2.0
Project Picker (single project)	Project **key** or project **name**.	CRM Customer Relationship Management
Issue Picker	**Issue Picker** app supported: comma separated list of **issue keys**. Admits + and - prefixes to add and remove single items.	CRM-1, CRM-12, HR-254 + CRM-1, HR-254 - CRM-12

Sprint	Jira Software's Sprint field can be set by writing on it the **numerical ID** of a Sprint.	Sprint's numeric ID can be obtained by moving the mouse pointer over some controls of JIRA UI as shown in this screenshot.  JIRA Dashboards Projects Issues Tempo More Create SP board Backlog QUICK FILTERS: Only My Issues Recently Updated SP Sprint 1 0 issues Start Sprint SP Sprint 2 1 issue SP Sprint 3 0 issues Mouse pointer Backlog 0 issues Create Sprint There are currently no issues in the backlog + Create issue Show Sprint numeric ID localhost:8080/rest/greenhopper/1.0/xboard/plan/sprints/actions?sprintId=2 Requires version **2.2.28** or higher.
Customer Request Type	JSD's Customer Request Type field can be set by writing on it a Request Type's **key**.	Request type keys can be obtained by executing **Copy a parsed text to a field** post-function for writing the value of a field **Customer Request Type** with different values. A Request Type key begins by a **project key** followed by /. The default Request Type keys for a project with key ISD are: - **Report a system problem**: isd/systemproblem - **Get IT help**: isd/getithelp - **Fix an account problem**: isd/accountproblem - **Get a guest wifi account**: isd/guestwifi - **Set up VPN to the office**: isd/vpn - **Request admin access**: isd/adminaccess - **Request a new account**: isd/newaccount - **Onboard new hires**: isd/newhires - **Desktop/Laptop support**: isd/compsupport - **Set up a phone line redirect**: isd/phoneredirect - **Request new software**: isd/newsoftware - **Request new hardware**: isd/newhardware - **Upgrade or change a server**: isd/upgradeserver Requires version **2.2.28** or higher.
"**Account**" and "**Team**" TEMPO Timesheet's custom fields	Values used for setting these fields are **internal numerical IDs**. These IDs can be found out.  JIRA Accounts Create Account Account A Account OA Atlassian Since version **2.2.26** field **Account** can also be set using a string value with the **name** or the **key** of the account. When field Account is not set the value returned is `-1` (versions of TEMPO Timesheet earlier than 8.0), or **null** (TEMPO Timesheet 8.0 or higher).	- 1 - 2 - 11

Note: All the examples above are shown in double quotes for readability, but if you are using basic parsing mode you should remove them.

Configuration Parameters

Parsing modes

This post-function provides **2 parsing modes**:

- **Basic**: in this simple parsing mode you can write free text and insert field codes with format **%{nnnnn}** or **%{nnnnn.i}** anywhere in your text. These field codes will be replaced at execution time with the string field values of the corresponding fields.

- **Advanced**: This is the new parsing mode that was been introduced in version 2.1.18. It requires the text to be parsed to be written as a **text-string expression** respecting strictly the parsers grammar, otherwise you will get an parsing error. The new mode allows to insert math and time formulas, and text formatting function calls (trim, toUpperCase, toLowerCase, replaceAll, etc), making it much more powerful than the basic mode.

  Automatic parsing mode converter: You can write your text in basic mode, and then switch to advanced mode. The text to be parsed will be automatically rewritten as a text-string expression. Now you can simply make the modifications you require, making use of text formatting functions, or inserting math or time expressions where needed.

Add or remove Single Items or Subsets of Items from Multi-valued fields (using prefixes + or -)

Syntax specification:

- **+ comma_separated_list_of_values** : adds a set of values to the ones currently set in a multi-valued field.

- **- comma_separated_list_of_values** : removes a set of values from the ones currently set in a multi-valued field.
  Don't write de double quotes.

  Syntax examples: (don't write the double quotes)
- **+ somevalue** : adds a new value called somevalue.
- **- valueA, valueB** : removes values valueA and valueB.
- **+ valueA, valueC, - valueC, valueD** : enables, checks or adds values valueA and valueB and disables, unchecks or removes values valueC and valueD.

  Types of fields prefixes + and - work on:
- **Checkboxes**
- **Select List (multiple choices)**
- **Group Picker (multiple groups)**
- **User Picker (multiple users)**
- **Version Picker (multiple versions)**
- **Fixed versions**
- **Affected versions**
- **Labels**
- **Components**
- **Issue Picker**

  Examples:
- **Add and remove a single or a set of items from multi valued fields**
- **Combine the values of several Multi-User picker fields**
- **Add all assignees of certain sub-task types to a "Multi-User Picker" custom field**

Set value of Selectable or Multi-Valued fields by Regular Expressions

Fields will be set with all valid options or items available for it that fulfills a certain regular expression (syntax: "**/regexp/**"), or not fulfilling it (syntax: "**//regexp/**"):

- **Checkboxes**
- **Radio Buttons**
- **Select List (single choice)**
- **Select List (multiple choices)**
- **Group Picker (multiple groups)**
- **User Picker (multiple users)**
- **Version Picker (multiple versions)**
- **Fixed versions**
- **Affected versions**
- **Labels**
- **Components**

  Examples:
- **Set a Select or Multi-Select field using regular expression to express the values to be assigned**

Change Issue Features: (more details at **Virtual Fields**)

- **Summary**
- **Description**
- **Assignee**
- **Environment**

- **Reporter**
- **Priority**
- **Resolution**
- **Issue status**: executes transitions in workflow to move issue to written status.
- **Issue status (delayed writing)**: same as **Issue status**, only that waits for current transition to finish execution.
- **Execute transition**: expects the name of a transition, and executes it if all the conditions and validations are satisfied.
- **Execute transition (delayed execution)**: same as **Execute transition**,only that waits for current transition to finish execution.
- **Attachments (current attachments will be replaced)**: writing **issue keys** on this field will make attachments from those issues to replace current attachments.
- **Attachments (only new attachments will be added)**: writing **issue keys** on this field will make attachments from those issues to be added to current issue.
- **Last comment**: it overwrites the last comment of the issue.
- **Last comment's visibility restrictions**: it set the visibility for the last comment of the issue. It expects the name of a **group** or of a **project role**.
- **New comment**: adds a new comment to the issue with no visibility restrictions. It expects the text of the comment.
- **Labels**: it overwrites the labels for the issue. Prefixes **+** and **-** can be used for adding or removing single labels.
- **Components**
- **Affected versions**
- **Fixed versions**
- **Security level**
- **Watchers**: it overwrites the current watchers of the issue. It expects a comma separated list of user names.
- **New watchers**: it expects a comma separated list of user names, and makes them watchers of the issue, maintaining current watchers.

Set Numeric and Date-Time fields: (more details at **Virtual Fields**)

- **Due date**
- **Original estimate (minutes)**
- **Remaining estimate (minutes)**
- **Total time spent (minutes)**
- **Add to time spent (minutes)**: adds the number of minutes written into this field to the current work log of the issue.
- **Date picker**
- **Date and Time picker**
- **Number** custom fields

  Fields introduced by plugin **JIRA Misc Custom Fields** are also supported.

  Although it's possible to set numeric and date and time fields with post-function copy parsed text, it's much better to use post-function **Mathematical and date-time expression calculator**, which allows to do complex mathematical and time calculations.

Combining with other post-functions

You will often use **Copy Parsed Text to a Field** in combination with other post-functions, like **Write field on linked issues or sub-tasks**, **Update issue fields** and **Set a field as a function of other fields**.

You should select an **Ephemeral field** in parameter **Target field**, in order to hold a temporary parsed text that will be used by the next post-function. In the case of post-functions **Write field on linked issues or sub-tasks** and **Update issue fields** the resulting parsed text stored in an **ephemeral field** will be written into a field in other issues (linked issues, transitively linked issues, sub-tasks, sibling sub-tasks, or any issue returned by a JQL query).

Usage Examples

Page: Add all assignees of certain sub-task types to a "Multi-User Picker" custom field
Page: Add and remove a single or a set of items from multi valued fields
Page: Add current user to comment
Page: Add or remove request participants
Page: Add watchers from a part of the issue summary: "Summary_text - watcher1, watcher2, watcher3, ..."
Page: Assign issue based on the value of a Cascading Select custom field
Page: Assign issue to last user who executed a certain transition in the workflow
Page: Automatically close resolved sub-tasks when parent issue is closed
Page: Automatically reopen parent issue when one of its sub-tasks is reopened
Page: Calculate the time elapsed between 2 transition executions
Page: Close parent issue when all sub-tasks are closed
Page: Combine the values of several Multi-User picker fields
Page: Compose a parsed text including the "full name" or a user selected in a User Picker custom field
Page: Compose dynamic text by inserting field values in a text template

Related Features

- **Parse field for extracting data**
- **Format field value**

Page: Copy issue labels to a custom field
Page: Copy the value of a user property into a user picker
Page: Create a comment in sub-tasks when parent transitions
Page: Execute transition in epic
Page: Getting the number of selected values in a custom field of type Multi Select
Page: Limit the number of hours a user can log per day
Page: Make a sub-task's status match parent issue's current status on creation
Page: Make parent issue progress through its workflow
Page: Moving story to "In Progress" when one of its sub-tasks is moved to "In Progress"
Page: Moving story to "Ready for QA" once all its sub-tasks are in "Ready for QA" status
Page: Parse Email adresses to watchers list
Page: Parsing text from last comment and appending it to issue's summary
Page: Remove versions selected in a version picker custom field
Page: Replace certain issue link types with different ones
Page: Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status
Page: Set a Select or Multi-Select field using regular expression to express the values to be assigned
Page: Set assignee depending on issue type
Page: Set field depending on time passed since issue creation
Page: Set priority for issues that have been in a certain status for longer than 24 hours
Page: Set security level based on groups and project roles the reporter or creator are in
Page: Transition linked issues in currently active sprint
Page: Transition only a sub-task among several ones
Page: Transition parent issue only when certain issue sub-task types are done
Page: Update Cascading Select custom field with a value of the field in parent issue
Page: Update checkboxes custom field if a file has been attached during a transition
Page: Validation on issue attachments
Page: Validation on MIME types of issue attachments
Page: Writing a comment to blocked issues when blocking issues are resolved