Historical field values

On this page

Overview

The expression parser offers multiple functions to retrieve historical field values.

Functions for accessing historical field values are available for the following fields:

- All **Custom Fields**
- Summary
- Description
- Assignee
- Reporter
- Due date
- Issue status
- Priority
- Resolution
- Environment
- Fix version/s
- Affects version/s
- Labels
- Components
- Security level

Available functions

Function	Input	Returned value
previousValue(%{... somefield})	FIELD	Returns a STRING with the previous value of a field for current issue. It will return **`null`** if field was previously uninitialized.
previousValue({... somefield})	FIELD	Returns a NUMBER with the previous value of a **numeric** or **date field** for current issue. It will return **`null`** if field was previously uninitialized.
previousValue(%{... somefield.i})	FIELD	Returns a STRING with the previous value of a **cascading** or **multi-cascading select field** for current issue **at level i** (with root level = 0). It will return **`null`** if field was previously uninitialized.
fieldHistory(%{... somefield})	FIELD	Returns a STRING [] with all the values that a field has ever had in the past for current issue. Values appear in the list in ascending ordered by setting time, i.e., older value has index `1`, and most recent value has index **`count(string_list)`** . Uninitialized field statuses are represented by **empty strings**.
fieldHistory({... somefield})	FIELD	Returns a NUMBER [] with all the values that a **numeric** or **date-time field** has ever had in the past for current issue. Values appear in the list in ascending ordered by setting time, i.e., older value has index `1`, and most recent value has index **`count(number_list)`** . Uninitialized field statuses are not represented.
fieldHistory(%{... somefield.i})	FIELD	Returns a STRING [] with all the values that a **cascading** or **multi-cascading select field** has ever had in the past **for level i** (with root level = 0) in current issue. Values appear in the list in ascending ordered by setting time, i.e., older value has index `1`, and most recent value has index **`count(string_list)`** . Uninitialized field statuses are represented by empty strings.

hasChanged(%{...somefield})	FIELD	Returns BOOLEAN **true** only if field has changed in current transition. Function **hasChanged(field_code)** is used when we set a validation that is incompatible with a condition in a same transition, typically when validating a value entered in the transition screen. When Jira evaluates the validations in a transition, it also reevaluates the conditions, and if they are not satisfied an **`Action X is invalid`** error message is shown and the transition is not executed. **Example:** Let's suppose we have a boolean condition like **`{...duedate} = null`** (i.e., **Due date = null**) in a transition, so that it's only shown when **Due date** is empty. This transition also has a transition screen containing field **Due date**, and a boolean validation **`{...duedate} != null`** , in order to make **Due date** required in the transition. The configuration described above will not work, since both condition and validation are mutually incompatible. We can fix it replacing the boolean condition with **`{...duedate} = null OR hasChanged(%{...duedate})`** .
hasChanged({...somefield})	FIELD	Returns BOOLEAN **true** only if **numeric** or **date-time field** field has changed in current transition.
hasChanged({...somefield.i})	FIELD	Returns BOOLEAN **true** only if **cascading select field** has changed **for level i** (with root level = 0) in current transition.