

Make different fields mandatory depending on the value of a Select List custom field

On this page

- [Features used to implement the example](#)
- [Example: Make different fields mandatory depending on the value of a Select List custom field](#)
- [Other variation of the usage example](#)
- [Other examples of that function](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Boolean validator with math, date-time or text-string terms](#)

Example: Make different fields mandatory depending on the value of a Select List custom field

We have 3 custom fields

1. "**Development Type**" with values "**Core**" and "**Custom**"
2. "**Story Points**"
3. "**Hours**"

So the requirement is, if Development Type = "**Core**", the Story Points field would be displayed and if Development Type = "**Custom**", the Hours field would be displayed.

It is possible to set 2 validations in your transition, in order to ensure that:

1. When "**Development Type** = Core" then field "**Story Points**" will be **set** and given a value **higher than 0**, and field "**Hours**" is **not set** or **zero**
2. When "**Development Type** = Custom" then field "**Hours**" will be **set** and given a value **higher than 0**, and field "**Story Points**" is **not set** or **zero**

To do it, you just need to use [Boolean validator with math, date-time or text-string terms](#) with the following configurations:

When "**Development Type** = Core" then field "**Story Points**" will be **set** and entered a value **higher than 0**, and field "**Hours**" is **not set** or **zero**:



```
1 %{12500} = "Core" IMPLIES ({12501} != null AND {12501} > 0 AND ({12502} = null OR {12502} = 0))
```

Logical connectives: and, or and not. Alternatively you can also use &, | and !.

Comparison operators: =, !=, >, >=, < and <=. Operators ~, !~, in, not in, any in and none in can be used with **strings, multi-valued fields and lists**.

[Check Syntax](#)

Logical literals: true and false. Literal null is used with = and != to check whether a field is initialized, e.g. {00012} != null checks whether **Due Date** is initialized.

NUMERICAL AND DATE-TIME TERMS

Numeric and Date-Time field values: insert field codes with format {nnnnn}.

Original estimate (minutes) - [Number] - {00068} ▾

Insert Numeric Value

Valid date-time literal formats: yyyy/MM/dd [hh:mm] or yyyy-MM-dd [hh:mm]. Time literals use format: hh:mm.

There is a set of **mathematical functions** and **time macros and functions** available to be used in your expression.

TEXT-STRING TERMS

Text-String field values: insert field codes with format %{nnnnn} or %{nnnnn.i} for referencing levels in cascading select fields (*i* = 0 for base level).

Summary - [Text] - %{00000} ▾

Insert String Value

String literals: written in **double quotes**, e.g., "This is a string literal."

String concatenation: use operator '+' to concatenate string values, e.g., "The summary of issue with key " + %{00015} + " is \" + %{00000} + "\"."

Escape character: character \ is used with characters "", \, 'n', 'r', 't', 'f' and 'b' to invoke an alternative interpretation.

There is a set of **string functions** available to be used in your expression.

Skip validation when:

Inhibit the validator under selected circumstances.

- Transition is triggered by a **bulk operation**.
- Transition is triggered by a **JIRA Workflow Toolbox post-function**.
- Current issue is being created by cloning. Only makes sense in *Create Issue* transition.

Message to show when validation fails:

Core type developments requires "Story Points" to be entered, and doesn't admit "Hours" to be set.

Field code injector:

Summary - [Text] - %{00000} ▾

Field codes with format %{nnnnn} can be inserted in the failure message and its translations, and they will be replaced with actual field values at runtime.

[Set translations for installed languages](#)

Expression used is: `%{12500} = "Core" IMPLIES ({12501} != null AND {12501} > 0 AND ({12502} = null OR {12502} = 0))`

Other variation of the usage example

When "Development Type = Custom" then field "Hours" will be **set** and given a value **higher than 0**, and field "Story Points" is **not set** or **zero**:



```
1 %{12500} = "Custom" IMPLIES ({12502} != null AND {12502} > 0 AND ({12501} = null OR {12501} = 0))
```

Logical connectives: and, or and not. Alternatively you can also use &, | and !.

Comparison operators: =, !=, >, >=, < and <=. Operators ~, !~, in, not in, any in and none in can be used with **strings, multi-valued fields and lists**.

[Check Syntax](#)

Logical literals: true and false. Literal null is used with = and != to check whether a field is initialized, e.g. {00012} != null checks whether **Due Date** is initialized.

NUMERICAL AND DATE-TIME TERMS

Numeric and Date-Time field values: insert field codes with format {nnnnn}.

Original estimate (minutes) - [Number] - {00068} ▾

Insert Numeric Value

Valid date-time literal formats: yyyy/MM/dd [hh:mm] or yyyy-MM-dd [hh:mm]. Time literals use format: hh:mm.

There is a set of **mathematical functions** and **time macros and functions** available to be used in your expression.

TEXT-STRING TERMS

Text-String field values: insert field codes with format %{nnnnn} or %{nnnnn.i} for referencing levels in cascading select fields (*i* = 0 for base level).

Summary - [Text] - %{00000} ▾

Insert String Value

String literals: written in **double quotes**, e.g., "This is a string literal."

String concatenation: use operator '+' to concatenate string values, e.g., "The summary of issue with key " + %{00015} + " is \" + %{00000} + "\"."

Escape character: character \ is used with characters "", \, 'n', 'r', 't', 'f' and 'b' to invoke an alternative interpretation.

There is a set of **string functions** available to be used in your expression.

Skip validation when:

Inhibit the validator under selected circumstances.

- Transition is triggered by a **bulk operation**.
- Transition is triggered by a **JIRA Workflow Toolbox post-function**.
- Current issue is being created by cloning. Only makes sense in *Create Issue* transition.

Message to show when validation fails:

Custom type developments requires "Hours" to be entered, and doesn't admit "Story Points" to be set.

Field code injector:

Summary - [Text] - %{00000} ▾

Field codes with format %{nnnnn} can be inserted in the failure message and its translations, and they will be replaced with actual field values at runtime.

[Set translations for installed languages](#)

Expression used is: `%{12500} = "Custom" IMPLIES ({12502} != null AND {12502} > 0 AND ({12501} = null OR {12501} = 0))`

Note that in the particular JIRA instance I used to build this example:

- `%{12500}` is field code for string value of field **"Development Type"**
- `{12501}` is field code for numeric value of field **"Story Points"**
- `{12502}` is field code for numeric value of field **"Hours"**

Once configured, your transition will look like this:

Triggers 0

Conditions 0

Validators 2

Post Functions 13

The transition requires the following criteria to be valid

Add validator

Only if the following boolean expression is true: ***%{Development Type} = "Core" IMPLIES ({Story Points} != null AND {Story Points} > 0 AND ({Hours} = null OR {Hours} = 0))***

Message to show when validation fails: **"Core type developments requires "Story Points" to be entered, and doesn't admit "Hours" to be set."**

Only if the following boolean expression is true: ***%{Development Type} = "Custom" IMPLIES ({Hours} != null AND {Hours} > 0 AND ({Story Points} = null OR {Story Points} = 0))***

Message to show when validation fails: **"Custom type developments requires "Hours" to be entered, and doesn't admit "Story Points" to be set."**

Other examples of that function

Page: Block a transition until all sub-tasks have certain fields populated

Page: Block an epic's transition depending on linked issues status and due date

Page: Block or hide a transition for an issue depending on its issue links

Page: Block or unblock a transition after an issue rested a specific time in a status

Page: Block transition until all sub-tasks are in a specific status category

Page: Close parent issue when all sub-tasks are closed

Page: Enforce a field (Select List) to be set when another field (Radio Button) has a certain value (works with any kind of field type)

Page: Ensure that all issues linked with a certain issue link type have "Due Date" field set

Page: If field A is populated then, field B must also be populated

Page: Limit issue creation per role and issue type

Page: Limit the number of hours a user can log per day

Page: Limit valid dates for work logs

Page: Make "Time Spent" field required when there is no time logged in the issue

Page: Make a custom field mandatory when priority is "Critical" or "Blocker" and issue type is "Incident"

Page: Make attachment mandatory depending on the value of certain custom field

Page: Make different fields mandatory depending on the value of a Select List custom field

Page: Make linked issues, sub-tasks and JQL selected issues progress through its workflows

Page: Make parent issue progress through its workflow

Page: Prevent issue creation if another issue with same field value already exists

Page: Reject duplicated file names in attachments

Page: Require at least one sub-task in status "Resolved" or "Closed" when "Testing required" is selected in Check-Box custom field

Page: Require issue link when resolving as duplicate

Page: Restrict parent issue from closing if it has sub-tasks that were created during a given parent issue status

Page: Restrict sub-task type creation depending on parent issue status

Page: Restrict sub-task type creation depending on parent issue type

Page: Set a condition in a global transition which only applies in a certain status

Page: Validate a custom field "Story Points" has been given a value in Fibonacci sequence

Page: Validate compatible values selection among dependent custom fields

Page: Validate only issue links created in transition screen

Page: Validate that multi-user picker custom field A does not contain any user in multi-user picker custom field B

Page: Validation and condition based on time expressions

Page: Validation based on the value of a date type project property

Page: Validation on issue attachments

Related Usage Examples

- Validate compatible values selection among dependent custom fields
 - example
 - validator
 - custom-field
- Validate a custom field "Story Points" has been given a value in Fibonacci sequence
 - example
 - validator
 - custom-field
- Validate that multi-user picker custom field A does not contain any user in multi-user picker custom field B
 - example
 - validator
 - custom-field
- Validation on the value of a Cascading Select field
 - example
 - validator
 - custom-field
- Make different fields mandatory depending on the value of a Select List custom field
 - example
 - validator
 - custom-field
- Make a custom field mandatory when priority is "Critical" or "Blocker" and issue type is "Incident"
 - example
 - validator
 - custom-field
- Enforce a field (Select List) to be set when another field (Radio Button) has a certain value (works with any kind of field type)
 - example
 - validator
 - custom-field
- Make attachment mandatory depending on the value of certain custom field
 - example
 - validator
 - custom-field
- Create a dynamic set of sub-tasks based on checkbox selection with unique summaries
 - example
 - post-function
 - custom-field
 - sub-task
- Total of all story points in an epic
 - example
 - custom-field
 - calculated-field

Page: Validation on MIME types of issue attachments

Page: Validation on sibling sub-tasks depending on issue type and status

Page: Validation on the value of a Cascading Select field

- Show timeliness of an issue based on two date pickers
 - example
 - custom-field
 - calculated-field
- Add and remove a single or a set of items from multi valued fields
 - example
 - post-function
 - custom-field
 - issue-links
 - sub-task
- Highest value of a custom field among linked issues
 - example
 - custom-field
 - calculated-field
- Google Maps location from address
 - example
 - calculated-field
 - custom-field
- Make certain custom field required in resolve screen only if the resolution was set to "Fixed"
 - example
 - validator
 - custom-field