

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

On this page

- [Features used to implement the example](#)
- [Example: Automatically move issues to "In Progress" status once all its blocker issues are moved to "Closed", "Resolved" or "Done" statuses](#)
- [Other examples of that functions](#)
- [Related Usage Examples](#)

Features used to implement the example

- [Write field on linked issues or sub-tasks](#)
- [Update issue fields](#)
- [Block or hide a transition for an issue depending on its issue links](#)
- [Boolean Validator with math, date-time or text-string terms](#)

Jira Workflow Toolbox provides features for making issues automatically progress through the workflow. Issues that can be transitioned this way are:

- Using post-function [Write field on linked issues or sub-tasks](#)
 - **Linked issues**
 - **Subtasks**
- Using post-function [Update issue fields](#)
 - Issues returned by a **JQL query**
 - Issues returned by a **Issue List expression**

Making issues progress through its workflow using a post-function

An issue can transition other issues progress through its workflow by executing a post-function for writing the **name of target status** (e.g., "Open", "In Progress", "Closed", ...) into virtual fields "Issue status" or "Issue status (delayed writing)", or the name of a transition into virtual fields "Execute transition" and "Execute transition (delayed execution)" on other issues.

Care must be taken to **write the name of the status or transition exactly as it is**. Since version 6.x, Jira shows almost always the status name in upper case. To check the real name of the status, try to edit it at **Administration > Issues > Statuses > Edit**. Since version **2.1.20**, JIRA Workflow Toolbox is case insensitive for issue status, so you can write the name of the status as you see it in UI.

Writing into virtual field "Issue status"

When the **name of a status** (e.g., "Open", "In Progress", "Closed", ...) is written into virtual field "Issue status" on another issue, that issue will progress through its workflow to that status, whenever there is a transition from current status to the new one, and conditions and validators in that transition are currently satisfied. **Post-functions in that transition will also be executed**, the same way they are when transition is triggered manually.

Obviously, if there isn't a transition from current status to target status in the workflow of the issue we want to move, it will remain in its current status.

Why is needed another virtual field: "Issue status (delayed writing)"?

"Issue status (delayed writing)" works like virtual field "Issue status", with the only difference that **effective status change is carried out after transition execution in current issue (i.e., the issue executing the post-function) has been completed**. On the other hand, **when writing into "Issue status", status change is done immediately (i.e. within transition)**. This delayed way of writing is very useful when the issue you want to make progress through its workflow (sub-tasks, linked issues or JQL selected issues) is **blocked by the status of current issue**. The cause for this blockage might be a condition or a validator that depends on current issue's status, like [Block or hide a transition for an issue depending on its issue links](#), [Condition and validation on sub-tasks](#), [Condition and validation based on JQL query](#) and [Boolean condition and validator with math, date-time or text-string terms](#).

Virtual fields "Execute transition" and "Execute transition (delayed execution)"

Since version **2.1.20**, there are two additional virtual fields for transitioning issues: "Execute transition" and "Execute transition (delayed execution)". These fields behave similarly to "Issue status" and "Issue status (delayed writing)" respectively, with the only difference that they expect the **name of a transition**, instead of the name of a status.

These new fields are particularly useful if you have **global transitions in your workflow**, or if you have **more than one transition available for going to target status**. With them you specify which transition you want to be executed.

Hiding a transition to interactive Jira users

Sometimes we add transitions to the workflow, intended to be triggered exclusively by post-functions. In these cases we hide the transition to interactive (human) Jira users using condition "**Transition is triggered by Jira Workflow Toolbox post-function**".

Features used for transitioning issues

- Post-function [Write field on linked issues or sub-tasks](#) or [Update issue fields](#), used to write into virtual fields "**Issue status**" or "**Issue status (delayed writing)**" the **name of the target status** we want to move issues to, or into virtual fields "**Execute transition**" or "**Execute transition (delayed execution)**" the **name of the transition** we want to execute.
 - Virtual fields "**Issue status**" or "**Issue status (delayed writing)**": when the **name of a status** is written into any of these fields, the plugin tries to execute any transition available from current issue status to the written status. Conditions and validations in the transition should be satisfied. These fields are selected in parameter **Target field** of post-function.
 - Virtual fields "**Execute transition**" or "**Execute transition (delayed execution)**": when the **name of a transition** is written into any of these fields, the plugin tries to execute a transition with the given name from current issue status. Conditions and validations in the transition should be satisfied. These fields are selected in parameter **Target field** of post-function.
-

Example: Automatically move issues to "In Progress" status once all its blocker issues are moved to "Closed", "Resolved" or "Done" statuses

In this example we edit 2 transitions in our workflows:

1. Transition "**Start Progress**": we add a validation into transition "**Start Progress**" in order to prevent its execution whenever there are blocking issues in statuses different from "**Done**", "**Resolved**" and "**Closed**".
2. Transitions "**Done**", "**Resolve Issue**" and "**Close Issue**": we add a post-function for moving blocked issues to "**In Progress**" status. The validator added previously in "**Start Progress**" transition will prevent the execution of this transition while there still are blocking issues not yet closed or resolved.

These 2 transitions might be in the same workflow, or belong to different workflows.

Configuration of transition "Start Progress"

You can use any of these 2 validators in transition "**Start progress**" to block its execution until all blocking issues are in statuses "**Resolved**", "**Closed**" or "**Done**":

Validator [Block or hide a transition for an issue depending on its issue links](#):

Issue link types:	☒ is blocked by ☐ blocks ☐ is cloned by ☐ clones ☐ is duplicated by ☐ duplicates ☐ has Epic ☐ is Epic of ☐ is caused by ☐ causes ☐ relates to ☐ relates to Selected issue link types will be allowed. Nevertheless, if you didn't select any issue link type, and checked "Allow unselected issue link types", there won't be applied any filter by issue link type, i.e., every issue link type will be allowed.
Issue types for linked issues:	☐ Epic ☐ Story ☐ Test Plan ☐ Bug ☐ New Feature ☒ Task ☐ Improvement ☐ Sub-task Linked issues of selected issue types will be allowed. Nevertheless, if you didn't select any issue type, and checked "Allow unselected issue types", there won't be applied any filter by issue type, i.e., every issue type will be allowed.

Statuses for linked issues:

- ☐  Open
- ☐  In Progress
- ☐  Reopened
- ☒  Resolved
- ☒  Closed
- ☐  To Do
- ☒  Done
- ☐  Acceptance
- ☐  Fail
- ☐  Pass
- ☐  Retest
- ☐  Active
- ☐  Inactive

Linked issues in selected statuses will be allowed. Nevertheless, if you didn't select any issue status, and checked "Allow unselected statuses", there won't be applied any filter by status, i.e., every issue status will be allowed.

Resolutions for linked issues:

- ☐ **UNRESOLVED**, i.e. no resolution value
- ☐ Fixed
- ☐ Won't Fix
- ☐ Duplicate
- ☐ Incomplete
- ☐ Cannot Reproduce

Linked issues with selected resolutions will be allowed. Nevertheless, if you didn't select any issue resolution, and checked "Allow unselected resolutions", there won't be applied any filter by resolution, i.e., every issue resolution will be allowed.

Linked issues must belong to:

- ☒ any project
- ☐ current project
- ☐ any but current project
- ☐ projects introduced:

Introduce a comma separated list of project keys. If you write field codes (format `%(nnnnn)`) they will be replaced with its values.

- ☐ ignore links to issues in other projects

Filtering by field values: Optional boolean expression that should be satisfied by linked issues. (Syntax Specification)	1 Leave field empty for no filtering. [Line 1 / Col 1] Logical connectives: or, and and not. Alternatively you can also use <code> </code>, <code>&</code> and <code>!</code>. Comparison operators: <code>=</code>, <code>!=</code>, <code>></code>, <code>>=</code>, <code><</code> and <code><=</code>. Operators <code>~</code>, <code>!~</code>, <code>in</code>, not in, any in and none in can be used with strings, multi-valued fields and lists. Logical literals: true and false. Literal null is used with <code>=</code> and <code>!=</code> to check whether a field is initialized, e.g. <code>{00012} != null</code> checks whether Due Date is initialized. String Field Code Injector: Summary - [Text] - %{00000} ▾ Field Code for Current Issue Field Code for Linked Issues Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068} ▾ Field Code for Current Issue Field Code for Linked Issues Example 1: <code>{00012} <= ^{00012}</code> will require that linked issues have <i>Due Date</i> equal or later than current issue's <i>Due Date</i>. Example 2: <code>!{00074} ~ ^{00074} AND ^{00017} in ["Blocker", "Critical"]</code> will require that linked issues have <i>Fixed versions</i> contained in current issue's <i>Fixed versions</i> and <i>Priority</i> is <i>Blocker</i> or <i>Critical</i>.
Minimum required number of issue links:	0 Minimum number of linked issues required to satisfy selected filtering conditions (issue link type, issue type, status, resolution, project and field values).
Maximum allowed number of issue links:	1000 Maximum number of linked issues allowed to satisfy selected filtering conditions (issue link type, issue type, status, resolution, project and field values).
Allow unselected issue link types:	☒ Unselected issue link types will be ignored, i.e., they will be allowed regardless of linked issues' issue type, status, resolution, field values and number. Nevertheless, if none of the issue link types are selected, checking this option will make the condition behave as if you had selected every issue link type.
Allow unselected issue types:	☒ Linked issues in unselected issue types will be ignored, i.e., they will be allowed regardless of their status, resolution, field values and number. Nevertheless, if none of the issue types are selected, checking this option will make the condition behave as if you had selected every issue type.
Allow unselected statuses:	☐ Linked issues in unselected statuses will be ignored, i.e., they will be allowed regardless of their resolution, field values and number. Nevertheless, if none of the statuses are selected, checking this option will make the condition behave as if you had selected every status.
Allow unselected resolutions:	☒ Linked issues in unselected resolutions will be ignored, i.e., they will be allowed regardless of their field values and number. Nevertheless, if none of the resolutions are selected, checking this option will make the condition behave as if you had selected every resolution.
Allow unsatisfied condition on field values:	☐ Linked issues not satisfying filter by field values will be ignored, i.e., they will be allowed regardless of their number.
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 <i>Create Issue</i> transition.
Message to show when validation fails:	All blocker issues must be in "Resolved", "Closed" or "Done" status before current issue start progress. Field code injector: Summary - [Text] - %{00000} ▾ Field codes with format <code>%{nnnn}</code> 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

Once configured, transition **"Start Progress"** will look like this:

TO DO

Start Progress

IN PROGRESS

Screen: None - it will happen instantly

Triggers 0

Conditions 1

Validators 1

Post Functions 10

The transition requires the following criteria to be valid

[Add validator](#)

Validation on linked issue:

At least 0 and no more than 1000 issue links with the following characteristics:

- Issue link types: **is blocked by**.
- Linked issue's issue type: **any**
- Linked issue's status: **Done, Closed or Resolved**.
- Linked issue's resolution: **any**
- Linked issue belongs to **any project**.

About the rest of issue links:

- Unselected issue link types are **allowed**.
- Unselected statuses are **not allowed**.

Message to show when validation fails: **"All blocker issues must be in "Resolved", "Closed" or "Done" status before current issue start progress."**

Validator [Boolean Validator with math, date-time or text-string terms](#) with the following configuration:

Boolean expression to be evaluated:
[Line 1 / Col 119]
Syntax Specification and Examples
?

```
1 count(linkedIssues("is blocked by")) = count(filterByStatus(linkedIssues("is blocked by"), "Resolved, Closed, Done"))
```

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**.

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.

CHECK SYNTAX

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:

All blocker issues must be in "Resolved", "Closed" or "Done" status before current issue start progress.

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](#)

Boolean validation: `count(linkedIssues("is blocked by")) = count(filterByStatus(linkedIssues("is blocked by"), "Resolved, Closed, Done"))`

Once configured, transition "Start Progress" will look like this:

TO DO

Start Progress

IN PROGRESS

Screen: None - it will happen instantly

Triggers 0

Conditions 1

Validators 1

Post Functions 10

The transition requires the following criteria to be valid

Add validator

Only if the following boolean expression is true: `count(linkedIssues("is blocked by")) = count(filterByStatus(linkedIssues("is blocked by"), "Resolved, Closed, Done"))`

Message to show when validation fails: "All blocker issues must be in "Resolved", "Closed" or "Done" status before current issue start progress."

Configuration of transitions "Done", "Resolve Issue" and "Close Issue"

Use post-function [Write field on linked issues or sub-tasks](#) to write the name of the transition we want to execute into field "Issue status (delayed writing)" of blocked issue. We use a boolean expression in order to check whether the rest of blocking issues are already resolved or closed, and only when this boolean expression is satisfied the writing operation is actually done:

Target fields and Source values: ?

Select the target fields that will be set and the source values for each of them.

Target field:

Summary - [Text] Add

Add a field to be set in selected issues.

Target Field	Type of Value	Source Value	Don't Overwrite	Actions
Execute transition (delayed execution)	Parsed text (basic mode)	Start Progress		<a>Edit <a>Remove

Filtering by issue link type:

☐ is blocked by
☒ blocks
☐ is cloned by
☐ clones
☐ is duplicated by
☐ duplicates
☐ is Epic of
☐ has Epic
☐ relates to
☐ relates to

Only issues linked to current issue by selected issue link types will be written.

Write also subtasks fulfilling condition on issue type, status and project:	☐ This option only makes sense when current issue itself is not a subtask.
Write also sibling subtasks fulfilling condition on issue type, status and project:	☐ Sibling subtasks are understood as subtasks with the same parent as current issue. This option only makes sense when current issue is itself a subtask.
Filtering linked issues or subtasks by issue type:	☐  Bug ☐  Epic ☐  Improvement ☐  New Feature ☐  Story ☐ ☒ Task ☐  Sub-task Selected issue types will be written, but if you don't select any, it won't be applied any filter by issue type. In that case all the issue types will be written.
Filtering linked issues or subtasks by status:	☒  Open ☐  In Progress ☒  Reopened ☐  Resolved ☐  Closed ☒  To Do ☐  Done Selected statuses will be written, but if you don't select any, it won't be applied any filter by status. In that case issues in any status will be written.

Linked issues or subtasks belong to:	☒ any project ☐ current project ☐ any but current project
Filtering by field values: Optional boolean expression that should be satisfied by linked issues and subtasks. (Syntax Specification)	1 Leave field empty for no filtering. [Line 1 / Col 1] <u>Logical connectives:</u> or, and and not. Alternatively you can also use , & and !. <u>Comparison operators:</u> =, !=, >, < and <=. Operators ~, !=, in, not in, any in and none in can be used with strings, multi-valued fields and lists. <u>Logical literals:</u> 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. String Field Code Injector: Summary - [Text] - %{00000} ▾ Field Code for Current Issue Field Code for Linked Issues / Subtasks Numeric/Date Field Code Injector: Original estimate (minutes) - [Number] - {00068} ▾ Field Code for Current Issue Field Code for Linked Issues / Subtasks Example 1: {00012} <= ^{00012} will require that linked issues and subtasks have Due Date equal or later than current issue's Due Date. Example 2: %{00074} ~ ^{00074} AND ^{00017} in ["Blocker", "Critical"] will require that linked issues and subtasks have Fixed versions contained in current issue's Fixed versions and Priority is Blocker or Critical.
Write linked issues and subtasks recursively:	☐ Issues and subtasks transitively linked will also be written, provided they fulfill stated filtering conditions. Issues are written recursively without depth limit, but each selected issue is written only once.
Additional options:	☐ Enable email notifications on issues to be written, according to applicable notification scheme. ☐ Update issue immediately after field writing. A specific entry will be created in issue history for each field writing.
Conditional execution: Optional boolean expression that should be satisfied in order to actually execute the post-function. (Syntax Specification)	1 Leave the field empty for executing the post-function unconditionally. Collection of Examples [Line 1 / Col 1] <u>Logical connectives:</u> and, or and not. Alternatively you can also use &, and !. <u>Comparison operators:</u> =, !=, >, < and <=. Operators in, not in, any in, none in, ~ and != can be used with strings, multi-valued fields and lists. <u>Logical literals:</u> 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. String Field Code Injector: Numeric/Date Field Code Injector: Summary - [Text] - %{00000} ▾ Original estimate (minutes) - [Number] - {00068} ▾

i If you are using Jira 7.0 or higher with versions of [Jira Workflow Toolbox](#) older than **2.2.8**, you should input the following boolean expression in parameter Filtering by field values:

```
count(filterByStatus(linkedIssues("is blocked by"), "Closed, Resolved, Done")) = count(linkedIssues("is blocked by")) - 1
```

This way it prevents the transition execution whenever the validation is not satisfied.

Once configured, transition transitions **"Resolve Issue"**, **"Close Issue"** and **"Done"** will look like this:

The following will be processed after the transition occurs

[Add post function](#)

1. Fields in the following issues will be written:

Inward issue link types: **none**

Outward issue link types: **blocks**.

Subtasks won't be written.

Sibling subtasks won't be written.

Issue types: **any**

Statuses: **To Do, Open and Reopened**.

Linked issues or subtasks may belong to **any** project.

Target fields and Source values:

Target Field	Type of Value	Source Value	Don't Overwrite
Execute transition (delayed execution)	Parsed text (basic mode)	Start Progress	

This feature will be run as user in field **Current user**.

by JWT

Other examples of that functions

Write field on linked issues or sub-tasks

Page: [Add and remove a single or a set of items from multi valued fields](#)

Page: [Automatically become watcher of every issue blocking an issue assigned to you](#)

Page: [Automatically close resolved sub-tasks when parent issue is closed](#)

Page: [Automatically resolve an epic when all its stories are resolved](#)

Page: [Compose dynamic text by inserting field values in a text template](#)

Page: [Copy "Due date" into a date type custom field in a linked issue if it's greater than current issue's "Due date"](#)

Page: [Copy attachments from one issue to another](#)

Page: [Create a comment in sub-tasks when parent transitions](#)

Page: [Creating a Jira Service Desk internal comment](#)

Page: [Creating a Jira Service Desk internal comment on linked issues](#)

Page: [Execute transition in epic](#)

Page: [Make linked issues, sub-tasks and JQL selected issues progress through its workflows](#)

Page: [Moving sub-tasks to "Open" status when parent issue moves to "In Progress"](#)

Page: [Sum sub-task's "Time Spent" \(work logs\) and add it to a certain linked issue](#)

Page: [Transition sub-tasks when parent is transitioned](#)

Update issue fields

Page: [Add and remove a single or a set of items from multi valued fields](#)

Page: [Compose dynamic text by inserting field values in a text template](#)

Page: [Creating a Jira Service Desk internal comment](#)

Page: [Creating a Jira Service Desk internal comment on linked issues](#)

Page: [Make linked issues, sub-tasks and JQL selected issues progress through its workflows](#)

Page: [Moving sub-tasks to "Open" status when parent issue moves to "In Progress"](#)

Page: [Parse Email addresses to watchers list](#)

Page: [Set priority for issues that have been in a certain status for longer than 24 hours](#)

Page: [Transition linked issues in currently active sprint](#)

Page: [Transition only a sub-task among several ones](#)

Page: [Using project properties to calculate custom sequence numbers](#)

Related Usage Examples

- [Prevent transitioning when there is a blocking issue](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
 - [sub-task](#)
 - [transition](#)
- [Make linked issues, sub-tasks and JQL selected issues progress through its workflows](#)
 - [example](#)
 - [condition](#)
 - [validator](#)
 - [post-function](#)
 - [issue-links](#)
 - [sub-task](#)
 - [transition](#)
- [Add and remove a single or a set of items from multi valued fields](#)
 - [example](#)
 - [post-function](#)
 - [custom-field](#)
 - [issue-links](#)
 - [sub-task](#)
- [Sum sub-task's "Time Spent" \(work logs\) and add it to a certain linked issue](#)
 - [example](#)
 - [post-function](#)
 - [issue-links](#)
 - [sub-task](#)
 - [work-log](#)
- [Sum "Time Spent" in all sub-tasks of issues linked with issue link types "LinkA", "LinkB", "LinkC"](#)
 - [example](#)
 - [post-function](#)
 - [issue-links](#)
 - [sub-task](#)
 - [work-log](#)
- [Block an epic's transition depending on linked issues status and due date](#)
 - [example](#)
 - [validator](#)
 - [issue-links](#)
 - [transition](#)

Page: Writing a comment to blocked issues when blocking issues are resolved

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

Page: Automatically resolve an epic when all its stories are resolved

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

Boolean Validator with math, date-time or text-string terms

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

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

- Prevent issue from moving forward if it's dependent on non-accepted tickets
 - example
 - validator
 - issue-links
 - transition
- Enforce linked issues in a specific project to be "Closed" before closing issue
 - example
 - validator
 - issue-links
 - transition
- Block or hide a transition for an issue depending on its issue links
 - example
 - validator
 - issue-links
 - transition
- Prevent issue from being "Closed" if blocking issues aren't yet closed
 - example
 - validator
 - issue-links
 - transition
- Prevent issue from being closed if it has links of type "is blocked by" to open issues
 - example
 - condition
 - validator
 - issue-links
 - transition
- Transition linked issues in currently active sprint
 - example
 - post-function
 - issue-links
 - transition
- Validation on sibling sub-tasks depending on issue type and status
 - example
 - validator
 - sub-task
 - transition
- Block a transition until all sub-tasks have certain fields populated
 - example
 - condition
 - validator
 - sub-task
 - transition
- Transition sub-tasks when parent is transitioned
 - example
 - post-function
 - sub-task
 - transition
 - outdated