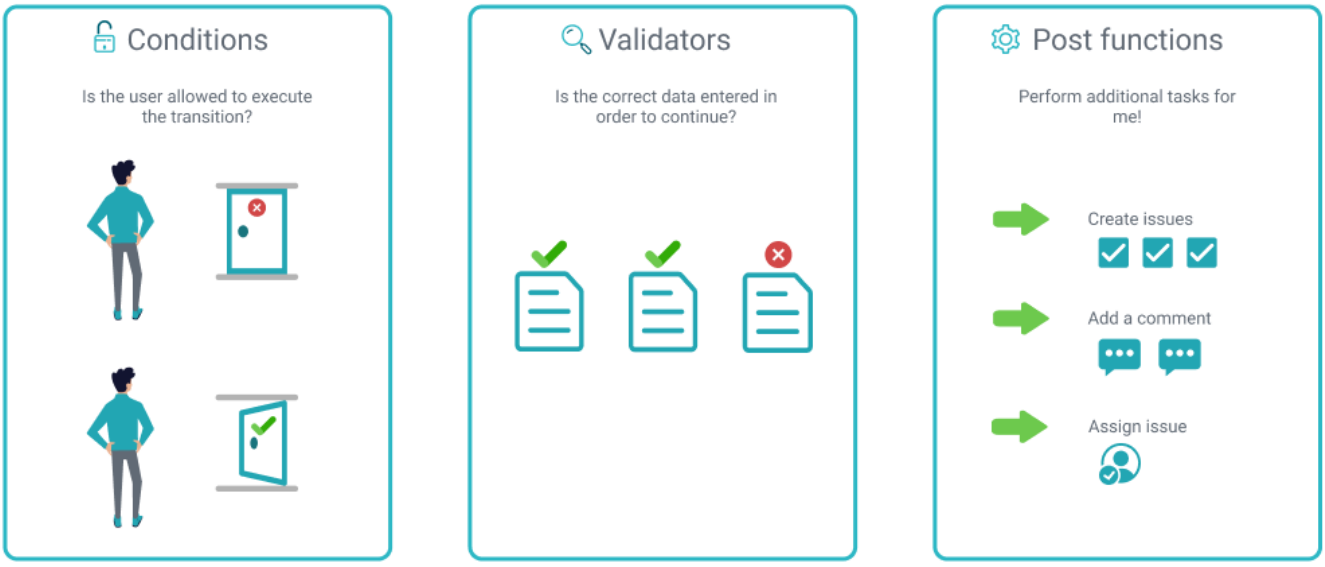
Get started with workflow functions

The way tasks and processes are managed in Jira is through **workflows**. A workflow maps out the **statuses** an issue can go through and the available **transitions** between the statuses that together define your entire process.

You can edit the overall workflow used in a project, or modify the way particular issue types are handled in the workflow. **But**: At some point, you might end up stuck in the middle of nowhere because of **feature limitations**.

At this point, Jira Workflow Toolbox for Jira comes in handy!

JWT for Jira enhances the way you configure workflows. It **extends the native functionality** by offering custom **conditions**, **validators**, and **post functions**.



Conditions

Conditions are used to control the transitions available to a user. If a condition fails, the user will **not see the transition button** on the Jira issue view, and so will not be able to execute the transition.

JWT offers the following conditions:

- Compare two values condition
- Condition based on cascading select list value
- Condition based on JQL query
- Condition based on regular expression
- Condition on JWT project property
- Condition on linked issues
- Condition on sub-tasks
- Except assignee
- Except reporter
- Except users in a field
- Fields required
- Hide transition from user (bulk operation only)
- Hide transition from user (JWT function only)

- Logical condition
- Only users in a field
- User is not in project role
- Users are/aren't in project role (condition)

To get more familiar with conditions, we recommend to take a look at our use case library:

Condition use cases and examples

Use case	JWT feature	Parser functions	Label
Hide transition to a previous status			
Set a condition in a global transition which only applies in a certain status			
Only user specified in project property is allowed to execute transition		projectProperty()	
All sub-tasks in the Closed status must have a specific resolution			
There must be at least one related Service Management request			
Only users in a project role can execute a transition			
Only watchers can execute a transition			
A developer must not execute the transition			
Resolution must be empty			
Make the description required			
Make the assignee required			
Hide transition from issue creator			
An issue must have at least 3 resolved Test Cases			
All sub-tasks with a resolution of Done must be in a specific status			
All sub-tasks must be resolved			STAFF PICK
All sub-tasks must be Done or Closed			

Specific sub-tasks must be resolved	
All blocking issues must be resolved	
Prevent users from selecting specific fix versions	
Assignee may only have a restricted number of assigned issues	
Current user must be reporter	

Validators

Validators are used to guarantee **accuracy of existing issue data** or data entered on a transition screen before a transition is performed.

JWT offers the following validators:

- Compare two values validator
- Fields required or changed
- Logical validator
- Users are/aren't in project role (validator)
- Validation based on JQL query
- Validation based on regular expression
- Validation of JWT project property
- Validation of linked issues
- Validation of sub-tasks

To get more familiar with validators, we recommend to take a look at our use case library:

Validator use cases and examples

Use case	JWT feature	Parser functions	Label
Prevent a transition depending of the number of components		numberOfSelectedItems()	
Prevent a closed issue from being reopened after resting 1 week closed		isInRole()	
Prevent issue creation if description contains less than 100 characters		replaceAll() length()	
Evaluate Assets objects		findPattern() findReplaceAll() replaceAll() toStringList()	

Block a transition until all sub-tasks have certain fields populated		count() filterByPredicate() subtasks()	STAFF PICK
Validate a Select List (cascading) custom field			
Prevent external users from creating issues		matches() toString() textOnStringList() toStringList() userDisplayName()	
Validate only issue links created in transition screen		count() transitionLinkedIssues() filterByProject() filterByIssueType()	
Prevent setting due dates outside business hours		withinCalendar()	STAFF PICK
Prevent the creation of sub-tasks unless the parent issue is in a certain status			
Halt a transition if an element is not contained in a list		toStringList()	
Prevent transitioning when there is a blocking issue			
Prevent creation of issues with a duplicate summary			
Validation based on the value of a date type project property		stringToDate() projectProperty()	
Restrict issue creation per issue type and project role		isInRole()	
Reject duplicated file names in attachments		count() toStringList() distinct()	
Prevent issue creation with the same field value (Boolean)		count() issuesFromJQL()	
Ensure that all issues linked with a certain issue link type have "Due Date" field set		count() linkedIssues() fieldValue()	
Block an Epic's transition until all the issues under that Epic are resolved		count() issuesUnderEpic() filterByResolution()	

Block an Epic's transition depending on linked issues status and due date		count() filterByPredicate() linkedIssues()
Make "Time spent" field required		
Close parent issue only when all sub-tasks are closed		
Prevent issue creation with the same field value		STAFF PICK
Ensure that an issue has at least one attachment		STAFF PICK
Proceed with a task only when all sub-tasks are completed		
Prevent issue from being "Closed" if blocking issues are not closed yet.		
Restrict the issue creation with specific issue types to certain project roles		

Post functions

Post functions are used to perform **additional processing** and help to automate tasks after a transition is performed.

JWT offers the following post functions:

- Add comment
- Add or disable option in (multi-) select list, radio button, or checkbox field
- Add or disable option in cascading select list field
- Add or remove watchers
- Assign to project role
- Copy cascading select list value
- Copy excerpted value
- Copy field values from linked issues or subtasks
- Copy field values from multiple issues
- Copy JWT project property
- Copy JWT user property
- Create issue
- Create issue link
- Delete issue link
- Execute remote action
- Format field value
- Log work
- Move issue
- Regular expression renderer
- Send email
- Set or create JWT project property
- Set or create JWT user property
- Transition issue
- Update field based on rules
- Update linked issue or sub-task

- Update or copy field values

To get more familiar with post functions, we recommend to take a look at our use case library:

Post function use cases and examples

Use case	JWT feature	Parser functions	Label
Add a hyperlink to an issue in an email			
Add comment with the request participants' display name		toString() textOnStringList() toStringList() userDisplayName()	
Alert the assignee of important issues			
Get attributes of Assets objects			
Set fix version based on its start and release date		toString() textOnStringList() unreleasedVersions() startDates() releaseDates() first()	
Translate the description		replaceAll() wikiToHTML() htmlToTxt()	
Get Hubspot contact information			
Obtain a value from an Elements Connect custom field		findPattern() toString()	
Create a sub-task for each component		toStringList()	
Create a personal space for a new employee			
Create Confluence page with links to issues			
Clone issue in external instance			
Read the information from a Trello card			
Get Checklist			
Set a Date Picker field to the nth day of the month		dayOfTheMonth() lastDayOfTheMonth() addDays()	
Create child issue within Advanced Roadmaps hierarchy			

Send email with the URL of the attachments included in the description		replaceAll() toString() attachmentUrls() findPattern()
Set Checklist		
Link issue to issue keys in its description		findPattern() toString()
Create issue under epic		toString() epic()
Set assignee based on a former assignee		previousValue()
Create a sub-task for every sub-task closed		findReplaceFirst() first() findPattern() toString() sum() toNumber()
Add watchers ignoring inactive users		usersInGroup() isActive() toString() filterByPredicate()
Copy formatted description of issue into email		wikiToHTML()
Set reporter as assignee if a User Picker field is empty		
Keep the status of an issue and its linked issues in sync		
Add user to field depending on selected options		
Set a date on the same week day on alternate weeks		addDays() nextDayOfTheWeek() modulus() weekOfTheYear() datePart()
Set the next fix version		floor() toNumber() substring() length()
Create a component to group issues related to UI design		
Automatically create a version when starting the release		
Create sub-tasks depending on selected values in custom field		toStringList() STAFF PICK
Add or remove request participants		
Retrieve the assets of an issue in Jira cloud		

Clone epic, tasks and sub-tasks		toString() toStringList() issueKeysToIssueList() replaceFirst() first() fieldValue() textOnStringList() findModify() subtasks() issuesUnderEpic()
Set the assignee of an external issue same as the transitioned issue		
Notify the reporter of an issue about its status by a Telegram message		
Set assignee depending on issue type		issueType()
Create an issue with a custom summary		previousValue()
Inline images in a generated email		replaceAll()
Create a sub-task for every user in a field and assign them		getMatchingValue() toStringList()
Automatically log work on a Jira issue		dateTimeToString()
Create an overview page for a software release		
Create a new employee account during an onboarding process		
Automatically link an issue to an external one		
Transition an external Jira ticket based on the linked internal one		
Link a Jira issue with the corresponding release ticket		
Create an external project for a new employee during an onboarding process		
Create a comment on an external Jira ticket		
Assign issue to current user		
Add a table with the elements of a text to an email		toString() textOnStringList() toStringList()
Create several sub-tasks depending on the component		
Create a sub-task for every recurring deadline within a task		addMonths() getMatchingValue() dateToString()

Add days skipping weekends and holidays to a Date Picker field		addTime()
Create several issues combining fields		nthElement() toStringList() modulus() count() ceil()
Match several values of a list		toString() distinct() filterByPredicate()
Change the assignee to the next evaluator		first() toStringList()
Prioritize the issues globally		indexOf() previousValue() issuesFromJQL()
Shorten the summary to a maximum number of characters		substring() length()
Send email after transitioning to specific status		
Automatically fill an insight custom field after a transition		
Create an issue in the current project		STAFF PICK
Create issues randomly		modulus() round() random()
Automatically log work spent in a specific status		timeDifference()
Triage issues created by email		toUpperCase() toString() findPattern() isInGroup() STAFF PICK
Set issue security level depending on reporter		issueSecurityLevel() STAFF PICK
Set Due Date with latest value among sub-tasks		fieldValue() max() siblingSubtasks()
Set assignee based on priority		priority()
Log absence time on another issue		stringToDate()

Keep track of important status updates		dateTimeToString()
Create three issues with individual summaries		getMatchingValue()
Alert the reporter		
Create an internal Service Management comment on linked issues		
Keep parent issue's priority in sync		
Assign important issues to the project lead		
Add attachments from current issue to cloning issues		
Add assignee as watcher to every blocking issue		
Keep the status of parents and sub-tasks in sync (post function use case)		
Fast-track transition issues assigned to the project lead		STAFF PICK
Reopen parent issue, if a sub-task is reopened		
Start Progress on parent, if sub-tasks are started		STAFF PICK
Move an issue to another issue type		
Move an issue to another project		
Add percentaged profit margin when closing issue		
Format the issue's summary according to specified rules		STAFF PICK
Remove blocking links when blocking issue is closed		
Set specific default assignee if not set		

Copy latest due date from sub-tasks	
Copy highest priority from linked issues	
Automatically assign a tester during your development process	
Extract an email address from the last comment	
Extract the issue priority from the summary	
Set a watcher based on custom field value	
Add watchers based on issue type	
Add internal Jira Service Management comment	

If you still have questions, feel free to refer to our support team.