Unified workflow using global transitions

Last Modified on 03/25/2024 12:14 pm EDT

Introduction

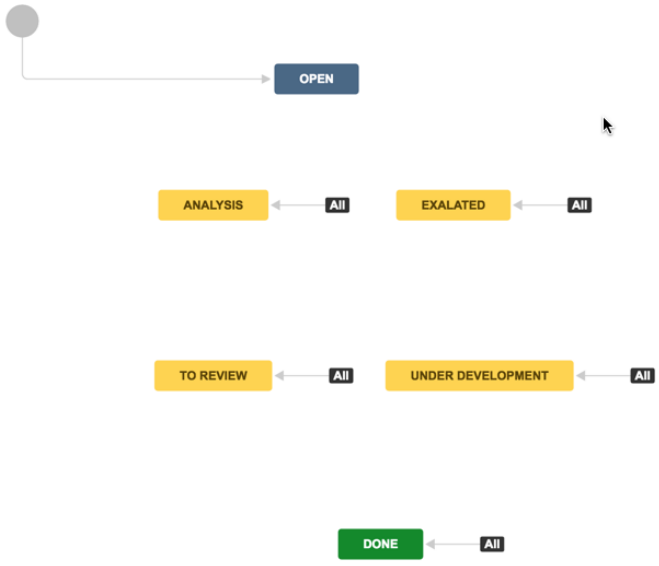
Warning: Despite our best efforts, code can change without notice due to a variety of factors. If you encounter an issue in any of the code shown here and find that a specific block of code is not correct, or is causing errors, please check with the Community to find an updated version.

You need to map two workflows. Issues get raised at the customer site, exalated to the software vendor, who handles the issue and resolves it.

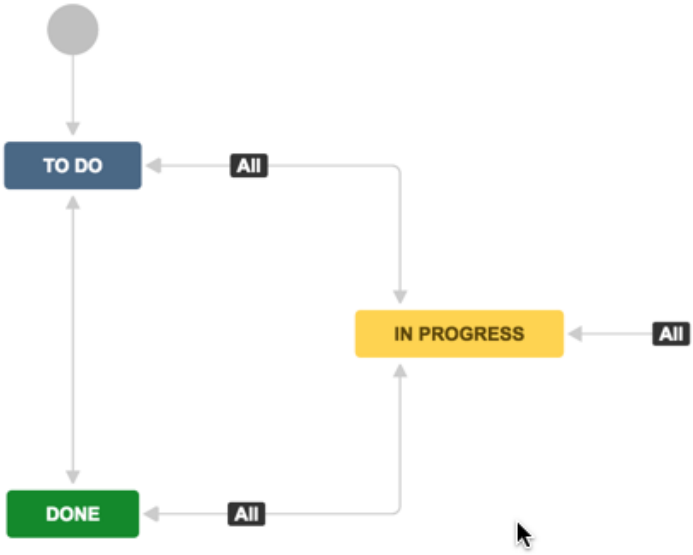
The workflows on either end are using global transitions, which allows you to progress an issue to a particular status, whatever the original status is.

For this example, we have set up the following workflows, one on the customer side, the other on the software vendor end.

Customer workflow



Software Vendor workflow



The intended status mapping should be:

Customer Status	Software Vendor Status
Analysis	Done
Exalated	To Do
Under Development	In Progress
To Review	Done
Done	Done

How does this work?

Whenever the software vendor changes the status of the synced issue, the status change is applied by looking up the corresponding target status in the statusMap, and assigning it to the **issue.doTransition**. In case the remote status is not set (because it is not provided by the software vendor), or if the mapping cannot be found, nothing happens, except for a comment being added to the issue.

To implement this mapping, add the snippets below to the **Change Processor**.

This configuration works if every transition in the workflow is a global transition.

On the customer side

```
1 // the transitionmap maps remote statuses to transitions which needs to be activated locally
2 def transitionMap = [
3   "To Do":"Exalated",
4   "In Progress":"Under development",
5   "Done":"To Review"
6 ]
7
8  /* transition the issue to the corresponding status.
9  ** If the remote status is null or the status is not found in the map,
10 ** add a comment to the issue such that the configuration can be adapted.
11 */
12
13 if (replica.status == null ||
14    replica.status.name == null ||
15    transitionMap.get(replica.status.name) == null) {
16  issue.comments = commentHelper.addComment("Remote status is unknown or cannot be mapped - can't handle
it", issue.comments)
17 } else {
18  workflowHelper.transition(issue, transitionMap.get(replica.status.name))
19 }
```

On the software vendor side

The Connection on the software vendor side is similar, except for the mapping

```
1 // the transitionmap maps remote statuses to transitions which needs to be activated locally
2 def transitionMap = [
3   "Exalated":"To Do",
4   "Analysis":"Done",
5   "Under development": "In Progress",
6   "To Review":"Done",
7   "Done":"Done"
8 ]
9
10 // transition the issue to the corresponding status.
11 // If the remote status is null or the status is not found in the map, progress issue to in review
12 if (replica.status == null ||
13     replica.status.name == null ||
14     transitionMap.get(replica.status.name) == null) {
15   issue.comments = commentHelper.addComment("Remote status is unknown - can't handle it", issue.comments)
16 } else {
17   workflowHelper.transition(issue, transitionMap.get(replica.status.name))
18 }
```

Need help?

Just send an email to support@exalate.com or book a support call.

ON THIS PAGE
Introduction

Join our Community
Visit our Service Desk
Find a Partner