

# Multiple Workflow Orchestrations

Last Modified on 02/10/2026 6:12 am EST

## 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.

Getting a workflow orchestration between 2 different workflows defined and implemented is already a challenge. The fact is that Jira allows having multiple workflows per project/issue type, and arranging that dance is a real challenge.

We assume that you already have a good understanding of the power of the `StatusSync.groovy` externalized script - which acts as your personal transition agent.

## Defining the Orchestration

To meet this challenge, it is very important to do a throughout analysis of the different permutations.

1. For each source workflow
  - 1.1. For each status
    - 1.1.1. For each target workflow
      - 1.1.1.1. For each status
        - 1.1.1.1.1. What transition needs to be made

This can grow exponentially, as the semantics of each transition can be different.

## The Status Matrix

The status - matrix is a first approach where you define for each source issue status the target issue status.

Transition	Open	In Progress	In Review	Closed
Todo	x			
Under development		x		
Done			x	

This results in a transition map

```
def transitionMap = {
  "Todo" : "Open",
  "Under development" : "In Progress",
  "Done" : "In Review"
}

statusSync.receive ( transitionMap, ...)
```

Seems simple, doesn't it?

Of course it is never that simple, because the source workflows can have conflicting target statuses.

Transition	Open	In Progress	In Review	Closed
Todo	x			
Under development		x		
Done			x if target issue is a bug	x if target issue is a task

How do you solve these apparently conflicting specifications? The difference is that the target status is different depending on the target status.

This translates into the following snippet:

```
//Define the different transitionMaps

def transitionMapBug = {
  "Todo" : "Open",
  "Under development" : "In Progress",
  "Done" : "In Review"
}

def transitionMapTask = {
  "Todo" : "Open",
  "Under development" : "In Progress",
  "Done" : "Closed"
}

if (issue.issueType?.name == "Bug") {
  statusSync.receive ( transitionMapBug, ...)
} else {
  statusSync.receive ( transitionMapTask, ...)
}
```

Depending on the issue type, one or another map is being used.

## Using Resolutions

In many cases, it is sufficient to know if an issue is being resolved or not.

- When a remote issue is not yet resolved, the local issue should be 'waiting for development'
- When a remote issue is resolved, the local issue should be moved to status 'to review'

The code snippet is as follows:

```
/*
** Two transitions have been configured on all workflows
**
** - autoResolve - an any to resolve transition
** - autoOpen - an any to open transition
** Both transitions are hidden for the user.
** the autoResolve sets the local resolution. and the autoOpen clears the local resolution.
*/
// the following line will transition an issue in case that the resolution of the remote issue changed

if (previous.resolution?.name != replica.resolution?.name) {
    // if the replica.resolution changed compared to the previous sync, transition depending on the value of resolution
    // if the value of the replica.resolution is null - use 'autoOpen', else 'autoResolve'

    workflowHelper.transition(replica.resolution ? "autoResolve" : "autoOpen", issue)
}
```

### ON THIS PAGE

Product

Introduction

[Release History](#) ?

[Defining the Orchestration](#)

[Glossary](#) ?

[The Status Matrix](#)

[Security](#) ?

[Using Resolutions](#)

[Pricing and Licensing](#) ?

### Resources

[Subscribe for a weekly Exalate hack](#) ?

[Academy](#) ?

[Blog](#) ?

[YouTube Channel](#) ?

[Ebooks](#) ?

### Still need help?

[Join our Community](#) ?

[Visit our Service Desk](#) ?

[Find a Partner](#) ?