How to Sync Status using External Scripts in Jira On-premise

Last Modified on 01/15/2026 3:47 pm EST

This page shows how to synchronize issue statuses in Jira on-premise bi-directionally. You can map workflows between two JIRA Instances or set the transition manually.

JIRA requires an issue status update by progressing this issue through the different workflow steps. You can achieve it in different ways:

- Manually model every step
- Using any one type of transitions
- Automatically progress to the correct status using advanced groovy scripting

Exalate provides different approaches to configure status synchronization when workflow transitions are global or there's only one transition to get to the right status:

- control the transition applied to your local issue
- map statuses between instances

Configuration

Let's consider you already have the Connection configured between two JIRA Instances.

Now you need to **configure status synchronization**

Map Statuses

EXALATE FROM 4.7.3

Outgoing sync

To send the status use the code below

```
replica.status = issue.status
```

Destination side

Incoming sync

```
def statusMap = ["Done": "Resolved", "In Progress": "In Action"] // ["remote status name": "local status name"]
def remoteStatusName = replica.status.name
issue.setStatus(statusMap[remoteStatusName] ?: remoteStatusName)
```

if the status is the same on both sides you can just do:

```
issue.status = replica.status
```

EXALATE FROM 4.7.2 AND LOWER

1. Create files from the Exalate public repository

Note: We store external scripts for Jira Server in a public repository. Copy the code from the repositories below and create ***.groovy*** files. You must keep the file name as below.

- Status.groovy

2. Upload the files to the ***$JIRA_HOME/scripts*** directory

3. Configure Sync Rules with the scripts provided below:

Add the snippets below to the end of the Sync Rules.

Source side

Outgoing sync

```
Status.send()
```

Destination side

Incoming sync

Add the code to a new line at the bottom of the incoming sync rules block.

1. If statuses are the same on both sides, use this code:

```
Status.receive()
```

2. If the statuses are different, use this code with your status mapping

```
Status.receive(useRemoteStatusByDefault = true, workflowMapping = [
 "Remote Status A" : "Local Status A",
 "Remote Status B" : "Local Status B",
 "Remote Status C" : "Local Status C",
 ], resolutionMapping = [:])
```

Include Status.receive(..) at the end on the incoming processor. Any other changes coded after it gets ignored.

Status.receive has the following parameters:

```
//default parameters
Status.receive(useRemoteStatusByDefault = true, workflowMapping = [:], resolutionMapping = [:])
```

useRemoteStatusByDefault = true

Use the remote status by default.

```
Status.receive(useRemoteStatusByDefault = true, workflowMapping = [:], resolutionMapping = [:]) // Exalate will look for a local status with the same name as the incoming status
 or
Status.receive(useRemoteStatusByDefault = false, workflowMapping = [:], resolutionMapping = [:])
```

workflowMapping = [:]

Defines the status mapping as on the example below

```
Status.receive(
 useRemoteStatusByDefault = true,
 workflowMapping = [
   "Remote Status A" : "Local Status A",
   "Remote Status B" : "Local Status B",
   "Remote Status C" : "Local Status C",
 ],
 resolutionMapping = [:]
)
```

resolutionMapping = [:]

Defines resolution mapping as on the example below

```
Status.receive(
 useRemoteStatusByDefault = false,
 workflowMapping = [:],
 resolutionMapping = ["Remote Resolution A": "Local Resolution A"
])
```

Control the Transition applied to your Local Issue

Source side

Outgoing sync

To send the status use the code below

```
replica.status = issue.status
```

Destination side

Incoming sync

workflowHelper.transition method allows you to set a local transition based on the remote issue status.

```
//if the local issue status is 'In Progress' and the remote issue status is 'Resolved' use 'Resolve' transition

if (issue.status?.name == "In Progress" && replica.status.name == "Resolved") {
   workflowHelper.transition(issue, "Resolve")
}
```

In case you need to set different transitions depending on the remote status you need to add the script for each transition separately. Check the example below:

```
//if the local issue status is 'In Progress' and the remote issue status is 'Resolved' use 'Resolve' transition

if (issue.status?.name == "In Progress" && replica.status.name == "Resolved") {
   workflowHelper.transition(issue, "Resolve")
}

//if the local issue status is 'Done' and the remote issue status is 'Resolved' use 'Close' transition

if (issue.status?.name == "Done" && replica.status.name == "Resolved") {
   workflowHelper.transition(issue, "Close")
}
```