How to Sync Status using External Scripts in Jira Cloud

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

Exalate provides different ways to configure status synchronization:

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

Configuration

Let's consider you already have the Connection configured between the Instances. Now you need to **configure Sync Rules** with the scripts provided below:

Source side

Outgoing sync

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

Destination side

Control the transition applied to your local issue

Use this approach if want to control the transition applied to your local issue. You can apply only one transition per synchronization using this approach. The workflowHelper.transition method allows you to set a local transition based on the remote issue status.

Incoming sync

```
//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")
}
```

Map statuses

Use this approach when you want to have statuses synced automatically based on the configured status mapping.

Incoming sync

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

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

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

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

- If you want to set a default status directly in the rules use this code

```
issue.setStatus("Done")
```

- If the remote side has not set the resolution, change the local issue status to Done

```
if(replica.resolution != null){
  issue.setStatus("Done")
}
```