

How to Sync Versions in Jira Cloud

Last Modified on 04/09/2024 10:15 am EDT

This article shows how to sync fix versions between Jira instances.

Jira version is an object that includes the following elements:

- version name
- version start date
- version release date
- version description

Basic versions synchronization involves receiving versions from a remote Jira instance. Usually, these versions don't exist on your local Jira. Exalate provides a way to create versions from the scripts to handle this situation. To create a new version on your system when necessary use [nodeHelper.createVersion](#). The example below shows how you can set up such behavior.

Source Side

Outgoing sync

send fix versions

```
//send the fix versions set on a synced issue
replica.fixVersions = issue.fixVersions
replica.affectedVersions = issue.affectedVersions
```

Destination Side

Incoming sync

Create the versions that do not exist on your side.

```
// for the create processor, be sure that the project is set to the issue variable before running the following code
issue.projectKey = "Foo" //Included only on create processor
...

// assign fix versions from JIRA A to JIRA B
issue.fixVersions = replica
.fixVersions
// ensure that all the fixVersions are available on B
.collect { v -> nodeHelper.createVersion(issue, v.name, v.description) }
// assign affected versions from JIRA A to JIRA B
issue.affectedVersions = replica
.affectedVersions
.collect { v -> nodeHelper.createVersion(issue, v.name, v.description) }
```

If you do not want exalate to create new versions but just use existing ones that match the other side versions:

```
// for the create processor, be sure that the project is set to the issue variable before running the following code
issue.projectKey = "Foo" //Included only on create processor
...

// assign fix versions from JIRA A to JIRA B
def project = nodeHelper.getProject(issue.projectKey)
issue.fixVersions = replica
    .fixVersions
// ensure that all the fixVersions are available on B
    .collect { v -> nodeHelper.getVersion(v.name, project) }
    .findAll{it != null}
// assign affected versions from JIRA A to JIRA B
issue.affectedVersions = replica
    .affectedVersions
    .collect { v -> nodeHelper.getVersion(v.name, project) }
    .findAll{it != null}
```

If you want to synchronize version start date, release date and description you can use the external script [versions.groovy](#), which has been developed specifically for such cases.

Product

[About Us](#)

ON THIS PAGE

[Release History](#)

[Glossary](#)

[Source Side](#)

[API Reference](#)

[Destination Side](#)

[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](#)