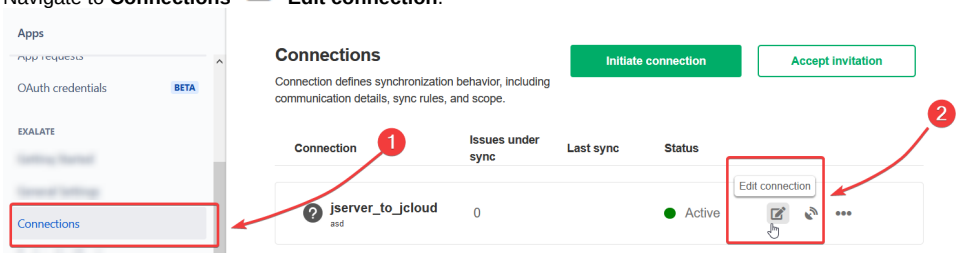
How to use scripts in Visual connections

This article shows examples of using scripts in Visual connections. In Visual mode, you can specify the mapping for various fields. You can also add scripts to sync specific custom fields, or fields that are not available in the visual mapping.

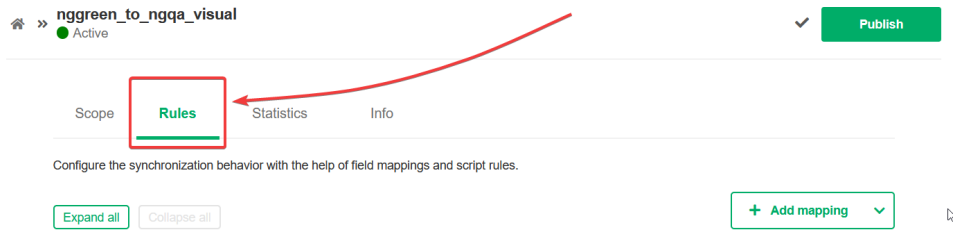
How to add a script to a Visual connection

To add a script to a Visual connection:

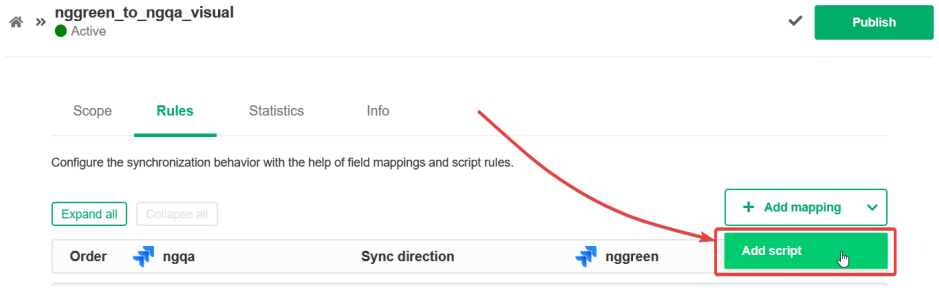
1. Navigate to **Connections** **Edit connection**.



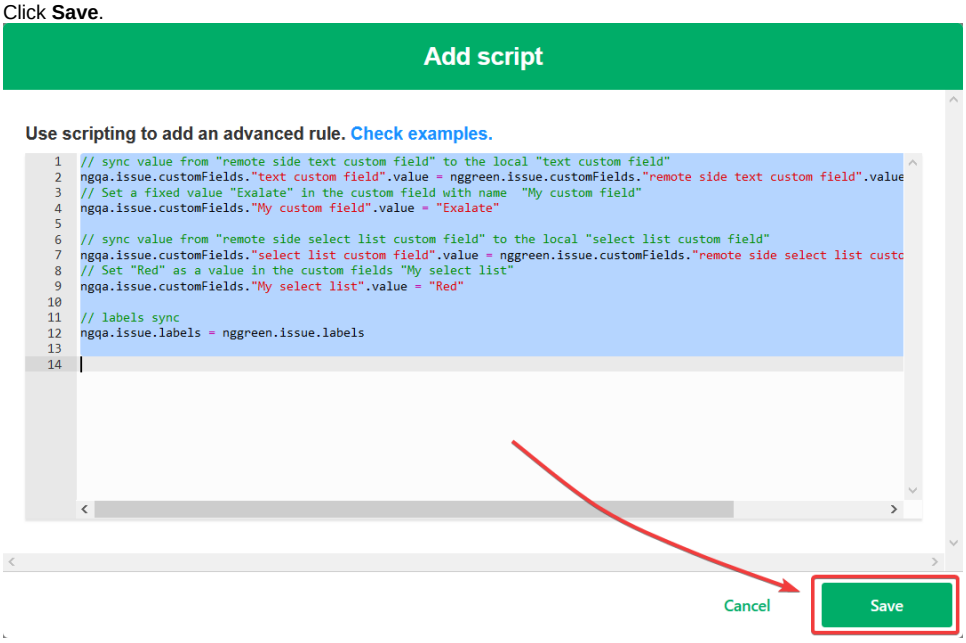
2. Select the **Rules** tab.



3. Click **Add script**.



4. Click **Save**.



Script examples

Syncing simple custom fields

This example shows how to sync custom field `user` from instance `your_instance_shortname` to custom field `name` on instance `remote_instance_shortname` in a Visual connection:

```
remote_instance_shortname.issue."name" = your_instance_shortname.issue."user"
```

`your_instance_shortname` is the source instance and `remote_instance_shortname` is the destination instance.

Use the following script to sync data in both directions:

```
your_instance_shortname.issue."user" = remote_instance_shortname.issue."name"
remote_instance_shortname.issue.issue."name" = your_instance_shortname.issue."user"
```

Syncing versions

For more complex cases, it is recommended to use the same scripts as for Script connections, but in `if` clauses.

With the `executionInstanceName` variable you can define what instance is affected by the script.

This example shows how to sync versions from `your_instance_shortname` to `remote_instance_shortname` in a Visual connection:

```
if (executionInstanceName == "remote_instance_shortname") {
  issue.projectKey = "name" //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) }
}
```

For Script connections this script would look like this:

```
issue.projectKey = "name" //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) }
```

For more information, check How to synchronize versions in Jira Cloud and How to synchronize versions in Jira on-premise.

Syncing comments that have no group or role level assign

With this script you can sync comments from one instance to another.

```
if (executionInstanceName == "your_instance_shortname") {
        replica.comments = commentHelper.filterLocal(issue.comments)
}
```

For Script connections this script would look like this:

```
replica.comments = commentHelper.filterLocal(issue.comments)
```

For more information, check these articles:

How to sync comments in Azure DevOps

How to sync comments in Zendesk

How to sync comments in ServiceNow

How to sync comments in Jira Cloud

> *Have more questions? Ask the community*

See also

How to synchronize versions in Jira Cloud

How to synchronize versions in Jira on-premise

How to sync comments in Azure DevOps

How to sync comments in Zendesk

How to sync comments in ServiceNow

How to sync comments in Jira Cloud

Back to General articles