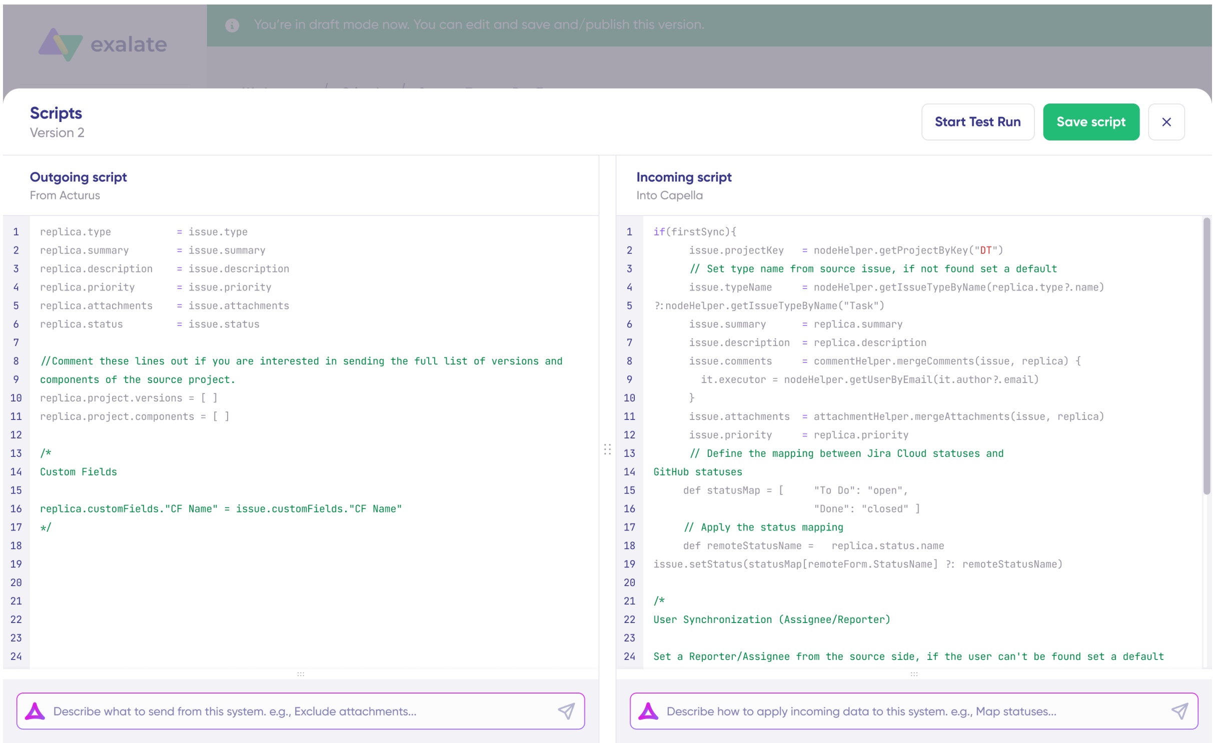
TestRun Configuration

Last Modified on 12/17/2025 10:49 am EST

TestRun allows you to test your scripts without creating actual entities on the destination side. This feature lets you preview and compare the replica before making changes in production, ensuring your configuration works as expected.

How to Use TestRun

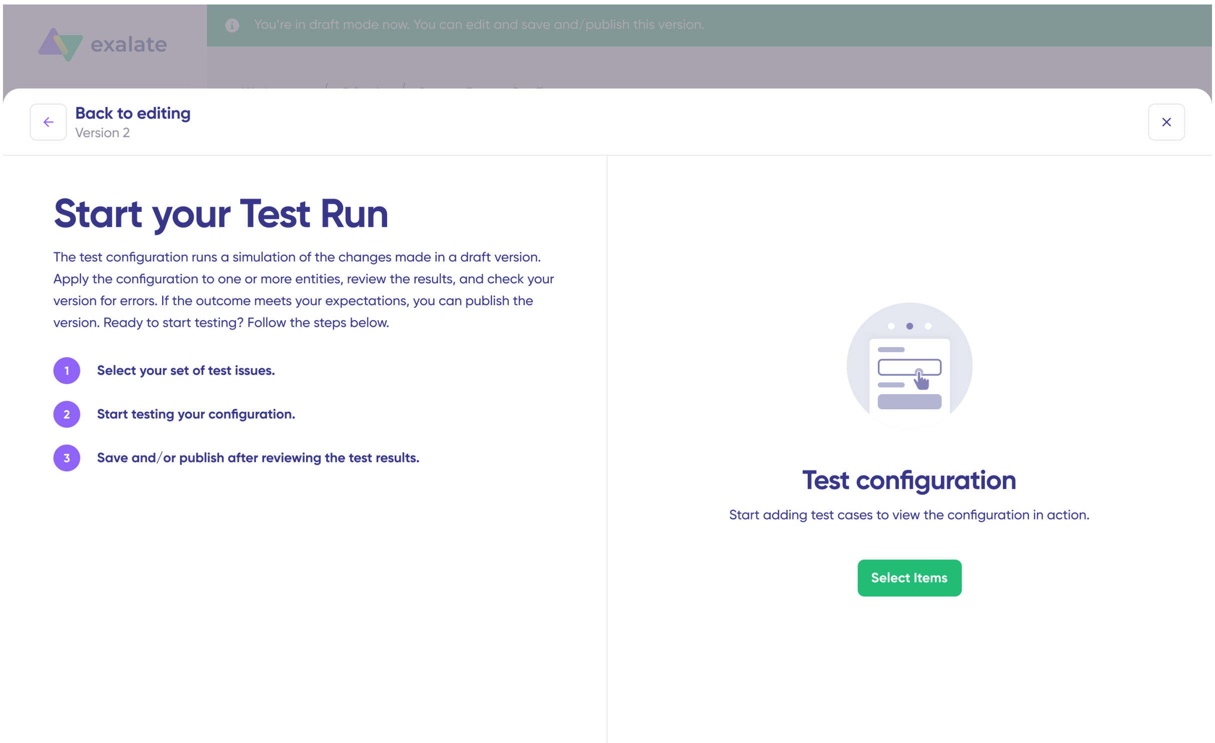
1. Create a **New Version** or click **Open latest Draft.**
2. Click **Edit** to start modifying your configuration.
3. Make any necessary changes to your sync rules.
4. Click **Start TestRun** to begin the simulation.


exalate
You're in draft mode now. You can edit and save and/publish this version.
Scripts
Version 2
Start Test Run
Save script
×
Outgoing script
From Acturus
Incoming script
Into Capella
Describe what to send from this system. e.g., Exclude attachments...
Describe how to apply incoming data to this system. e.g., Map statuses...

```
replica.type            = issue.type
replica.summary         = issue.summary
replica.description     = issue.description
replica.priority        = issue.priority
replica.attachments     = issue.attachments
replica.status          = issue.status

//Comment these lines out if you are interested in sending the full list of versions and
components of the source project.
replica.project.versions = [ ]
replica.project.components = [ ]

/*
Custom Fields

replica.customFields."CF Name" = issue.customFields."CF Name"
*/
```

```
if(firstSync){
      issue.projectKey   = nodeHelper.getProjectByKey("DT")
      // Set type name from source issue, if not found set a default
      issue.typeName     = nodeHelper.getIssueTypeByName(replica.type?.name)
?:nodeHelper.getIssueTypeByName("Task")
      issue.summary      = replica.summary
      issue.description  = replica.description
      issue.comments     = commentHelper.mergeComments(issue, replica) {
        it.executor = nodeHelper.getUserByEmail(it.author?.email)
      }
      issue.attachments  = attachmentHelper.mergeAttachments(issue, replica)
      issue.priority     = replica.priority
      // Define the mapping between Jira Cloud statuses and
GitHub statuses
     def statusMap = [     "To Do": "open",
                           "Done": "closed" ]
     // Apply the status mapping
     def remoteStatusName =   replica.status.name
issue.setStatus(statusMap[remoteForm.StatusName] ?: remoteStatusName)

/*
User Synchronization (Assignee/Reporter)

Set a Reporter/Assignee from the source side, if the user can't be found set a default
```

5. Click **Select Issues**



exalate

You're in draft mode now. You can edit and save and/publish this version.

Back to editing
Version 2

Start your Test Run

The test configuration runs a simulation of the changes made in a draft version. Apply the configuration to one or more entities, review the results, and check your version for errors. If the outcome meets your expectations, you can publish the version. Ready to start testing? Follow the steps below.

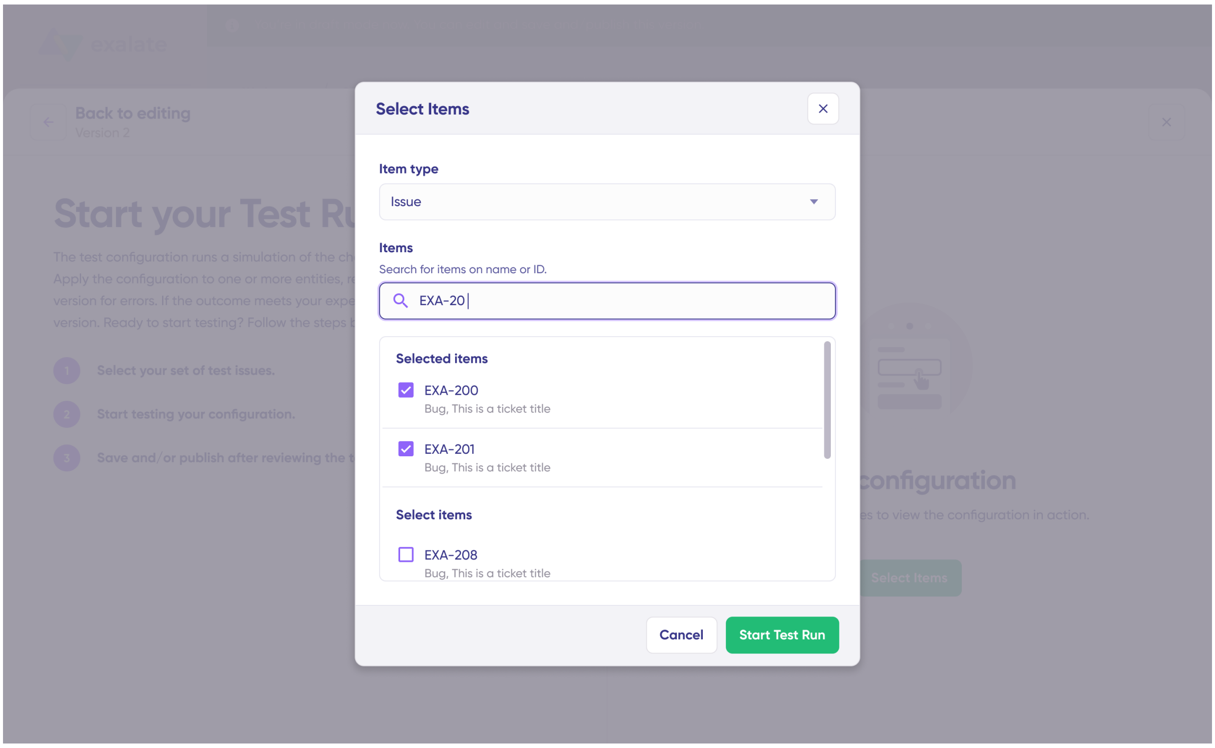
1. Select your set of test issues.
2. Start testing your configuration.
3. Save and/or publish after reviewing the test results.

Test configuration

Start adding test cases to view the configuration in action.

Select Items

6. Search for entities by an entity key and choose one or more entities to apply the test configuration to.



Select Items

Item type

Issue

Items

Search for items on name or ID.

EXA-20

Selected items

☑ EXA-200
Bug, This is a ticket title

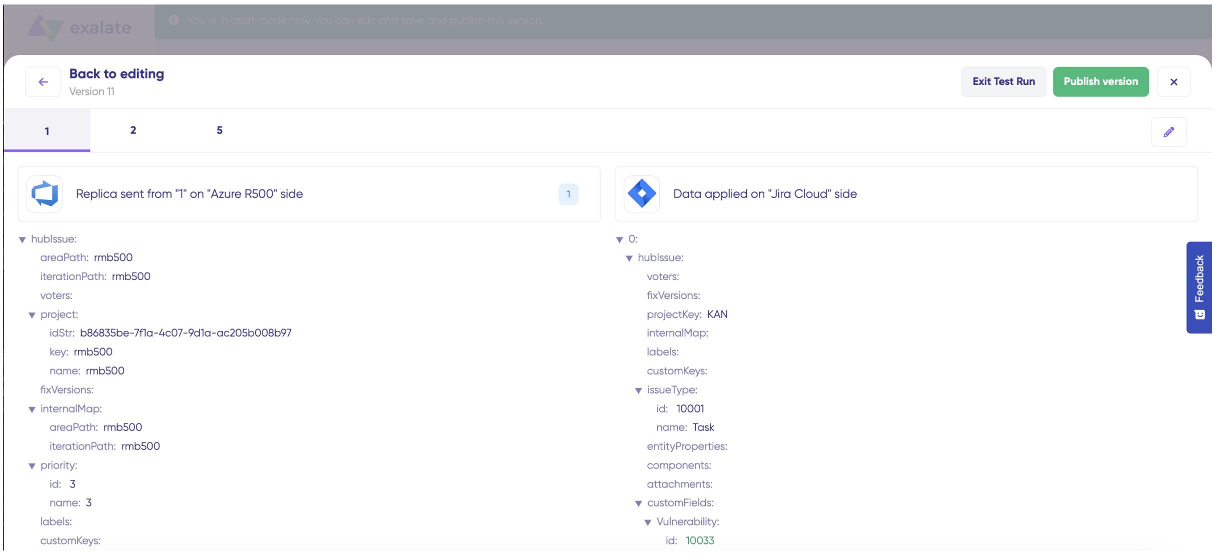
☑ EXA-201
Bug, This is a ticket title

Select items

□ EXA-208
Bug, This is a ticket title

Cancel   Start Test Run

7. Click **Start TestRun** again to see the **outgoing and incoming replicas** for each selected issue.



You're in draft mode now. You can edit and save and publish this version.

Back to editing
Version 11

Replica sent from "1" on "Azure R500" side

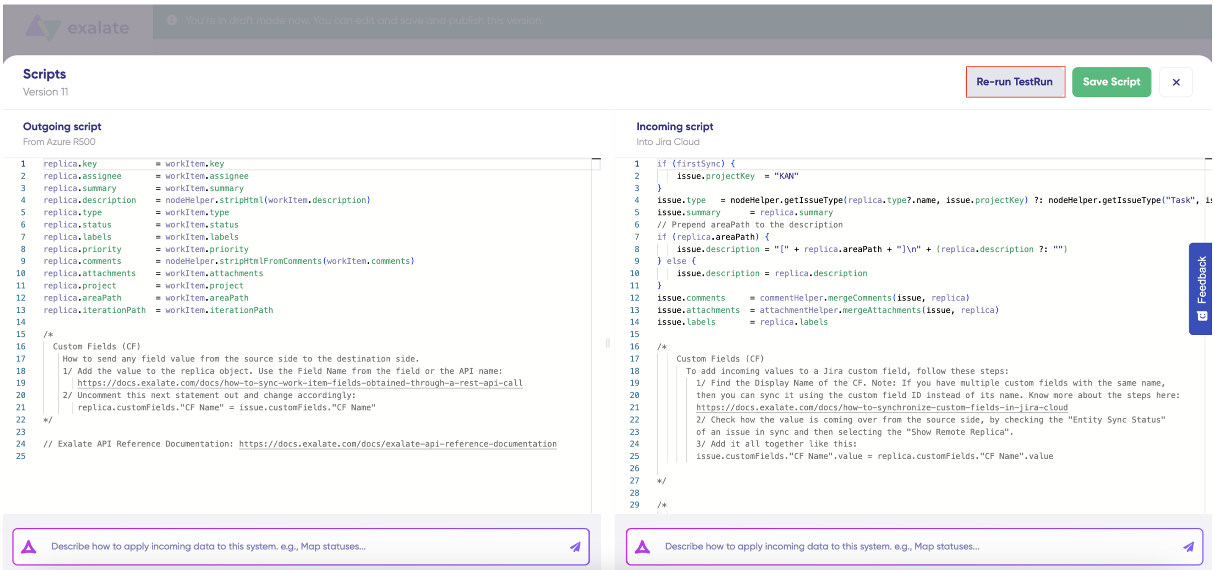
```
▼ hubIssue:
    areaPath: rmb500
    iterationPath: rmb500
    voters:
  ▼ project:
      idStr: b86835be-7f1a-4c07-9d1a-ac205b008b97
      key: rmb500
      name: rmb500
    fixVersions:
  ▼ internalMap:
      areaPath: rmb500
      iterationPath: rmb500
  ▼ priority:
      id: 3
      name: 3
    labels:
    customKeys:
```

Data applied on "Jira Cloud" side

```
▼ 0:
  ▼ hubIssue:
      voters:
      fixVersions:
      projectKey: KAN
      internalMap:
      labels:
      customKeys:
    ▼ issueType:
        id: 10001
        name: Task
      entityProperties:
      components:
      attachments:
    ▼ customFields:
      ▼ Vulnerability:
          id: 10033
```

Refine Your Configuration

- If adjustments are needed, click **Back to Editing**, modify the scripts, and then select **Re-run TestRun** to test the new version on the same set of issues.



You're in draft mode now. You can edit and save and publish this version.

Scripts
Version 11

Outgoing script
From Azure R500

```
replica.key             = workItem.key
replica.assignee        = workItem.assignee
replica.summary         = workItem.summary
replica.description     = nodeHelper.stripHtml(workItem.description)
replica.type            = workItem.type
replica.status          = workItem.status
replica.labels          = workItem.labels
replica.priority        = workItem.priority
replica.comments        = nodeHelper.stripHtmlFromComments(workItem.comments)
replica.attachments     = workItem.attachments
replica.project         = workItem.project
replica.areaPath        = workItem.areaPath
replica.iterationPath   = workItem.iterationPath

/*
  Custom Fields (CF)
    How to send any field value from the source side to the destination side.
    1/ Add the value to the replica object. Use the Field Name from the field or the API name:
       https://docs.exalate.com/docs/how-to-sync-work-item-fields-obtained-through-a-rest-api-call
    2/ Uncomment this next statement out and change accordingly:
       replica.customFields."CF Name" = issue.customFields."CF Name"
*/

// Exalate API Reference Documentation: https://docs.exalate.com/docs/exalate-api-reference-documentation
```

Describe how to apply incoming data to this system. e.g., Map statuses...

Incoming script
Into Jira Cloud

```
if (firstSync) {
    issue.projectKey  = "KAN"
}
issue.type   = nodeHelper.getIssueType(replica.type?.name, issue.projectKey) ?: nodeHelper.getIssueType("Task", is
issue.summary      = replica.summary
// Prepend areaPath to the description
if (replica.areaPath) {
    issue.description = "[" + replica.areaPath + "]\n" + (replica.description ?: "")
} else {
    issue.description = replica.description
}
issue.comments     = commentHelper.mergeComments(issue, replica)
issue.attachments  = attachmentHelper.mergeAttachments(issue, replica)
issue.labels       = replica.labels

/*
    Custom Fields (CF)
      To add incoming values to a Jira custom field, follow these steps:
        1/ Find the Display Name of the CF. Note: If you have multiple custom fields with the same name,
        then you can sync it using the custom field ID instead of its name. Know more about the steps here:
        https://docs.exalate.com/docs/how-to-synchronize-custom-fields-in-jira-cloud
        2/ Check how the value is coming over from the source side, by checking the "Entity Sync Status"
        of an issue in sync and then selecting the "Show Remote Replica".
        3/ Add it all together like this:
        issue.customFields."CF Name".value = replica.customFields."CF Name".value

*/

/*
```

Describe how to apply incoming data to this system. e.g., Map statuses...

- To change the test issues, click the **Edit icon**, add or remove issues, and restart the TestRun.

Finalize and Publish

- Once satisfied with the results, click **Save Scripts** to store the changes.
- Click **Publish** to apply the updated configuration to your live synchronization.

Still need help?

Join our Community

Visit our Service Desk

Find a Partner