

Using Aida AI for Script Configuration

Last Modified on 01/15/2026 9:31 am EST

Aida is Exalate's AI-powered integration assistant embedded into the Exalate script editor to help you generate and modify synchronization scripts through natural language prompts. Instead of writing Groovy code manually, you can describe what you want to achieve, and Aida will draft the code for you.

Accessing Aida

1. Navigate to your connection
2. Click **Add new version** (or open the latest draft version)
3. Click **Edit** to enter the script editor

The screenshot shows the exalate platform interface. At the top, a banner says "You're looking at an active version of this connection. To edit, create a new version or select an existing draft." The navigation bar includes "Workspaces" (selected), "Orlando", and "Support Team to Dev Team". On the left sidebar, there are links for "Workspaces", "Users", "Settings", and "Resources". The main content area shows a connection titled "Support Team to Dev Team" with a "Dev Team" icon and a "Support Team" icon. Below this are buttons for "Version 3 (Active)", "+ New version", "Open latest draft", and "Sync Monitor". The "Triggers" section lists two items: "Issue" (Project = EXA) and "Sprint" (Project = SUP). The "Scripts" section is divided into "Outgoing script" (From Dev Team) and "Incoming script" (Into Support Team), both with an "Edit" button.

You're looking at an active version of this connection. To edit, create a new version or select an existing draft.

Workspaces / Orlando / Support Team to Dev Team

Connection: Support Team to Dev Team

Dev Team → Support Team

Version 3 (Active) + New version Open latest draft ⋮ Item Sync Monitor >

Triggers

+ Add Trigger

Item type	Query	Status
Issue	Project = EXA	<input type="checkbox"/> Edit ⋮
Sprint	Project = SUP	<input type="checkbox"/> Edit ⋮

Scripts

Edit

Outgoing script
From Dev Team

```
1 replica.type      = issue.type
2 replica.summary  = issue.summary
3 replica.description = issue.description
```

Incoming script
Into Support Team

```
1 if(firstSync){
2   issue.projectKey = nodeHelper.getProjectKey("DT")
3   // Set type name from source issue, if not found set a default
```

John Doe
System Admin

Logout

You'll see Aida chat interfaces at the bottom of both the **Outgoing script** and **Incoming script** sections. Each Aida chat works independently, allowing you to configure both sides of your connection simultaneously.

Scripts
Version 2

Outgoing script
From Jira Cloud

```
1 // Jira Issues are now work items. The term "issues" in the code below works despite this change.
2 replikaKey = $IssueKey
3 replikaType = "Issue"
4 replikaAssignee = $IssueAssignee
5 replikaTitle = $IssueSummary
6 replikaNumber = $IssueSummary
7 replikaLastUpdated = $IssueLastUpdated
8 replikaLabels = $IssueLabels
9 replikaComments = $IssueComments
10 replikaLastComment = $IssueLastComment
11 replikaStatus = $IssueStatus
12 replikaLastUpdatedAt = $IssueLastUpdatedAt
13 replikaPriority = $IssuePriority
14 replikaLastUpdatedAt = $IssueLastUpdatedAt
15 replikaProject = $IssueProject
16
17 // Document these lines. If you want to send the full list of versions and components of the source project.
18 replikaProjectVersions = []
19 replikaProjectComponents = []
20 replikaProjectAttachments = []
21
22 /*
23  * Custom Fields (CF)
24  * How to send any field value from the source side to the destination side.
25  * 1. Create a field in the source project using the Name of the specific field.
26  * 2. Uncomment this next statement out and change accordingly:
27  *    replikaCustomFields["CF Name"] = liston.out.fields["CF Name"]
28 */
29
30
31 // Exalate APL Reference Documentation: https://docs.exalate.com/docs/exalate-apl/reference-documentation
```

Incoming script
Into Azure RG00

```
1 //FirstSync
2 // Set Type now True Source entity, if not found set a default
3 var replikaProjectKey = "REPLIKE"
4 workItem.typecode = isnone(replikaProjectKey) ? "Task" : "Issue"
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
```

Start Test Run **Save Script** **X**

How Aida Works

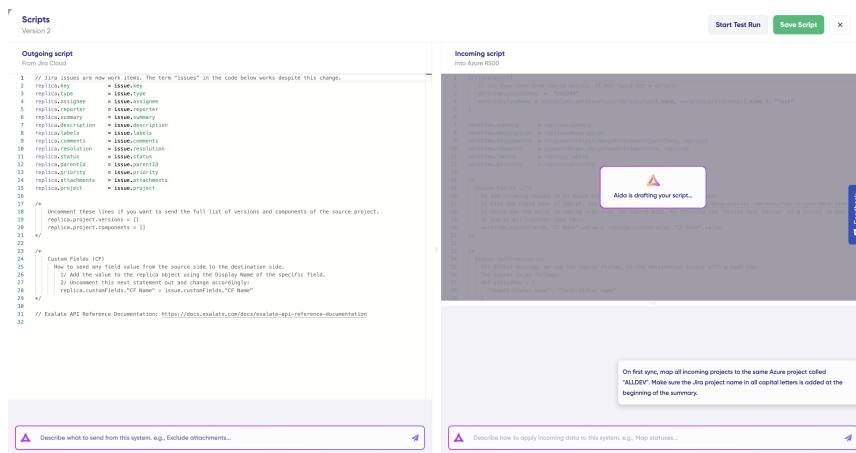
Aida helps you in two ways:

- **For Outgoing scripts:** Describe what data should leave your system. For example, "Exclude attachments" or "Only sync high-priority issues."
- **For Incoming scripts:** Describe how incoming data should be applied to your system. For example, "Map statuses" or "Set a default assignee if the user can't be found."

Using Aida

Submitting a Prompt

1. In the Aida input field, type your request in plain language
2. Click the send button (or press Enter)
3. Aida will begin drafting your script



Example prompts:

- "Exclude attachments from sync."
- "Map priority field to remote system."
- "On first sync, map all incoming projects to the same Azure project called 'ALLDEV'"
- "Set a Reporter/Assignee from the source side, if the user can't be found, set a default user."

Reviewing Aida's Changes

Once Aida finishes, you'll see:

- **Green highlighting:** New lines that will be added
- **Red highlighting:** Lines that will be removed
- A floating panel with **Insert** and **Discard** buttons

- Aida's explanation of the changes in the chat

The script editor becomes read-only during review, so you must choose to either insert or discard the changes before making manual edits.

The screenshot shows the Aida interface with two main script panels: 'Outgoing script' and 'Incoming script'.

Outgoing script (From Jira Cloud):

```

1 // Jira issues are now work items. The term "issues" in the code below works despite this change.
2 replica.key      = issue.key
3 replica.type     = issue.type
4 replica.assignee = issue.assignee
5 replica.reporter = issue.reporter
6 replica.summary  = issue.summary
7 replica.description = issue.description
8 replica.labels   = issue.labels
9 replica.components = issue.components
10 replica.resolution = issue.resolution
11 replica.status   = issue.status
12 replica.parentId = issue.parentId
13 replica.priority  = issue.priority
14 replica.workItems = issue.workItems
15 replica.project  = issue.project
16
17 /* Uncomment these lines if you want to send the full list of versions and components of the source project.
18 replica.project.versions = []
19 replica.project.components = []
20 */
21
22 */
23 */
24 Custom Fields (CF)
25 How to send any field value from the source side to the destination side
26 1/ Add the value to the replica object using the Display Name of the specific field.
27 2/ Uncomment this next statement out and change accordingly:
28 replica.customFields."CF Name" = issue.customFields."CF Name"
29 */
30
31 // Exalate API Reference Documentation: https://docs.exalate.com/docs/exalate-api-reference-documentation
32

```

Incoming script (Into Azure R500):

```

1 1 if(firstSync)
2   - // Set type name from source entity, if not found set a default
3   - workItem.projectKey = "rmb500"
4   - 2+ // Set all incoming issues to the 'ALLDEV' Azure project
5   - 3+ def workItemKey = "ALLDEV"
6   - 4+ workItem.typeName = modelHelper.getIssueType(replica.type7.name, workItem.projectKey7.name ?: "Task"
7   - 5+ // Prepend the Jira project name (in all caps) to the summary
8   - 6+ def jiraProjectName = (replica.project7.name ?: "").toUpperCase()
9   - 7+ workItem.summary = jiraProjectName ? (jiraProjectName + ":" + replica.summary)
10  - 8+ workItem.summary = replica.summary
11
12 6 11
13 7 workItem.summary = replica.summary
14 8 workItem.description = replica.description
15 9 workItem.attachments = attachmentHelper.mergeAttachments(workItem, replica)
16 10 workItem.comments = commentHelper.mergeComments(workItem, replica)
17 11 workItem.labels = replica.labels
18 12 workItem.priority = replica.priority
19 13
18 */
19 19 *Custom Fields (CF)
20 20 To add field values to an Azure DevOps custom field, follow these steps:
21 21 1/ Set the Field Name of the CF. You can also use the API name: https://docs.exalate.com/docs/how-to-sync-work-items
22 22 2/ Check how the value is coming over from the source side, by checking the "Entity Sync Status" of a ticket in Jira
23 23 3/ Add it all together like this:
24 24 workItem.customFields."CF Name".value = replica.customFields."CF Name".value
25 25 */
26 26
23 */
24 28 Status Synchronization

```

Feedback and Notes:

- Outgoing Script:** A purple box with a pencil icon and the text "Describe what to send from this system, e.g., Exclude attachments..."
- Incoming Script:** A purple box with a pencil icon and the text "Describe how to apply incoming data to this system, e.g., Map statuses..."
- Information Panel:** A purple box containing Aida's explanation of the implemented requirements.

Working with Scripts on Both Sides (Incoming and Outgoing)

The Outgoing and Incoming Aida chats work independently, so you can:

- Submit a prompt in the Outgoing script while Aida is generating code for the Incoming script
- Maintain separate conversation contexts for each script direction

Adjusting Your Workspace

You can resize the script panels and Aida chat windows to focus on what matters most:

- Drag the handle at the top of the Aida chat to increase or decrease its height
- Drag the divider between Outgoing and Incoming scripts to adjust their widths

Important Notes

- Aida is a helpful assistant, but you should always review the generated code before applying it
- Test your configuration using **Start Test Run** before publishing changes to production
- Aida works best when you provide clear, detailed descriptions of your sync requirements
- You can always manually edit the scripts after applying Aida's suggestions

Product Aida

[About Us](#) 

[Using Aida](#) 

[Glossary](#) 

[Adjusting Your Workspace](#)

[API Reference](#) 

[Important Notes](#)

[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](#) 