Jira to Jira Integration

Last Modified on 03/28/2025 4:41 am EDT

Jira to Jira Integration

Getting two Jira instances to exchange data through native solutions. However, the problem is that the available options are limited in applicability. They only allow the integration of certain fields without the possibility to customize the mapping.

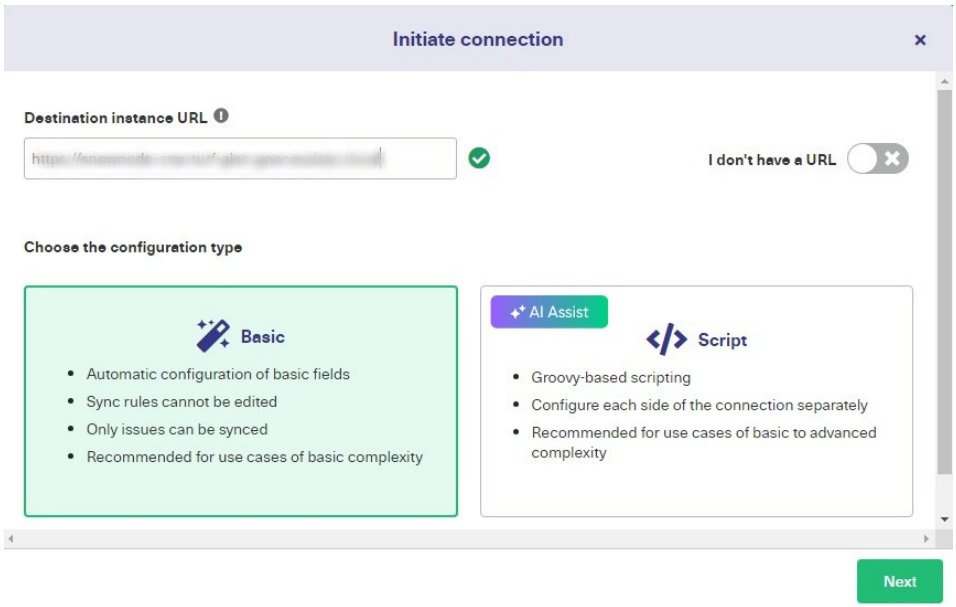
By contrast, Exalate allows you to customize every connection using an AI-powered scripting engine. It supports Jira Service Management, Jira Software, and Jira Work Management.

You can install it on Jira Cloud or Jira on-premise or even set up a private connection in case any of your instances are behind a firewall.

Exalate Installation and Configuration Steps

Here is a brief overview of a Jira-to-Jira configuration.

1. Install the Exalate app on Jira. You can start from the Atlassian marketplace.
2. Choose the connection type: Basic or Script. You will also see an option for the Visual mode, but it is the BETA version.



To learn how to set up a connection between two Jira instances using the Script mode, go to the Getting Started guide for a detailed visual breakdown. Just choose the two platforms you want to configure, and you'll get a comprehensive explanation of how to proceed.

Advanced Jira to Jira Integration Use Cases with AI Assist

Exalate provides an AI-powered scripting engine embedded in the configuration panel. This allows users to explore multiple advanced use cases.

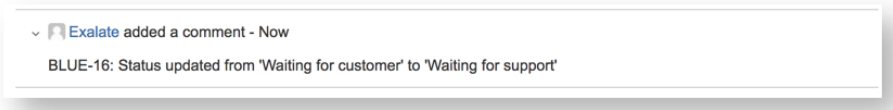
It also makes Script Helpers available to simplify the scripting of connections from scratch. For a step-by-step breakdown, check out a detailed Jira to Jira integration guide.

Here are some integration use cases for multiple Jira syncs.

Use Case 1: Map and Sync Statuses between Two Jira Instances

If you have two connected Jira instances, and you want the status change to be reflected on the other side as a comment, you need a rule to make sure the right statuses are mapped on both sides.

This scenario is common for cross-platform collaborations, such as when you connect your internal Jira On-premise with an MSP's Jira Cloud.



For instance, when the Jira A issue status changes to "**Waiting for customer"**, the Jira B issue automatically receives a comment indicating that the status has changed to "**Waiting for Support**".

Enter the code snippet or script mapping you want for your use case, or use AI Assist to generate the code by typing in a detailed prompt describing what you want to sync.

Go through the generated output to confirm if it aligns with your expectations. You can continue refining the prompt until you get what you need.

```
if (!replica.status.equals(previous?.status)) {
  // construct the comment
  String message = "${replica.key}: Status updated from '${previous?.status?.name}' to '${replica?.status?.name}'
"
  issue.comments = commentHelper.addComment(message, issue.comments)
}
```

Click **Discard** if the generated code is incorrect. If the generated script is correct, click **Insert Changes**. Once you're satisfied with the scripting, click **Publish** to save and implement changes.

Note: Review the AI Assist prompting guidelines to improve your prompts and, thus, the output.

Use Case 2: Map and Sync Story Points between Jira Instances

If the default fields and entities in Jira don't meet your requirements, you can add custom fields with a solid guarantee that Exalate can fetch and exchange data between them.

The prompt to the AI Assist can be something like this:

> *"I want to sync a Jira custom field named "Story Points" with another Jira custom field named "Story Points"*

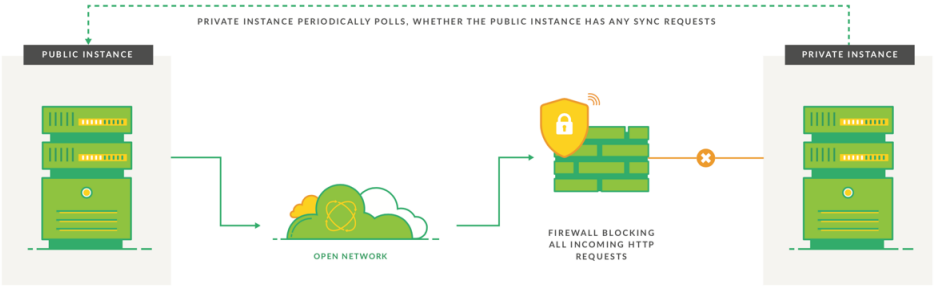
Here is the code snippet generated by AI Assist for the Jira incoming sync.

```
issue.customFields."Story Points".value = replica.customFields."Story Points".value
```

You can discard, accept, or refine the prompt to nail down the specifics of your use case.

Use Case 3: Sync a Private Jira Instance with a Public Jira Instance

Say you have two Jira instances: the private instance (Jira On-Premise) is not accessible from the outside network, and the other instance (Jira Cloud) is public.

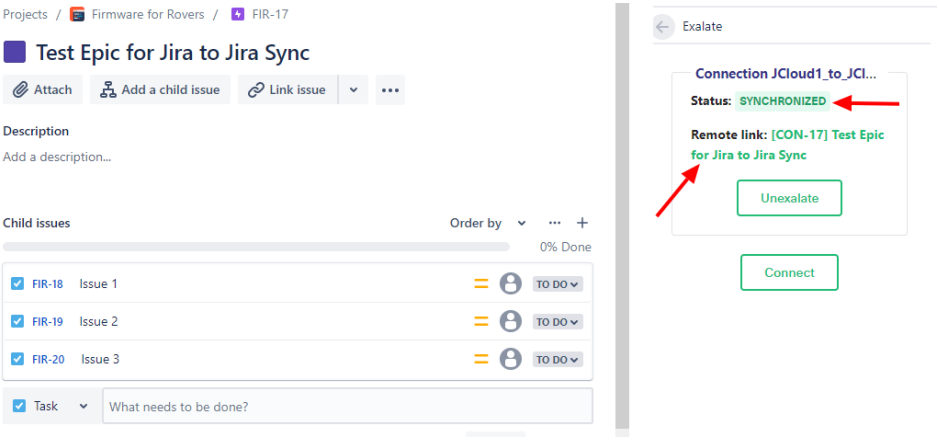


To sync both sides, follow the steps to create a Jira-to-Jira connection. But instead of entering a URL, click on **“I don’t have a URL”.** This indicates to the server that one of the instances should be kept private.

You can also connect two private Jira instances via Exalate.

Use Case 4: Sync Epics Between Jira Cloud and Jira On-Premise

When you create a Jira epic and add issues to it on your instance, the changes should appear on the other end without having to replicate the same issues manually.



For this to work, you need two simple lines of code.

On the Jira Cloud incoming side, enter the following line of code:

```
Epic.receive()
```

This *receive()* method fetches all the contents (project, description, status, priority, etc.) of an epic

from the incoming payload stored in the replica.

On the Jira On-premise outgoing side, enter the following line of code:

```
Epic.send()
```

The *send()* method forwards the contents of the Epic to the Jira Cloud side.

Read more about this use case on our blog.

Use Case 5: Sync User Mentions in Comments Between Jira Cloud and Jira On-premise

Exalate allows you to synchronize user mentions when users exist in both systems. You can also use this functionality when the user does not exist in one of the systems. In this case, a custom message must be synced with the user's mention.

The outgoing script on the Jira Cloud side should look like this:

```
replica.comments = issue.comments.collect {
    comment ->
    def matcher  = comment.body =~ /[~accountid:([w:-]+)]/
    def newCommentBody = comment.body
    matcher.each {
        target = nodeHelper.getUser(it[0])?.email ?: "Stranger"
        newCommentBody = newCommentBody.replace(it[0],target)
    }
    comment.body = newCommentBody
    comment
}
```

The *nodeHelper.getUser()* method replaces the user mention with the email address of the corresponding user for each such comment.

So the replica sent to the on-premise instance contains the actual email address instead of the user mention.

The incoming script on the Jira On-premise side should look like this:

```
for(comment in replica.addedComments){
    def newCommentBody=comment.body
    def matcher  = comment.body =~ /[a-zA-Z0-9+._-]+@[a-zA-Z0-9._-]+.[a-zA-Z0-9_-]+/
     matcher.each {
        x->
        if (nodeHelper.getUserByEmail(x)){
            def target = "[~" + nodeHelper.getUserByEmail(x).username + "]"
            newCommentBody = newCommentBody.replace(x,target)
        }
        else
              newCommentBody = newCommentBody.replace(x,"[External user with email ${x} mentioned]")
     }
    comment.body=  newCommentBody
}
issue.comments     = commentHelper.mergeComments(issue, replica)
```

Over here, the *nodeHelper.getUserByEmail()* method fetches the user account corresponding to the email address before replacing it with the username.

To find the entire code for both the incoming and outgoing data, refer to this comprehensive guide.

AI Assist, like any other AI, can make mistakes. So, try to be as precise and detailed as possible with your prompts.

If you have a specific use case to discuss, book a demo with our engineers.

Note: The code snippet might not work precisely as intended due to changes to the environment or other reasons. If you encounter any problems, reach out to us for clarification.

Automate Jira Integration Using Triggers

Exalate uses Jira Query Language (JQL) to set trigger conditions on the Jira side.

```
project = DEMO AND labels = vital
```

Any project with the name “DEMO” and the label “vital” will be synced automatically.

Supported Jira Cloud and On-Premise Entities

You can sync any entities or issues between any permutation of Jira Cloud and Jira On-premise.

Check out the comprehensive list of supported Jira Cloud and On-premise entities.

This is a sample mapping between Jira On-premise and Jira Cloud:

Jira Cloud issue <> Jira On-premise issue

- `short description↔ short description`
- `description ↔ description`
- `priority ↔ priority`
- `state ↔ status`
- `comments↔ comments`
- `attachments ↔ attachments`
- `labels ↔ labels`
- `custom fields ↔ custom fields`
- `third-party plugin fields`
- `any field available via REST APIs`

Jira Cloud Epic ↔ Jira On-premise Epic

- `All available issue fields`
- `parentID ↔ parentID`
- `resolution ↔ resolution`
- `project ↔ project`

Video Tutorials

- Watch the installation and configuration tutorial for Jira.
- Watch the installation and configuration demo for all connectors.

Other resources

- Download the Jira ServiceNow integration eBook.
- Talk to Aida, your AI-powered integration sidekick, and get answers to your questions faster.
- Check out the detailed security and architecture whitepaper.
- Visit the Exalate Academy to get access to learning materials.
- Subscribe to Exalate Hack to get email updates and expert tips about the product.

Product
About Us
Release History
Glossary
API Reference
Security
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

ON THIS PAGE