



Jira Azure DevOps Integration: The Complete Step-by-Step 2023 Guide



Table of contents

What are the Benefits of Integrating Jira and Azure DevOps?

- What is Jira?
- What is Azure DevOps?
- The Benefits of a Jira and Azure DevOps Integration

How to Set up a Jira Azure DevOps Integration in 6 Steps

- Step 1 - Install Exalate on Jira
- Step 2 - Install Exalate on Azure DevOps
- Step 3 - Connect Your Jira and Azure DevOps Instances
- Step 4 - Configure Your Connection to Determine What Gets Shared
- Step 5 - Set Up Automated Synchronization Triggers
- Step 6 - Start Synchronizing Tasks

Common Use Cases for a Jira Azure DevOps Integration

- Backend and Frontend Developers
- Customer Service and Engineering Teams
- Marketing and Design Teams

Conclusion



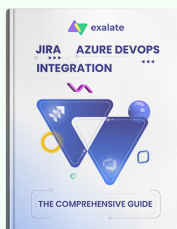
More and more teams are using work management platforms like Jira and Azure DevOps to organize their work and manage data. Getting the most out of these tools is key to improving business performance and nudging your productivity up. That's where the right solution for integration, like a Jira Azure DevOps integration, comes into the picture.

Integrating the software you use to store business information can make your life much easier and your collaborations way more efficient. If your teams are well connected, then they can take advantage of everyone's expertise and complement each other more effectively.

So in this guide, you'll learn how to set up a Jira Azure DevOps integration following 6 straightforward steps.

Here's an overview of what's covered in this blog post:

- [What are the Benefits of Integrating Jira and Azure DevOps](#)
- [How to Set up a Jira Azure DevOps Integration in 6 Steps](#)
 - [Continue with the Basic Mode](#)
 - [Continue with the Visual Mode](#)
 - [Continue with the Script Mode](#)
- [Common Use Cases](#)



Get the Jira Azure DevOps Sync Guide

Learn how to integrate Jira and Azure DevOps, step-by-step.

[GET THE EBOOK](#)

What are the Benefits of Integrating Jira and Azure DevOps?

What is Jira?

Jira is a project management and issue tracking platform that lets you assign tasks to team members and track your progress.



It is highly customizable and suitable for teams working with a range of different software. It is particularly suited to teams using agile methodologies. You can create sprints, for example. It is a versatile tool though and can handle many different use cases.

What is Azure DevOps?

Azure DevOps has a range of functions, including project management, version control, and build automation. It has features for teams using agile and waterfall methodologies.



Azure DevOps works well with Visual Studio and Eclipse, though can be used with other software as well. Its work item system can be used to represent bugs, issues, or anything else you need.

Both Jira and Azure DevOps have a range of expansions that you can use to give them extra functionality that can be used to integrate them.

The Benefits of a Jira and Azure DevOps Integration

To make sure you get the most out of your integration, it is important to understand what your goals are. If you keep the potential benefits in mind when setting things up, you'll be able to make sure the integration delivers everything it can.

Essentially, the integration is going to automatically sync data between your two platforms. This will help reduce work for both teams using the integration.

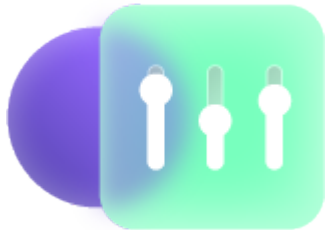
It prevents you from duplicating information and potentially working on the same problems. It also allows you to take advantage of knowledge and research carried out by each other and helps you all work towards the same goals.

Sharing issues, but tailoring them to the needs of each team allows each team to work more efficiently. So you wouldn't be distracted by unnecessary information, and you can add fields for your own exclusive use case.

There are three particular features to be taken into consideration when setting up a Jira Azure DevOps integration, or any other integrations:

Decentralized Integration

Though you are connecting your teams together, you need them to be able to act separately and retain control over their own data and what and how they share it. They may also have security or legal concerns, especially if they're handling customers' critical data.



An integration should give both sides of the connection the ability to see what is being sent out and passed into their systems and to adjust what they need without consulting the other team. Decisions on what is shared should be theirs and they should be able to change how incoming data is used as required.

Reliability

Even in well-designed systems, things sometimes go wrong so your solution needs to handle these problems gracefully. Any errors or changes should be detected and the integration should be able to recover without manual intervention.

Downtime is another unfortunate fact of life that we need to account for when integrating platforms. The solution needs to be able to recover when one or both sides of the integration become unreachable.

Flexibility

Your teams' needs will change over time. Different projects often bring different ways of working. The data we collect from customers may broaden or lessen in scope.

Sometimes this means we'll need to tune our fields. At other times we may need to share different things entirely. When you make these changes, our solution needs to handle them without fuss on either side.

The tool I've chosen here is called [Exalate](#). Exalate was designed with these issues in mind and can deliver the smoothest integration possible. It gives your team autonomy, enables them to work flexibly, and is reliable enough to handle errors.

Now that we've chosen our solution, let's get down to setting up the sync.

How to Set up a Jira Azure DevOps Integration in 6 Steps

The first two steps are to install Exalate on both platforms. After that, you'll connect the instances, and then you'll see how information flowing between the instances can be controlled.

We'll get to the step-by-step process of the Jira Azure DevOps integration, but if you prefer watching a tutorial, you can go ahead and watch this video instead.



<https://youtu.be/7P2paKVVukc>

Step 1 - Install Exalate on Jira

The first step is to install Exalate on Jira. You need to know what version of Jira you're using.






This guide assumes you're using Jira cloud, though you can also read a [Jira Cloud installation guide](#) in Exalate's documentation. Otherwise, please take a look at this guide for [Jira Server and Data Center](#) users.

Sign in to your Jira account and then look for the cog at the upper right of the screen.




Q Search   

Settings

JIRA SETTINGS

-  **System**
Manage your general configuration, global permissions, look and feel and more.
-  **Products**
Manage your Jira products' settings and integrations.
-  **Projects**
Manage your project settings, categories, and more.
-  **Issues**
Configure your issue types, workflows, screens, custom fields and more.
-  **Apps**
Add and manage Jira Marketplace apps.

PERSONAL SETTINGS

-  **Atlassian account settings** 
Manage your language, time zone, and other profile information.
-  **Personal Jira settings**
Manage your email notifications and other Jira settings.

Click it, and then click “apps” in the menu. You may have to re-enter your login credentials as you’ve selected an admin function.


On the next screen, look at the left-hand menu. If it isn’t already selected, click “Find new apps” under the “Atlassian Marketplace” heading.

In the field that says “Search the Marketplace”, type in “Exalate” and press enter.

Discover apps and integrations for Jira


Sort Free for all teams More Filters Categories


10 results



Exalate Jira Issue Sync & more
Two-way Jira sync tool. Supports Jira to Jira Integration, ServiceNow, Github, Zendesk, Salesforce, Azure DevOps integration...
IT & helpdesk, Integrations, Tasks, Workflow


★★★★★ 80 **ADDED**


 **CLOUD FORTIFIED**




Exalate: ServiceNow to Jira Integration
Experience a seamless, real-time and two-way ServiceNow Jira Integration
Admin tools, Continuous integration, IT & helpde...

★★★★★ 3 271 downloads

 **CLOUD SECURITY PARTICIPANT**



Exalate: Zendesk to Jira Integration
Experience a seamless, real-time and two-way Zendesk Jira Integration
Admin tools, Continuous integration, IT & helpde...

 **CLOUD SECURITY PARTICIPANT**

Click the “Exalate Jira Issue Sync & more” app at the top. You will also see Exalate apps for other platforms, such as Azure DevOps, ServiceNow, and GitHub. So be sure to pick the correct one.

Click the “Try it free” button and you’ll then be taken through a couple of screens.

Follow the process to start your trial and take you back to Jira, where you’ll see a confirmation message.

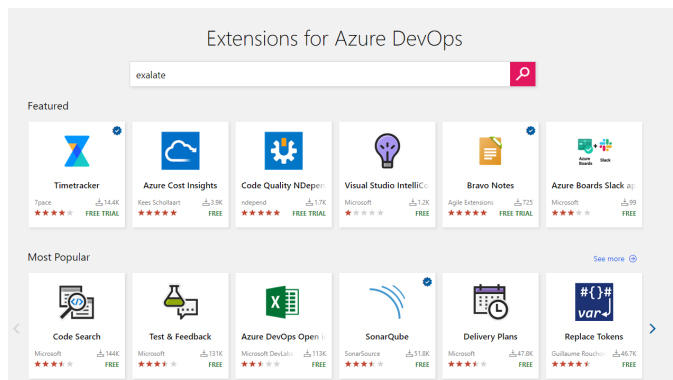
Click “Get Started” to complete the installation on Jira.

Step 2 - Install Exalate on Azure DevOps

Now you need to install Exalate on Azure DevOps. Here’s a guide to [installing it from the marketplace](#). You should also check out the general documentation on Azure DevOps [here](#).

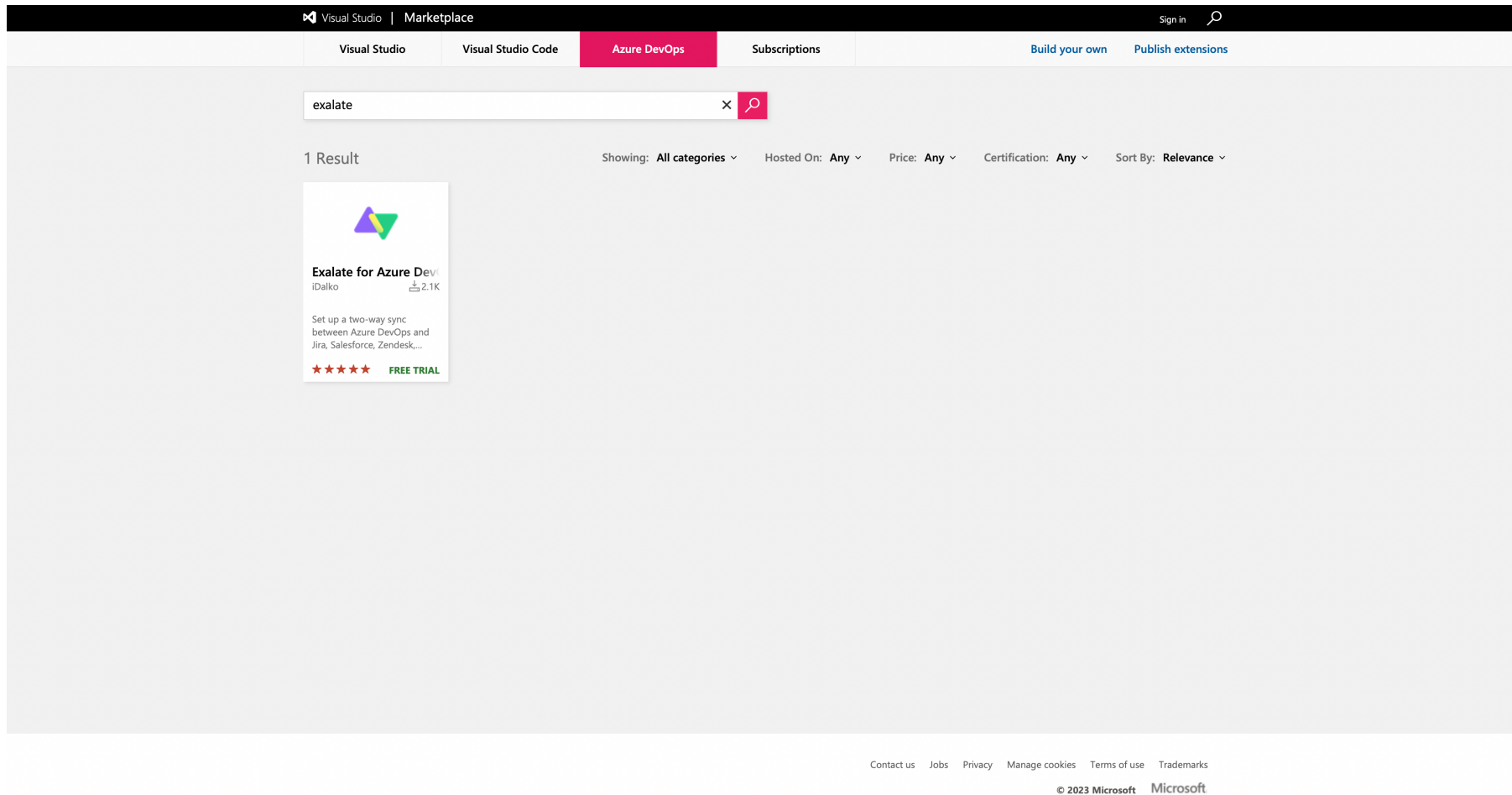
This guide focuses on marketplace installation, but you can also [install it via Docker](#) if you’re running your own server.

Note: For installing Exalate on Azure DevOps you will need to [generate a Personal Access Token \(PAT\)](#).




To install it via the marketplace, log in to Azure DevOps, then click the shopping bag icon at the top right of the screen.

Select “Browse Marketplace”. Type “Exalate” into the search field.



Click the “Exalate for Azure DevOps” app and then on the next screen, click the green “Get” button to activate the free trial.

Azure DevOps > Azure Boards > Exalate for Azure DevOps Integration



Exalate for Azure DevOps Integration

iDalko | 📄 2,101 installs | ★★★★★ (4) | Paid

Set up a two-way sync between Azure DevOps and Jira, Salesforce, Zendesk, GitHub, or any other work management systems. Experience seamless collaboration across internal teams and company borders.

Get
30 days free trial

Overview
Pricing
Q & A
Rating & Review

Set up a two-way synchronization between Azure DevOps and Jira, ServiceNow, Salesforce, Zendesk, Github or any other work management systems.

Experience a seamless collaboration across internal teams and company borders.

Exalate Features include:

Limitless flexibility to fit your unique synchronization scenarios. Sync almost anything including title, description, tags, custom fields, attachments, comments, state, etc.

- Use the groovy-based scripting engine for maximum flexibility in configuration.
- Use the no-code mode for an intuitive, seamless, drag-n-drop builder for easy configuration.
- Use the Basic mode with the Free Plan for automatic configuration of basic fields such as title, description, comments, attachments, and straight out-of-the-box sync scenarios.

Categories

Azure Boards

Tags

almmicrofocus

azuredevops

github

hpcq

integration

jcloud

jira

jiraserver

servicenow

zendesk

Works with

Azure DevOps Services


You’ll be taken through an installation wizard. Select your organization from the dropdown box and click the “Install” button.

Click “Proceed to organization” on the next screen and Exalate is installed.



 Organization  Done

Select an Azure DevOps organization

 This extension is already installed on this organization: [tejabhutada](#).

[Go to Marketplace](#)

Now you'll need to get an evaluation license. To do this, look at Exalate's left-hand menu. Click "License details", then click on "30-day-trial".

A pop-up will appear. Enter your email address. Wait a few minutes, then check your email for a message with your evaluation key.

On the Exalate license details screen, click the green "License Key" button. Paste in the code you were sent in the email.

Click "Update" and you're done.

Note: After installing Exalate as shown above proceed to [verify it on Azure DevOps](#).

Step 3 - Connect Your Jira and Azure DevOps Instances

Now that you've installed Exalate on each platform, it's time to connect them and set up the Jira Azure DevOps integration.

You can initiate the connection from either platform. For this guide, I'll start from the Azure DevOps side.

Log in, and if you're not there already, navigate to Exalate by clicking the marketplace icon, then "Manage extensions", then clicking "Exalate" from the left-hand menu under the "Extensions" heading.

In the Exalate menu, click "Connections". You'll see a list of any previously created connections, though if this is your first time here, there won't be any. Click the green "Initiate connection" button at the top right to get started.

Next, choose your destination URL. In this case, that's the address of your Jira instance.

If you were doing this step in Jira, you would use your Azure DevOps instance address instead.



Now a quick check is done to detect if Exalate is installed on the node you entered. If so, you are prompted to select anyone out of the above 3 configuration modes: The **Basic**, **Visual**, or **Script** mode.

I'll take you through all of them one by one, but in case you want to skip and go to a certain configuration mode directly, please jump to the relevant section.

Continue with the Basic Mode

After clicking “Next” on the screen above, you are redirected to another screen that prompts you to select the project you want to synchronize work items for.

Initiate connection ×

Select a project for the incoming sync

Exalate generates default sync rules to synchronize basic work item fields. By default the following work item data will be synchronized: summary, description, comments, attachments and work item types.

Please select the project where you want to create work items, received from the other side.*

DOPS | ▾


[← Previous](#) [Next](#)

Hit “Next” after selecting the project.


You need to confirm if you have admin access to the Jira side. If not please click on “No, I don’t have admin access”. When you click this, you will be shown an invitation code, which you need to copy and paste on the Jira side by clicking on “Accept Invitation” in the “Connections” tab.

Initiate connection ×

Do you have admin access to the destination instance?

 **Yes, I have admin access**

- You will be redirected to the destination instance to establish the connection

 **No, I don't have admin access**

- You will generate an invitation and send it to the destination instance admin to establish the connection

[← Previous](#) [Initiate](#)

After clicking “Initiate”, the same process is followed - selecting the project on the Jira side.

Click “Confirm” and your connection has been successfully established. Once that is done, you can immediately start to synchronize your first issue by entering the issue key and then “Exalate” the issue. You can even [bulk exalate](#) issues/work items or [create triggers](#) for synchronizing them.

Accept invitation ×

Select a project for the incoming sync

Exalate generates default sync rules to synchronize basic issue fields. You can adapt the sync rules later. By default the following issue data will be synchronized: summary, description, comments, labels and attachments.

Please select the project where you want to create issues, received from the other side.*


 ▾

[< Previous](#) [Confirm](#)


Wait for a short while and then you'll see your issue is synchronized.


Accept invitation ✕

Congratulations!




JiraScript





Azure DevOps



Connection is established.

Configure Sync

Continue with the Visual Mode

Visual Mode is a great way to sync your work items/ issues without having to write code. The UI is very intuitive, so business and technical users alike can use it with ease.

Click “Next” on the screen that asks you to choose between the 3 modes, name your connection, and enter a description.

Initiate connection ✕

Connection information

Local instance short name*

Remote instance short name*

Connection name*

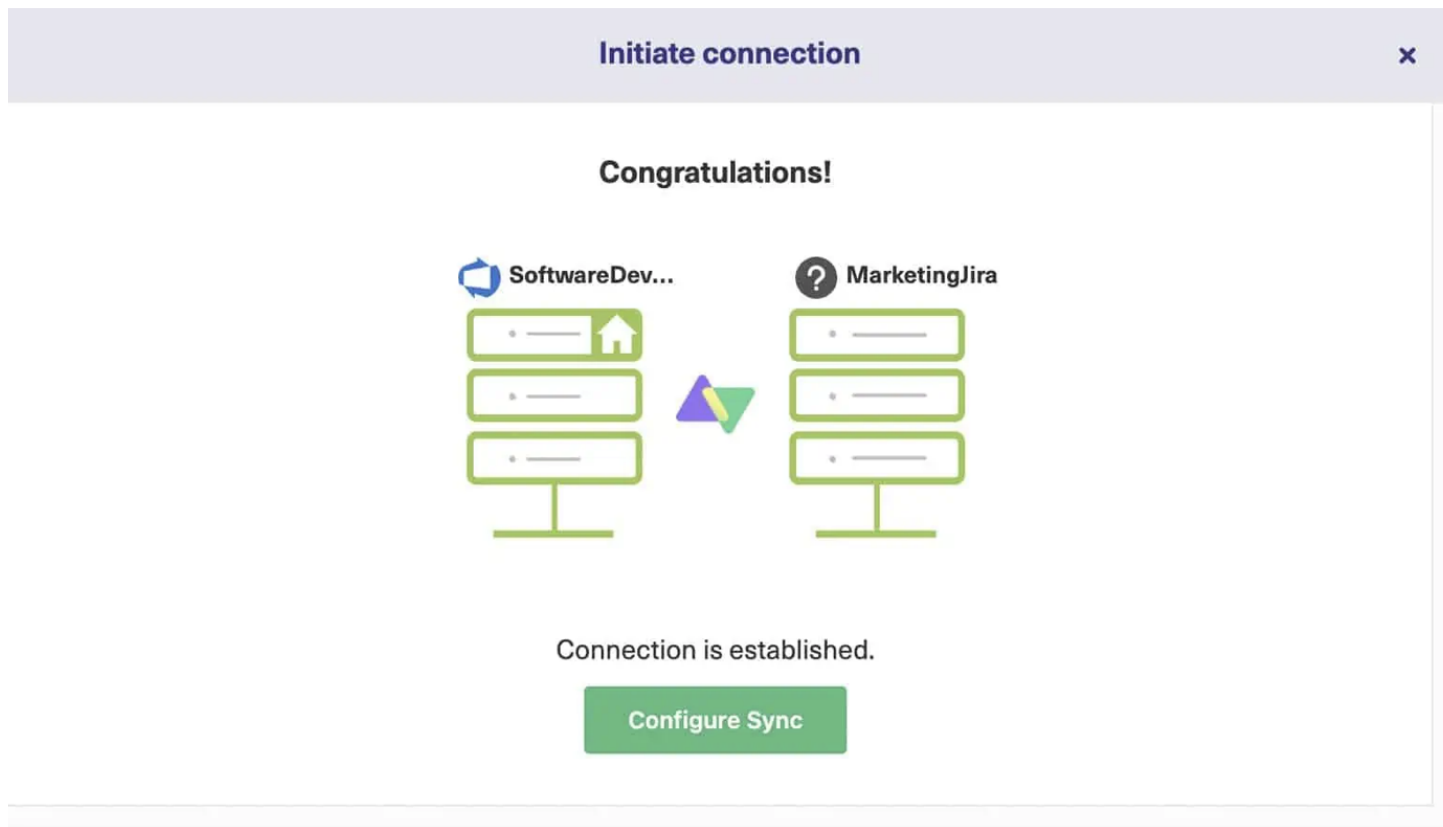
Description

[< Previous](#) [Next](#)

You can name each side of the connection, and the names are combined to generate a connection name. Easily adjust this if you like.

After clicking “Next”, you are required to confirm the admin access to the Jira instance by clicking the “Initiate” button. You will be redirected to the Jira instance for verification.

Once the connection is successfully established, you can move ahead to “Configure Sync”.



Next, select projects on both the Azure DevOps and the Jira side between which you want the synchronization to happen. The “Sync Method” allows you to decide whether you want manual or automatic sync.

» SoftwareDevelopmentADO_to_MarketingJira

● Active

[← Back to Connections](#)

Scope

Rules

1

2

Define the context for the synchronization: what issues you want to sync and how to start the synchronization process.



SoftwareDevelopmentADO



MarketingJira

Select Project

Select Project | v

Filter entities

Sync method

Manual → | v

Manual ← | v

Select Project

Select Project | v

Filter entities

Next

For both the projects, you can further “Filter entities” and configure the way your synchronization works.

You can filter work items or issues based on their Type, Priority, etc. Click “More” for advanced filters like “Assigned to/ Assignee”, and “Created by/ Reporter”.

Click “Save” for confirming the changes. You can “Cancel” them anytime.

Filters

Select conditions to filter entities for synchronization:

Tags Tags v	Type Task x v	State State v
Priority Priority v	Title Title	Description Description
Assigned To Email or Account ID v	Created By Email or Account ID v	Area Area
Iteration Iteration		

[Cancel](#) [Save](#)

The “Rules” tab provides you with default mappings like “Type” in ADO being mapped to “Issue Type” in Jira. You can edit these default mappings by clicking the edit icon in front of every entry. You can also delete them if you want to.

To add additional mappings click the “Add Mapping” button.

» **SoftwareDevelopmentADO_to_MarketingJira** ● Active [Back to Connections](#) [Publish](#)

Scope 1 ————— 2 Rules

Configure the synchronization behavior with the help of field mappings and script rules.

[Expand all](#) [Collapse all](#) [+ Add mapping](#)

Order	SoftwareDevelopmentADO	Sync direction	MarketingJira	
1	Tags	↔	Labels	
2	> Type	↔	Issue type	
3	> State	↔	Status	

Here, you can add additional mappings you want between Azure DevOps and Jira.

Add mapping

<p>SoftwareDevelopmentADO *</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">Created By ▼</div> <p>The user value of Created By is based on the received text value from Hiring manager.</p> <p>If no matching value *</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">Do nothing ▼</div> <div style="border: 1px solid #ccc; padding: 5px; background-color: #f0f0f0;"><p>Set a default value</p><p>Report an error</p><p style="background-color: #28a745; color: white; padding: 5px;">Do nothing</p></div>	<p>Sync direction</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 0 auto; width: 60px;">← → ▼</div>	<p>MarketingJira *</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">Hiring manager ▼ ⓘ</div> <p>The text value of Hiring manager is based on the received user value from Created By.</p> <p>If no matching value *</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">Do nothing ▼</div>
---	--	---

CancelSave

Select the entities to be matched on both sides. Also, select the “Sync direction”. You can even specify what must happen if no matching value is found. You can either “Set a default value”, “Report an error” or simply “Do nothing”.

Click “Save” to confirm changes, else go back to the “Rules” tab by canceling the window. Once all this is done, don't forget to “Publish” these changes.

We just saw how easy it is to configure your connection in Visual mode without having to code. Also, you can edit the connection anytime, by clicking on the “Edit Connection” icon at the right of the connection name in the “Connections” tab.



Continue with the Script Mode

Moving forward with the Script Mode, you name the connection just as discussed for the “Visual Mode”.

There’s also a description field here, which is optional. It is very useful to fill this in as later, you may have many connections so the more information, the better.

Fill the form in and try to make it as descriptive as possible. Then click “Next” to continue.



Next, you need to pick a project on the Azure DevOps side, that will be synchronized with the other platform. Choose one from the drop-down list, and then click the green “Initiate” button.

Initiate connection ×

On the “JiraScript” side, you (or their application administrator) need to **Accept the Invitation.**

Use the following invitation code:

[Copy invitation code](#)

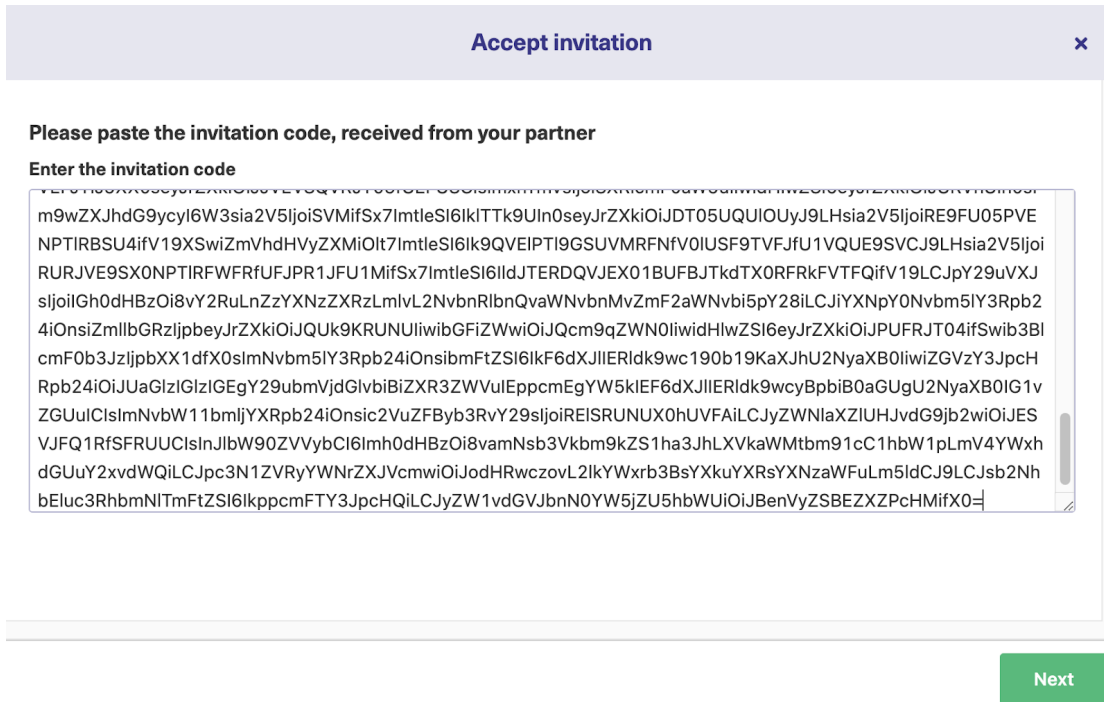
Done

Exalate now generates an invitation code that you need to copy and paste from one side into the other. Click “Copy invitation code” to copy it to your clipboard. It is best to paste it somewhere safe at this point, such as a text file.

In Jira, click the “Settings” cog icon and select “Add-ons”. Click “Connections” from the left-hand Exalate menu.

Jira’s connection screen looks pretty much the same as in Azure DevOps. One advantage of Exalate is it gives you a similar UI across multiple platforms, so once it’s installed and you know how to use it, you can connect to the other platforms easily.

Click the “Accept invitation” button in the top right.



The “Accept invitation” screen has a large field for you to paste into the code you just generated in Azure DevOps. Do that, and click “Next”.

You’ll see a few options screens that look similar to the ones you saw when creating the connection. The first screen will ask you to select a project from the drop-down list to use for the synchronization on the Jira side. After selecting the project, click “Confirm”.

After clicking the “Confirm” button, you’re done setting up the connection.

Now you can move on to the next steps, where you’ll learn how to control exactly what your integration does.

Step 4 - Configure Your Connection to Determine What Gets Shared

Exalate lets you modify each connection to decide exactly what gets copied to and from each side.

It does that using scripting. If you are used to working with scripting or programming languages, you should find it very straightforward, but anyone can make changes with a little practice.

In the Visual Mode

We have already discussed how to edit the “Scope” and “Rules” for Visual Mode connection in the “Configure Sync” option in step 3.

These scopes and rules can be modified once again by clicking the “Edit Connection” option shown next to the Connection name.

In the Script Mode

Exalate uses scripting to edit the connections established using the Script Mode. If you are used to working with scripting or programming languages, you should find it very straightforward, but anyone can make changes with a little practice.

The Sync rules use the 'Groovy' scripting language. So if you're familiar with that, this shouldn't be too difficult. Otherwise, remember that you need to get everything right when working with a scripting language, but the rules aren't too hard to follow when you get used to them.

Exalate is robust enough to handle mistakes, so don't be afraid to experiment. If you get it wrong, you can always go back again and fix it.

To start making changes, have a look for your connection in the list, move the mouse over it, and click the edit icon which appears. This guide uses AzureDevOps, but the process is similar in Jira.

You can also click the remote button to go to the other side of the connection or click the three dots to either delete or deactivate the connection.

On the "Edit Connection" screen, there are 4 tabs: "Rules", "Triggers", "Statistics" and "Info".

We'll look at the "Rules" in a second and the "Triggers" in the next step.

```
» Azure DevOps_to_JiraScript
Active
Back to Connections Publish

Rules Triggers Statistics Info

- Outgoing sync
1 replica.key = issue.key
2 replica.type = issue.type
3 replica.assignee = issue.assignee
4 replica.reporter = issue.reporter
5 replica.summary = issue.summary
6 replica.description = issue.description
7 replica.labels = issue.labels
8 replica.comments = issue.comments
9 replica.resolution = issue.resolution
10 replica.status = issue.status
11 replica.parentId = issue.parentId
12 replica.priority = issue.priority
13 replica.attachments = issue.attachments
14 replica.project = issue.project
15
16 //Comment these lines out if you are interested in sending the full list of versions and components of the source project.
17 replica.project.versions = []
18 replica.project.components = []
19
20 /-
21 Custom Fields
22 replica.customFields."CF Name" = issue.customFields."CF Name"
23
24 /-

- Incoming sync
1 = !({firstSync})
2 = issue.projectKey = "10"
```

The statistics tab gives you info on what items are being synced and also tells you when synchronization last took place. The info tab gives you a few details about the connection, including the URL of the other side. These tabs are useful if you want to confirm your connection is working as intended.

For now, click on the “Rules” tab.

At the top is a list of outgoing sync rules. These show how items in Azure DevOps are mapped to Jira. The incoming sync rules show how information from Jira is mapped to Azure DevOps items.

If you want to change what AzureDevOps sends over the connection, you edit the **outgoing rules**. The **incoming rules** let you change how the data from the connection is mapped onto the synchronized items.

Of course, if you are looking at this screen in Jira, those relationships will be reversed.

Looking at the rules, you may be wondering how to edit them. All you do is click in either box, edit the text, and then click the green “Publish” button in the top right when you’ve finished. Check this [article on sync rules](#) to learn more.

Looking at the outgoing rules, you can see that there are many fields like **replica.key = workItem.key**. That means the key field can be retrieved by the other side of the connection using the same name.

If you don’t want to share any particular field, you can remove it, or replace it with something else.

To remove it, simply delete the line. Or you can comment by adding ‘//’ in front of the particular line so it's ignored for synchronization.

If you want to share a specific value you could, for example, replace **replica.status = workItem.status** with **replica.status = “from Azure DevOps”**.

The Jira side will see incoming items with the status field set to “from Azure DevOps”.

In a similar way, you could give a specific value to items in the incoming sync that you get from Jira. Perhaps you could change **`workItem.description = replica.description`** to **`workItem.description = "from Jira"`**.

If you want to assign items from Jira to a specific person, you can add a line to the incoming rules that says **`workItem.assignee = "Peter"`**.

You can also see commented code.

Lines that begin with `/**` and sections bound by `/**` and `*/` are comments. Comments are there as information and don't affect the synchronization. You can copy and paste from a commented section to an uncommented section to activate particular lines, as well as remove the `/**` from the start of commented lines.

You can also add `/**` to the start of lines if you want the particular sync to be ignored. That's a useful alternative to deleting them, as it means you can easily reactivate them by removing the comment slashes.

There are all sorts of ways you can edit the sync rules to make them more useful to you. After you've been working with your integration for a while, you might have new ideas for how to refine it, so it can be worth coming back to make further edits and improve your workflow.

Step 5 - Set Up Automated Synchronization Triggers

Next, click on the "Triggers" tab. This screen lets you create triggers that decide which items are shared by the connection. The triggers are written in work item query language or [WIQL](#) for Azure DevOps and in JQL (Jira Query Language) for Jira.

In addition to using the "Edit Connections" page to reach the synchronization triggers, there's also a dedicated entry for them in the left-hand menu. The interface, in that case, is almost the same, with the exception that you can see the connection as a separate entry in that list.

Note: The “Triggers” tab for Visual Mode connections can be accessed by clicking the left submenu in the “Exalate” app.

Click the white “Create trigger” button to create your new trigger. There are several fields on the screen. At the top, you can select the type of entity the trigger will deal with. In the large field with “If” at the top, you enter a WIQL query that selects the items you want.

For example, `[Work Item Type] = 'Task'`, will select items with the type “Task”. You can base the query on any of your item’s fields. You can also create multiple conditions for a trigger using logical operators.

If you write `[Work Item Type] = 'Task' AND`

`[System.AssignedTo] = 'Kirsty'`, you’ll sync items that meet both of those conditions.

Have a look at the [documentation](#) to get an idea of what you can do, and try modifying the rules to work with the fields you want.

Below that, there’s a notes field. Here you can write a description explaining what the trigger does, and why it is there. As with the connection description, adding a detail here can potentially make life easier for you later.

Triggers Create Trigger

Entity type	When	If	Then sync via Connection	Status	Action
Issue	Events: create/update	labels = demo	ADO_to_jira	<input checked="" type="checkbox"/>	⋮
Issue	Events: create/update	issuetype=Task	azuredevops_to_jira	<input checked="" type="checkbox"/>	⋮
Issue	Events: create/update	labels=gio	gioz_to_giojscrip	<input checked="" type="checkbox"/>	⋮
Issue	Events: create/update	labels=sync	Jiracloud_to_jiraserver	<input checked="" type="checkbox"/>	⋮

There's also an "Active" switch, which you'll need to activate if you want the trigger to work. This switch makes it easy to turn the trigger on and off without having to delete or redo it.

When you're ready, click the "Add" button, and your new trigger will be created, and added to the list.

You can add as many triggers as you like, giving you fine control over what kinds of items are shared. They can be activated or deactivated easily, and modified whenever needed.

Step 6 - Start Synchronizing Tasks

Now that your connection is ready and you know how to configure it, work items will be shared between the platforms according to the rules you have defined. If you create new items that match your rules, they will be synced, as will existing rules that meet the same criteria.

Exalate will show you how many items have been synced on the connections screen. Synchronization doesn't happen immediately, so please wait a few minutes if you don't see the exchange take place.

Common Use Cases for a Jira Azure DevOps Integration

Jira and Azure DevOps are both versatile platforms adaptable to a wide range of business needs. Let's take a look at a few situations where a Jira Azure DevOps integration could be useful. These examples will give you ideas, but your situation may be different. Hopefully, you will have your own ideas for how integration can work.

Backend and Frontend Developers

In medium to large-sized software projects, different teams usually focus on different parts of the product. You may have a team of backend developers handling your database and business logic, while frontend developers focus on the website, apps, and user experience.

These teams will sometimes have to handle common issues, and will both collect information on them. Sometimes a bug or issue might need to be diagnosed and it may not be clear where it originates. Perhaps a new feature needs work done on the backend and frontend. In these cases, information on relevant issues can be synchronized.



You can copy the whole issue from one system to another, or copy particular fields. You can use labels, or any other field to define which issues you need to share so that both teams are able to make information available to each other as needed.

Each team can decide what they send, and decide how incoming information is entered into their own system.

Customer Service and Engineering Teams

Your customer service team handles customer feedback. A lot of that will include problems and bugs that need to be solved by your engineers. The customers will want to know about any solutions or fixes that can be applied, so will need some feedback.

You don't necessarily want the engineers talking to customers directly, so here the customer service team acts as a filter, handling most issues themselves, and passing on problems they can't solve. They may modify feedback to make it more customer-friendly, and store it so they can handle the same problems again without needing to contact engineering.

An integration can help make sure that each team gets the data presented in the way it needs and enables them to take advantage of the other team's data when they need it. You can send the fields they need to be aware of and keep the information they don't need out of their way.

Marketing and Design Teams

Your marketing team conducts research to find out what your customers want. Your design team will assess what they ask for and try to deliver it in your product. As with the previous situation, you are collecting data from customers and passing it on to a team that needs to use the information.

Here, the designers don't need to provide customers with feedback on any issues they raise, but the marketing team might need to know what ideas the design team has to solve problems and conduct further research on how customers react to them.



There may also be things customers want that aren't possible for budget or technical reasons, and it is useful for marketing to know this, so they can focus their research on more useful areas.

Your integration can help handle the flow of information between the teams. The marketing team's results get sent to the designers and the designers can pass their findings back to the marketers. They can do this in the form of new issues, or comments on the existing feedback.

Conclusion

With a Jira Azure DevOps integration, your teams can share information effortlessly. With an effective integration tool, you can let teams get on with things without having to worry about downtime or problems. Your teams can both make changes and control what the integration does, making it evolve along with their needs.

As well as Jira and Azure DevOps, you can use Exalate to integrate other platforms, such as ServiceNow, GitHub, Zendesk, and more. The more connected your teams are, the easier it is to get them all pushing in the same direction.

You may also want to read [this ebook](#) on cross-company integration to learn more about sharing information between companies.

Recommended Reads:

- [Jira to Jira Integration: The Comprehensive Guide to Jira Sync](#)
- [How to Set up an Azure DevOps Salesforce Integration](#)
- [Jira ServiceNow Integration: How to Set up an Integration in 6 Steps](#)
- [How to Set Up an Azure DevOps ServiceNow Integration](#)
- [How to Set up an Azure DevOps GitHub Integration](#)
- [Jira Integrations: Integrate Jira and Other Systems Bidirectionally](#)