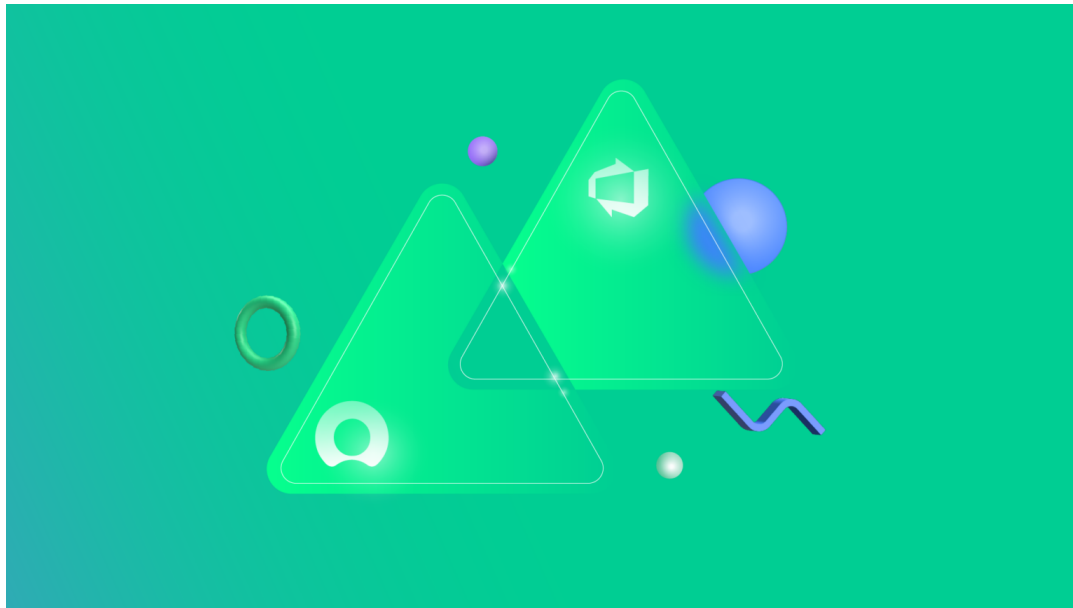




# How to Set Up an Azure DevOps ServiceNow Integration: The 2023 Step-by-Step Guide



## Table of contents

### Why Integrate Azure DevOps with ServiceNow?

- What is Azure DevOps?
- What is ServiceNow?
- Why Integrate Azure DevOps with ServiceNow?

### How to Choose the Right Technology for Setting up an Azure DevOps ServiceNow Integration

- Decentralized Integration (Autonomy)
- Reliability
- Flexibility

### How to Set up an Azure DevOps ServiceNow Integration (a Step by Step Process)

- Step 1: Install Exalate on Azure DevOps
- Step 2: Install Exalate on ServiceNow
- Step 3: Set up a connection between Azure DevOps and ServiceNow
- Step 4: Configure Your Connection to Determine What Information Gets Shared
- Step 5: Set Up Automated Synchronization Triggers

### Common Pitfalls to Avoid after Setting up the Azure DevOps ServiceNow Integration

- Role Clarification



- Too Many Messages

Conclusion



Keeping track of the vast quantities of information circulating around your business can be a headache, but also an opportunity if you know how to sync data and collaborate effectively with other teams. Let's say you're working in Azure DevOps and you need to integrate with another team in ServiceNow. An Azure DevOps ServiceNow integration is what can make everything run way more smoothly.

Connecting your teams effectively is definitely a big challenge. So in this blog post, we'll walk you through a step-by-step process to learn how to integrate two of the most popular work management systems with the least fuss possible.

### Here is an overview of what we will cover in this blog post:

- [Why Integrate Azure DevOps with ServiceNow?](#)
- [How to Choose the Right Technology for Setting up an Azure DevOps ServiceNow Integration](#)
- [How to Set Up the Azure DevOps ServiceNow Integration \(a Step by Step Process\)](#)
  - [Continue with the Basic Mode](#)
  - [Continue with the Script Mode](#)
- [Common Pitfalls to Avoid after Setting up the Azure DevOps ServiceNow Integration](#)



## Azure DevOps ServiceNow Sync Guide

Learn how to achieve a seamless integration between Azure DevOps and ServiceNow instances, step-by-step.

[GET THE EBOOK](#)

## Why Integrate Azure DevOps with ServiceNow?

### What is Azure DevOps?

Azure DevOps is an outstanding tool that can be used for a range of purposes, including managing servers and websites, as well as product development.

It is hugely popular with engineers and makes running projects much easier. Packed with features, it includes versioning, project and requirements management, build automation, and is designed to help with the full application lifecycle.



## Azure DevOps

There are all sorts of ways to use it, but it is likely to be used by technically adept teams working on various kinds of software projects. There are two versions of it. It can be used in the cloud or teams can host it on their own server.

It has a [marketplace](#) full of apps that let you add features and integrate them with other platforms. It can also be extended directly via its SDK.

## What is ServiceNow?

ServiceNow also has a strong range of capabilities, but where AzureDevOps caters to developers, ServiceNow is mostly focused on service and support.



Using ServiceNow is a great way to handle customer requests and organize the way your customer support team interacts with clients. It lets you keep track of incidents and problems and allows your team to stay on top of all the information generated by customers.

## Why Integrate Azure DevOps with ServiceNow?

Often your teams will need to share issues, and the specifics of what they need to know will differ. Keeping them on the same page without duplicating or losing information is quite a challenge. So a tool that can take care of it automatically will be a huge asset.



The customer service team usually becomes aware of problems that the engineers need to know about, and will pass the issues on to the support team. The engineers will fix these problems afterward, and the customer service team can then pass the result on to the clients.

This data can be exchanged manually but there are several problems with doing that. It's time-consuming, some issues may not be passed on, duplicate information may be shared and data might be passed in the wrong format, or with the wrong level of detail for the person receiving it to use.

With an automated system, you can ensure that data is transferred between teams when it is captured by either platform. You can set conditions for the exchange, meaning you can filter what is shared.

By letting an integration tool take control of the process, you can save yourself work and ensure data is exchanged consistently under the conditions you specify.

## How to Choose the Right Technology for Setting up an Azure DevOps ServiceNow Integration

There is an increasing number of software integration solutions available to help you integrate different platforms. When picking one, it is important to clarify exactly what problems you need to solve. There are three particular criteria to bear in mind when choosing a solution for an Azure DevOps ServiceNow integration.

### Decentralized Integration (Autonomy)

Teams need to be able to change the configuration of their integrations independently. It is inconvenient to have to double-check everything with other teams, particularly if they are not in the same company, or don't speak the same language.

The systems should also allow teams to maintain privacy, so they only share the information they want to. If a solution can guarantee your autonomy, then there'll be no need to leave your familiar environment even when integrating with other companies and other platforms.



## Reliability

The integration solution must be able to cope with downtime or problems on either side of the connection and to recover when things are back up. Outages happen when working online, and the system should be capable of rescheduling data exchange without anything being lost, and without engineers needing to become involved.

## Flexibility

The teams on either side of the connection will likely discuss their requirements when the integration is set up, but these will change over time. They may want to expand the range of data shared or tune it, as the teams grow and requirements change.



They may also change the format of the information entered into their system. The integration must be able to handle these changes and adapt to the needs of users on each end of the connection.

For our integration, we've chosen [Exalate](#), a solution built to solve these three particular challenges. It is designed to be robust, flexible, and to allow your teams to operate independently.

Now let's look at how to set up the integration.

## How to Set up an Azure DevOps ServiceNow Integration (a Step by Step Process)

To set up our integration, we'll go through several steps. First, we'll install Exalate on both platforms. Then we'll create a connection between them. After that, we'll look into the connection settings in more depth to configure what data is being sent and received. We'll also see how to control the conditions for data exchange.

Are you a fan of tutorial videos? Then watch this step-by-step Azure DevOps ServiceNow integration tutorial.

<https://youtu.be/laYomtZtVOs>

### Step 1: Install Exalate on Azure DevOps

First of all, we need to install Exalate on our Azure DevOps instance. For more detail, read the [Exalate documentation on Azure DevOps](#).

You can install Exalate via the Microsoft Azure marketplace which will deploy the node onto the Exalate cloud. Alternatively, you can also install Exalate via Docker, in which case, you can go ahead and read [this](#).

For the purposes of this guide, we'll install directly from the marketplace. Read [this doc](#) for more information.

First of all, make sure you are logged in as an admin. If you look at the bottom left of the screen after logging in, there's a cog icon and a link saying 'Organization settings'. You might need to scroll down if you can't see it. Click it. On the screen that follows, click 'Extensions' in the left-hand menu.

The screenshot shows the Azure DevOps Organization Settings interface. The left-hand navigation menu is expanded to show the 'Overview' option under the 'General' section. The main content area displays the 'Overview' settings for an organization, including fields for Name, Privacy URL, Description, Time zone, and Region. A 'Save' button is visible at the bottom of the settings section. Below the settings, the 'Organization owner' is listed as John Doe with a 'Change owner' button.

**Organization Settings**

Search Settings

**General**

- Overview
- Projects
- Users
- Billing
- Auditing
- Global notifications
- Usage
- Extensions
- Azure Active Directory

**Security**

- Policies
- Permissions

**Boards**

- Process

**Pipelines**

- Agent pools
- Settings
- Deployment pools
- Parallel jobs

**Overview**

Name

Use the new URL: [https://dev.azure.com/Placeholder](#)

[Learn more about URLs](#)

Privacy URL

[Learn more about the Privacy URL](#)

Description

Add organization description

Time zone

UTC

Region

East Asia

[Learn more about the Region](#)

Save ⓘ Changes made will affect all projects and members of the organization

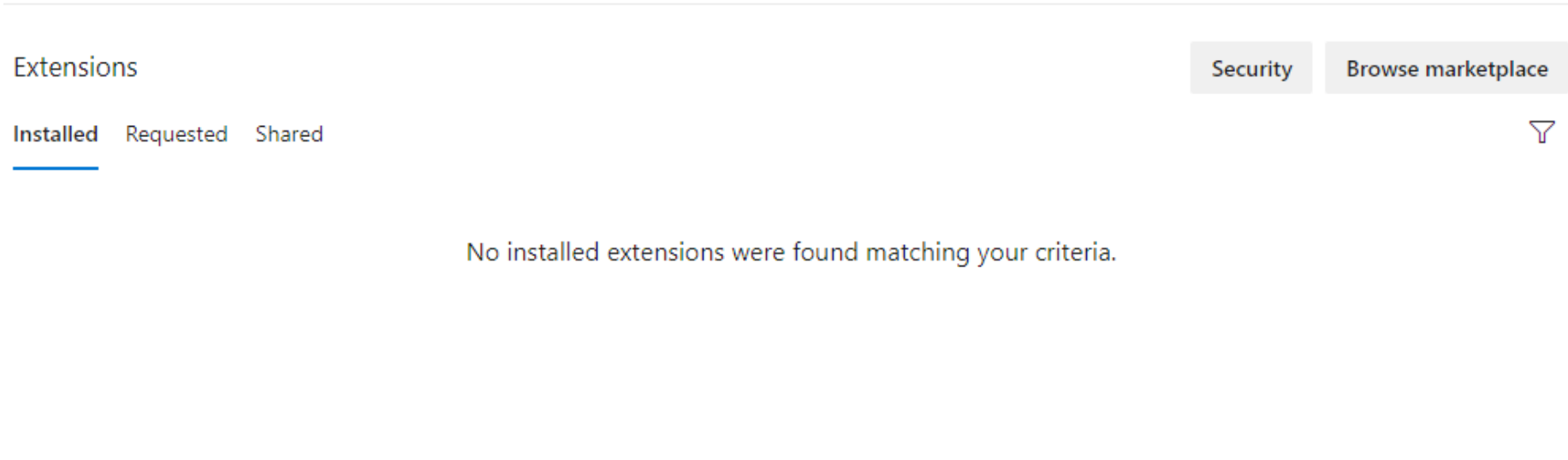
**Organization owner**

John Doe  
[john.doe@placeholder.com](#)

[Learn more about the organization owner](#)

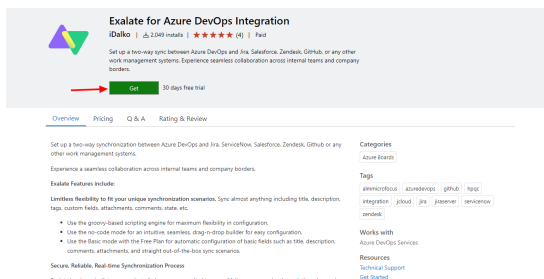
Change owner

On the extensions screen, click the 'Browse marketplace' in the top right.

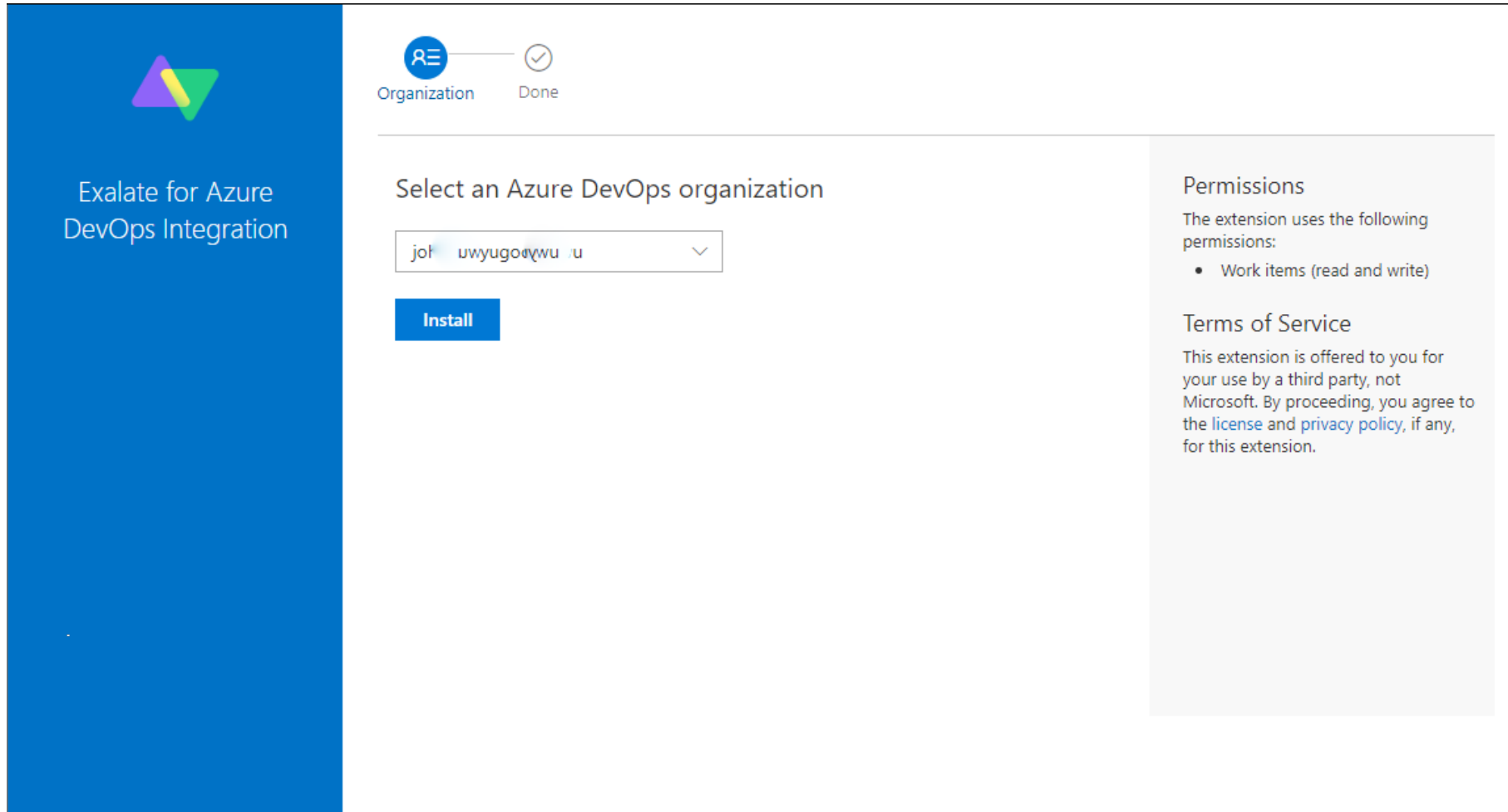


In the marketplace search field, look for Exalate for Azure DevOps.

Click it and you'll be taken to its marketplace page. Click the green 'Get' button to install it.



Next, you have to pick an organization to attach your installation to. Choose one from the drop-down list and click the blue 'Install' button.



Organization Done

### Select an Azure DevOps organization

joh...uyyugoa...u

**Install**

#### Permissions

The extension uses the following permissions:


- Work items (read and write)

#### Terms of Service

This extension is offered to you for your use by a third party, not Microsoft. By proceeding, you agree to the [license](#) and [privacy policy](#), if any, for this extension.


You'll also need to request an [evaluation license](#) to use Exalate. You can do that by clicking "Exalate" in the left-hand Azure DevOps menu, then clicking "License details" in Exalate's menu.

**i** Your Exalate is not under license. You are able to synchronize only when your partner is paying for the synchronization with a network license.




**30-DAY TRIAL**  
Take Exalate for a test-drive with a fully functional license for 30 days.

**MOST POPULAR**



**BUY INSTANCE LICENSE**  
Purchase a standard Exalate instance license. Each side of the connection will require a valid license.



**BUY NETWORK LICENSE**  
Don't want to connect with just one other instance? Choose the network licensing model to connect with multiple trackers.

License Key [🔗](#)

Click on the 30-day trial image and then enter your email into the popup that appears. You'll get an email with an evaluation code, which you should copy. Back in Exalate, in the license details area, click the green "License Key" button at the bottom left.

**30-DAY TRIAL**  
Take Exalate for a test-drive with a fully functional license for 30 days.

**MOST POPULAR**

**BUY INSTANCE LICENSE**  
Purchase a standard Exalate instance license. Each side of the connection will require a valid license.

**BUY NETWORK LICENSE**  
Don't want to connect with just one other instance? Choose the network licensing model to connect with multiple trackers.

[License Key](#)

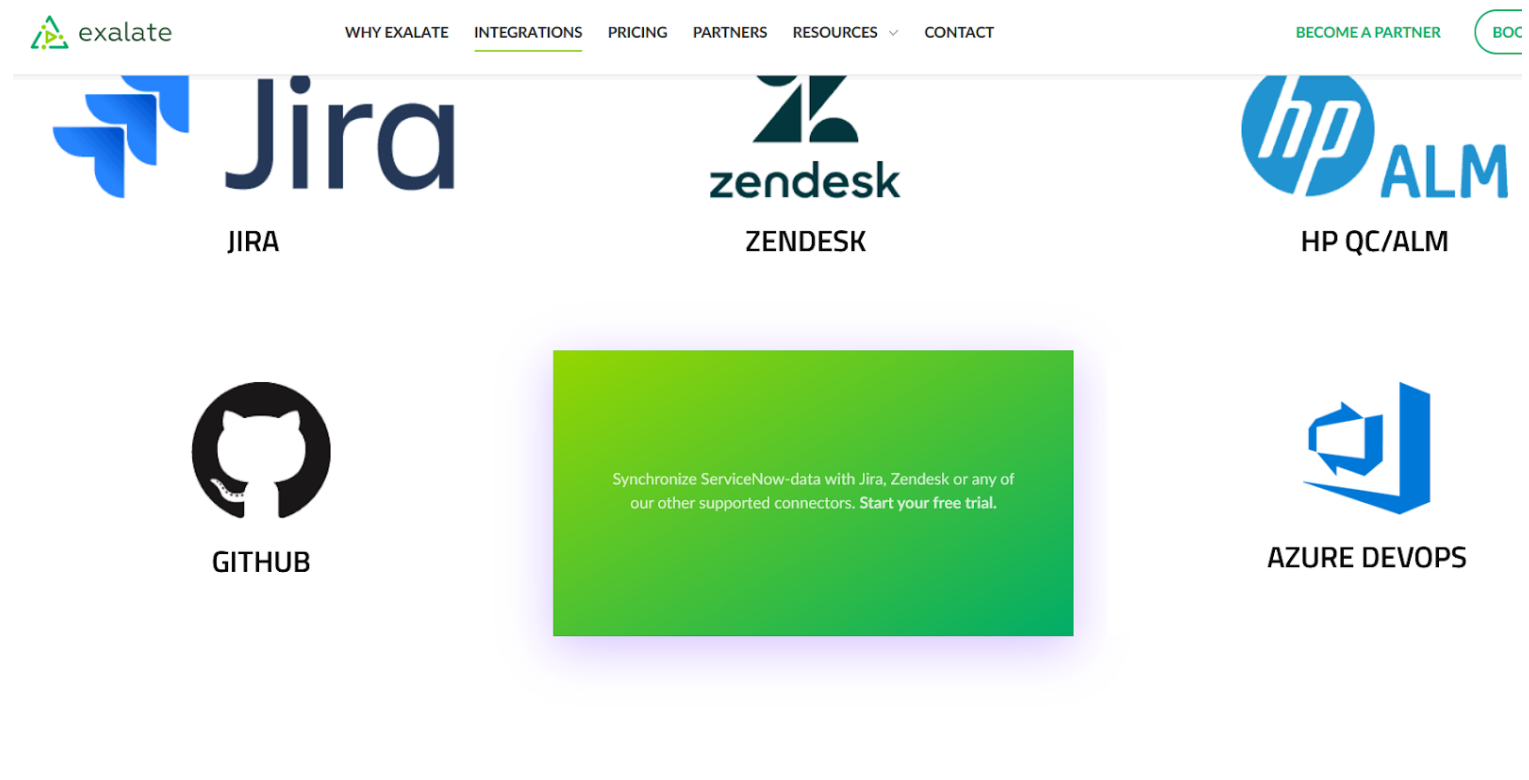
[Documentation](#) [Support](#) [Report a bug](#)

Paste your key in and click “Update” to register your license.

You’re now all set on Azure DevOps. When we get to step 3, come back here to configure your connection.

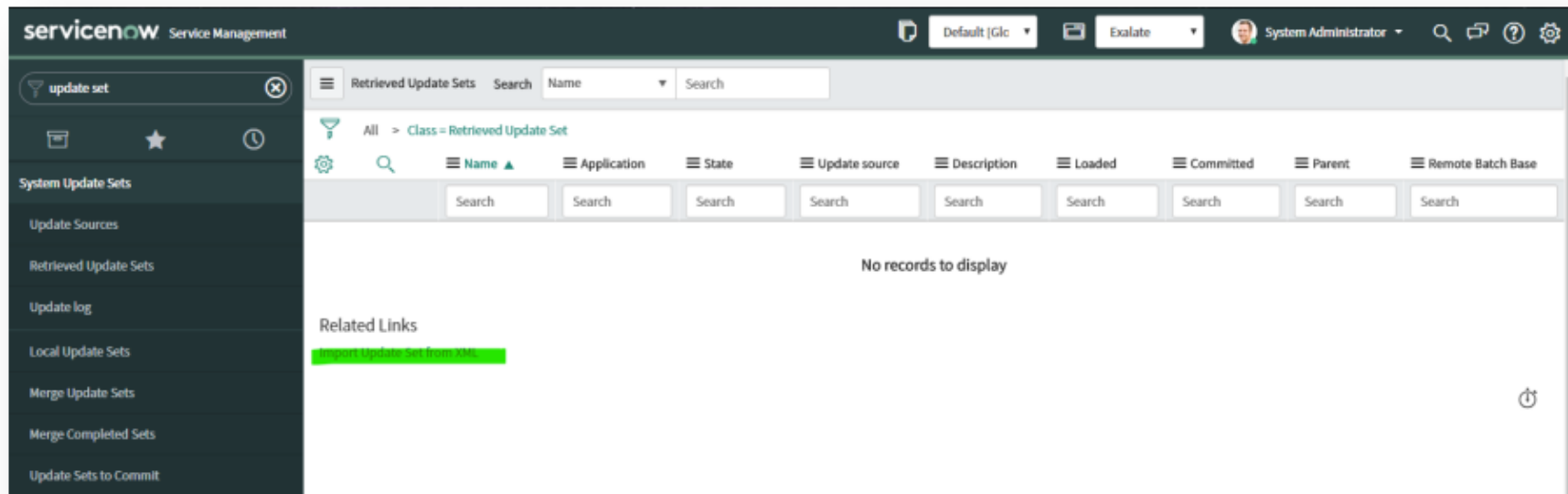
## Step 2: Install Exalate on ServiceNow

Now, we’ll install Exalate on our ServiceNow instance. For more detail, read this [documentation](#). There are a few different ways to do it, this guide will take you through one of them.



Firstly, request an Exalate node by going to [this page](#). You'll need to click ServiceNow, then fill in your details on the form that pops up. You'll get an email soon afterward, containing the URL of your node.

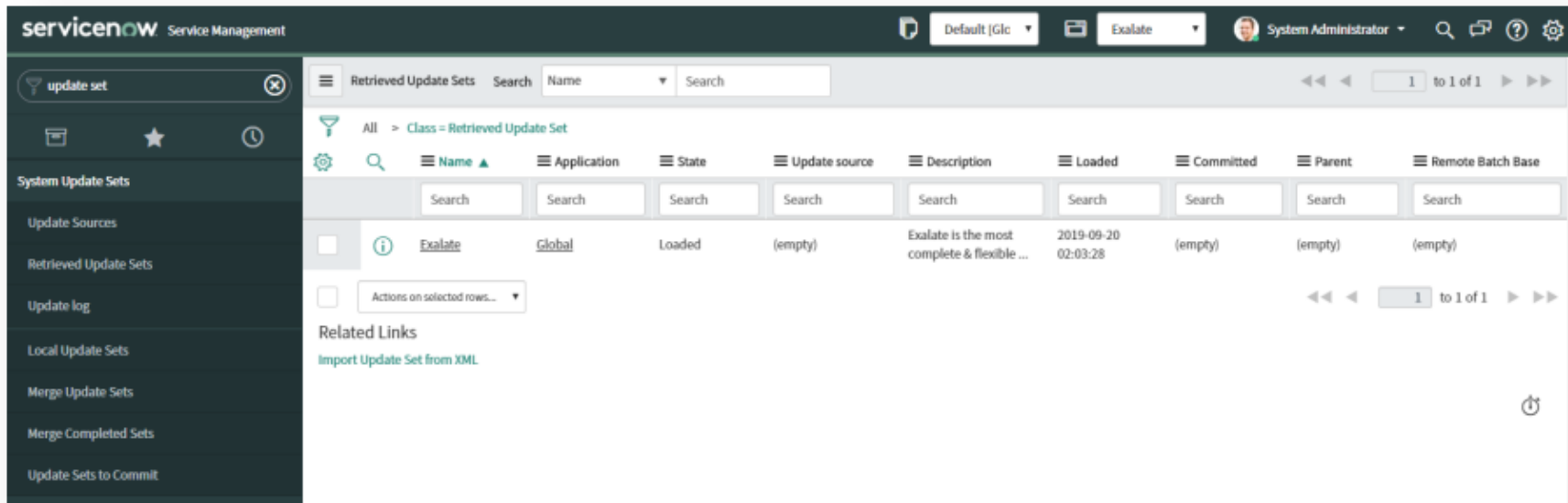
Next, download an XML 'update set' from Exalate. The information in this file tells ServiceNow how to work with Exalate. Download it [here](#).



Once you've stored the XML file somewhere safe, log in to your ServiceNow account. In the left-side menu, click the 'All applications' icon, if it isn't already selected.



Look for the 'System Update Sets' entry and click to open it. This menu can get very crowded, so use the search field to locate it if you have trouble finding it. Then click 'Retrieved Update Sets'.



Under the 'Related Links' heading, click the text that says 'Import Update Set from XML'. Next, click the 'Choose File' button and select the XML file downloaded earlier. Click the 'Upload' button to complete the process.

The Exalate XML file will be uploaded and listed. You should be able to see it. Click the file and then click the 'Preview Update Set'. If it asks you to update your setup, click the 'Accept remote update' button to do so.

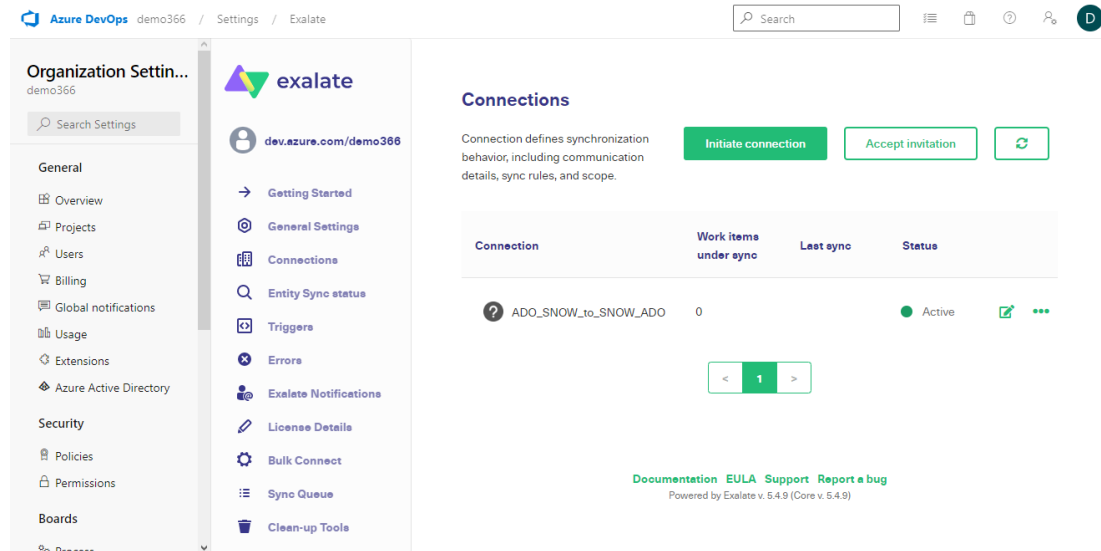
Once that's done, click the 'Commit Update Set' button. Exalate is now installed on ServiceNow. Next, we'll go over how to set up a connection between our platforms.

### Step 3: Set up a connection between Azure DevOps and ServiceNow

To create a connection between Azure DevOps and ServiceNow, we log into one platform and create an invitation, which we then paste into the other.

This step and most of what follows is essentially the same for these and other platforms, so Exalate allows you to connect multiple platforms very easily once you are familiar with it.

We can create our invitation on either platform. For this example, we'll start in Azure DevOps. Find Exalate in the 'Organization Settings' under 'Extensions'.



The screenshot shows the Azure DevOps interface for the 'demo366' organization. The left sidebar is open to 'Organization Settings' > 'Extensions'. The main content area displays the 'exalate' extension settings. At the top, there are buttons for 'Initiate connection', 'Accept invitation', and a refresh icon. Below this is a table of connections:

Connection	Work items under sync	Last sync	Status
ADO_SNOW_to_SNOW_ADO	0		Active

At the bottom of the table, there is a pagination control showing '1' of 1 items. Below the table, there are links for 'Documentation', 'EULA', 'Support', and 'Report a bug'. The footer of the extension page reads 'Powered by Exalate v. 5.4.9 (Core v. 5.4.9)'.

If you aren't already there, click 'Connections' in Exalate's left-side menu. You should see either a list of existing connections or a message saying you don't have any. Either way, click the green 'Initiate connection' button to get started.

On the next screen, enter the URL of your destination instance. Again, that will be our Exalate ServiceNow node. After you enter the URL, a few more fields will appear.


### Initiate connection ×

**Destination instance URL** ⓘ

 ✓ I don't have a URL


**Choose the configuration type**

**FREE PLAN**

 **Basic**


- Automatic configuration of basic fields
- Sync rules cannot be edited
- Recommended for use cases of basic complexity

**30-DAY TRIAL** ⓘ

 **Visual**

- Low-code, visual interface
- Configure both sides of the connection using a single interface
- Recommended for use cases of basic to intermediate complexity

**30-DAY TRIAL**

 **Script**

- Groovy-based scripting
- Configure each side of the connection separately
- Recommended for use cases of basic to advanced complexity

Next

You will be required to select the configuration type of your connection. Exalate offers two types: Basic and Script.

The Basic mode has predefined mappings and sync rules that cannot be modified. It is suitable for use cases of basic complexity. Exalate offers a [Free Plan](#) that comes with the Basic mode connection offering up to 1000 free syncs per month. It is a great way to see how Exalate works firsthand.

The Script mode allows you to configure each side of the connection separately using Groovy based scripting. It can be used for advanced integration cases. With the Script mode you can generate your own rules and share almost any kind of information with the other side.

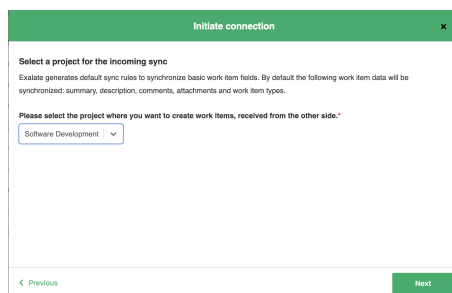
You can even [upgrade](#) your Basic connection to a Script one anytime you want!

Let us have a look at both these modes one by one.

### Continue with the Basic mode

Click 'Next' when you select 'Basic' on the screen above.


Select the project on the Azure DevOps side. This will be the project you want the ServiceNow entities to sync into. Choose the one you want from the dropdown list and hit 'Next'.




Exalate will then ask you to verify if you have admin access to the other side. In our case, it is ServiceNow. Click 'Yes, I have admin access' if you have access. In case you don't have admin access, an invitation code will be generated for you. Copy this invitation code and paste it manually on the ServiceNow side. We will see how to do this in the Script mode.

**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 dont have admin access**

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

[← Previous](#)

**Initiate**

Hit 'Initiate' after.

After a quick verification is done on the ServiceNow side, your Basic mode connection between Azure DevOps and ServiceNow is established. You can now move on and sync your first Incident or Work Item.

Here, we have entered a work item number to start the synchronization from Azure DevOps.



**Connection established successfully** ✓

Sync your first work item to see how it works.

Please enter a work item key from the project **Software Development** to proceed



If you navigate to the ServiceNow side right now you will come across the same screen asking you to enter the Incident key to sync.

In either case, click 'Exalate'.

Wait for some time for the synchronization to happen.

The screenshot displays the Azure DevOps user interface. On the left, the 'Organization Settings' sidebar is visible, with 'Exalate' selected at the bottom. The main content area shows a modal window titled 'Exalate' with a close button (X). Inside the modal, the heading 'Syncing work item' is centered. Below the heading, there are two columns representing the source and target systems. The left column is labeled 'azure\_demo...' and contains three green rectangular boxes representing work items. The right column is labeled '? ven03945' and also contains three green rectangular boxes. A purple and green arrow points from the source to the target. Below the work items, a progress bar shows five stages: 'Preparing' (green circle with a checkmark), 'Sending' (green circle with a checkmark), 'Waiting for remote' (green circle with a refresh icon), 'Finalizing' (grey circle), and 'Synchronized' (grey circle). At the bottom right of the modal, it says 'Powered by Exalate v. 5.4.9 (Core v. 5.4.9)'.

After a brief pause, a successful synchronization looks as shown in the screen above. The Work Item you have synchronized creates an Incident in ServiceNow. Status and any other key updates will be now synced bidirectionally between them.

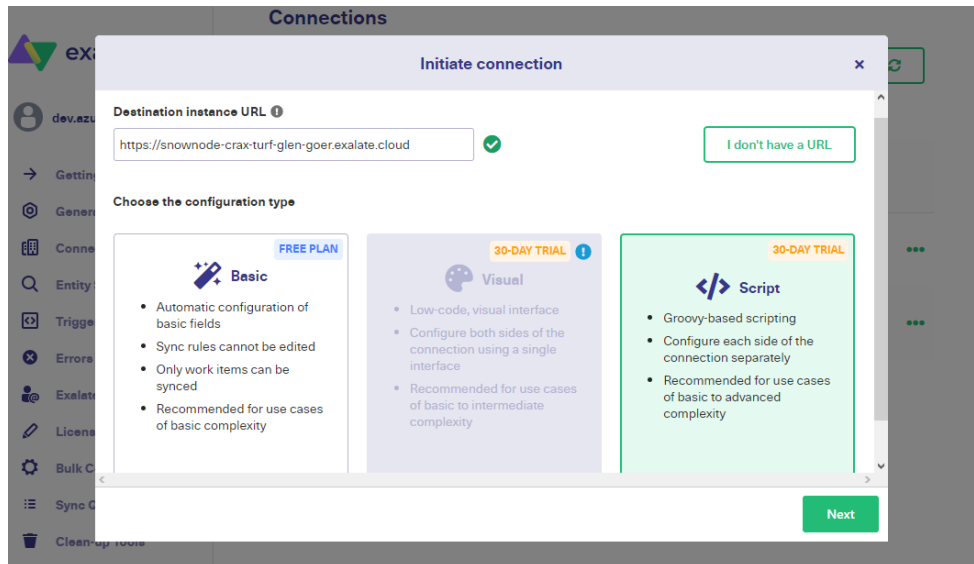
You can also follow the Work Item or Incident by clicking the remote link generated on the Azure DevOps and the ServiceNow sides.

Using the Basic mode, you can synchronize Work Items or Incidents as shown above. To sync existing Azure DevOps and ServiceNow entities you can create [triggers](#) or use the '[Bulk Connect](#)' option. We will cover triggers in detail in a short while.

For now, let us move on to

## Continue with the Script Mode

To continue with this mode, select 'Script' on the screen to choose the configuration type and click 'Next'.





You can give each side of the connection a name, and the names will be used to generate a name for the connection. You can also give your connection an optional description.

It is worth thinking about these steps, as when you have several connections it is useful to be able to tell which is which. It also helps other people who may need to work with these connections later.

**Initiate connection** ✕

**Connection information**

<p><b>Local instance short name*</b></p> <div style="border: 1px solid #ccc; padding: 5px; display: flex; align-items: center;"> Azure DevOps</div>	<p><b>Remote instance short name*</b></p> <div style="border: 1px solid #ccc; padding: 5px; display: flex; align-items: center;"> ServiceNow</div>
---	--

**Connection name\***

Azure DevOps \_to\_ ServiceNow

**Description**

This is a connection between Azure DevOps and ServiceNow

---

[< Previous](#)Next

After entering the details, click the green 'Next' button.

You need to pick the project that is going to be synchronized via the connection. Use the drop-down box to choose from the available projects and then click the green 'Initiate' button.



**Select a project for the incoming sync**

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

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

 A dropdown menu with a blue border, containing the text "Software Development" and a small downward-pointing chevron icon.

[← Previous](#)

Initiate

Exalate will generate an invitation code. Copy this and paste it somewhere safe. You can choose to close the window or click 'Go to remote' to go to the ServiceNow instance. For the former approach, you'll be taken back to the connections screen where you'll see your connection marked as 'Pending'.

Head over to ServiceNow if you are not already redirected.

**Connections**

Connection defines synchronization behavior, including communication details, sync rules, and scope.

[Initiate connection](#) [Accept invitation](#) [Refresh](#)

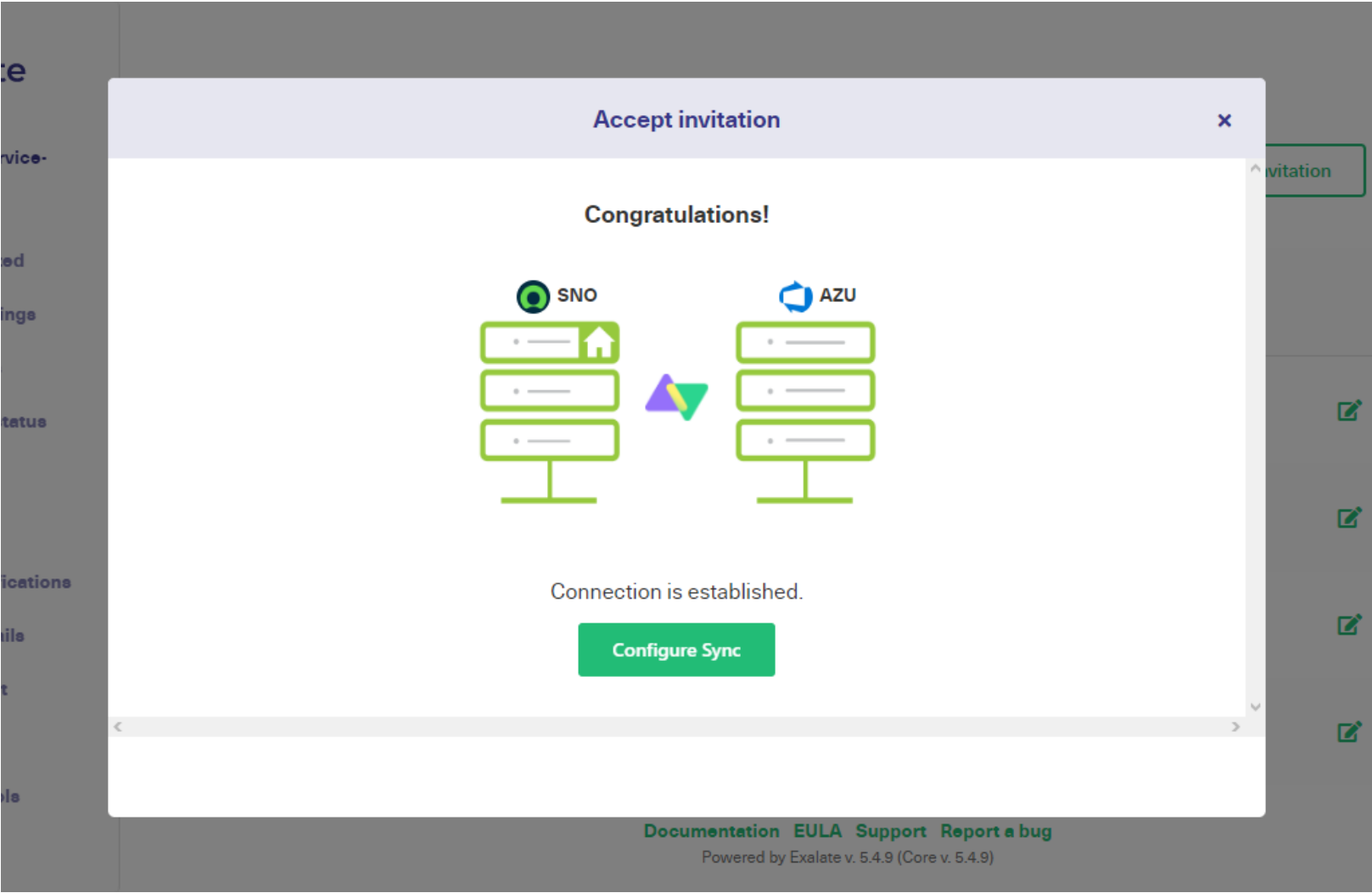
Connection	Entities under sync	Last sync	Status
SNOWRackspace_to_HLAGJira EPSO connection, please do not delete!	2	incident INC00... 2 months ago	● Deactivated <a href="#">Edit</a> <a href="#">More</a>
snow_to_jc_sonal	3	incident INC00... 1 month ago	● Deactivated <a href="#">Edit</a> <a href="#">More</a>
SNOW1_to_SNOW2	6	incident INC00... 3 months ago	● Deactivated <a href="#">Edit</a> <a href="#">More</a>
SNOW_to_VioletSupport	2	sn_customerservice_c ase CS000... 2 weeks ago	● Deactivated <a href="#">Edit</a> <a href="#">More</a>

[Documentation](#) [EULA](#) [Support](#) [Report a bug](#)

Powered by Exalate v. 5.4.9 (Core v. 5.4.9)



On the next screen, you'll see a text field. Paste in the invitation code generated in Azure DevOps. Then click the green 'Next' button.



The connection between Azure DevOps and ServiceNow is successful. You can start configuring it right away or you can choose to close this window and configure it as shown in step 4.

**Connections**

Connection defines synchronization behavior, including communication details, sync rules, and scope.

[Initiate connection](#) [Accept invitation](#)

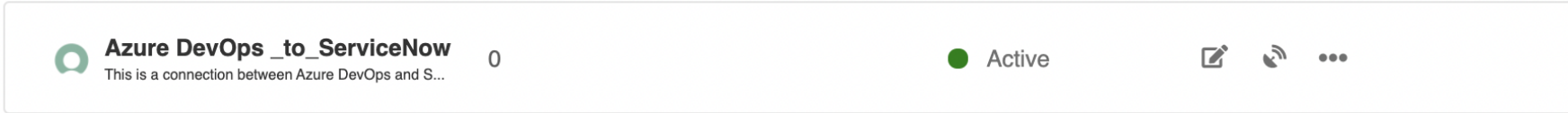
Connection	Entities under sync	Last sync	Status
<b>ServiceNow_to_Jira</b> This is a connection between ServiceNow and Jira	0		● Active
<b>Azure DevOps_to_ServiceNow</b> This is a connection between Azure DevOps and S...	0		● Active

< 1 >

Back in the connection list, we can see our connection is there, with the details we selected earlier. Now, we will edit the connection and control what it does, and when it does it.

### Step 4: Configure Your Connection to Determine What Information Gets Shared

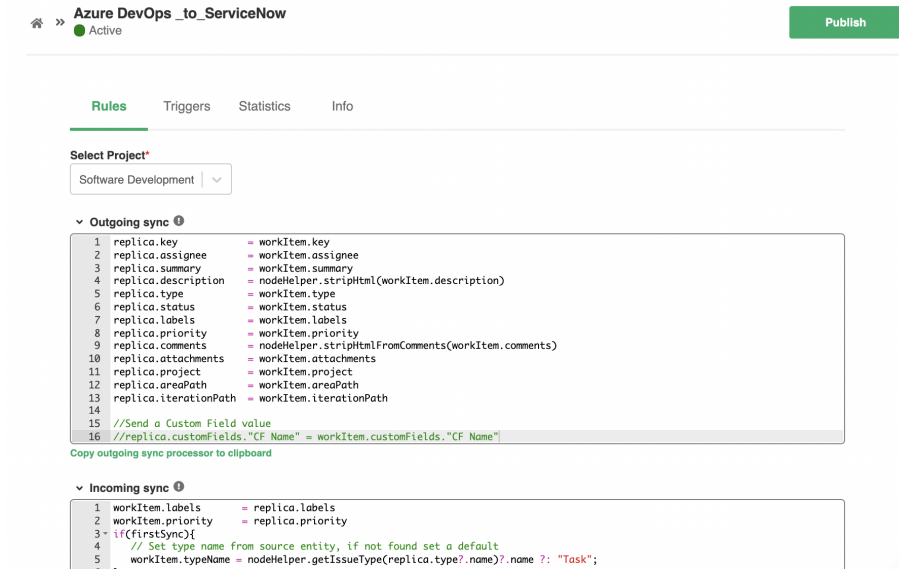
In the Azure DevOps connections list, hover the mouse over the connection we just created.



Click the edit icon that appears in front of its name. From here we can edit the synchronization triggers, which we'll do in the next step. We can also look at the sync rules, and view statistics and information about our connection.

The statistics tell us how many issues are under sync, which is a useful way to check that the changes we make are working correctly.

For now, click the 'Rules' tab.



We can see a list of outgoing sync rules and incoming sync rules. Since we are in Azure DevOps, the outgoing sync rules refer to items sent from Azure DevOps to ServiceNow. The incoming rules refer to items sent from ServiceNow to AzureDevOps.

If we view the same screen in ServiceNow, we will see a similar screen, but these relationships will be reversed.

Let's look more closely at the incoming rules in Azure DevOps.

▼ Incoming sync ⓘ

```
1 workItem.labels      = replica.labels
2 workItem.priority    = replica.priority
3- if(firstSync){
4   // Set type name from source entity, if not found set a default
5   workItem.typeName = nodeHelper.getIssueType(replica.type?.name)?.name ?: "Task";
6 }
7
8 workItem.summary     = replica.summary
9 workItem.description = replica.description
10 workItem.attachments = attachmentHelper.mergeAttachments(workItem, replica)
11 workItem.comments   = commentHelper.mergeComments(workItem, replica)
12- /*
13 Area Path Sync
14 This also works for iterationPath field
15
16 Set Area Path Manually
17 workItem.areaPath = "Name of the project\\name of the area"
18
19 Set Area Path based on remote side drop-down list
20 Change "area-path-select-list" to actual custom field name
21 workItem.areaPath = replica.customFields."area-path-select-list"?.value?.value
22
23 Set Area Path based on remote side text field
24 Change "area-path" to actual custom field name
25 workItem.areaPath = replica.customFields."area-path".value
26 */
27
28- /*
29 Status Synchronization
30
31 Sync status according to the mapping [remote workItem status: local workItem status]
32 If statuses are the same on both sides don't include them in the mapping
33 def statusMapping = ["Open":"New", "To Do":"Open"]
34 def remoteStatusName = replica.status.name
35 workItem.setStatus(statusMapping[remoteStatusName] ?: remoteStatusName)
36 */
37
```

[Copy incoming processor to clipboard](#)



There are several lines, such as `workItem.summary = replica.summary` that control how data is mapped from one platform to the other. 'Replica' here refers to the payload exchanged between our Exalate nodes. It contains the details of the item from ServiceNow that is being synced to the 'Work Item' in Azure DevOps.

In this case, there are several fields that are mapped directly onto each other. The summary, description, priority, and label fields will all be copied directly when the fields are synced.

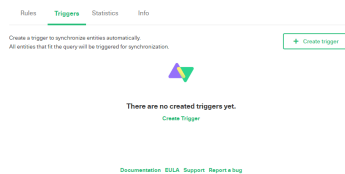
You can modify these as required. If you don't want a field mapped, delete the relevant line. If you want fields to map differently, change them. You could map them to a different field, for example by writing `workItem.labels = replica.priority`. You could also add specific text of your own, for example, `workItem.description = 'synced from ServiceNow'`.

The attachments and comments fields use functions that automatically handle these items. These are a little more complicated, but you can investigate them and change them if you need too. Again, you can easily delete them in case you don't want those fields mapped.

The commented areas, that start with `/*` and end with `*/`, give you information about what you can do with the edit rules. To learn more, read our [script helpers](#) guide.

## Step 5: Set Up Automated Synchronization Triggers

Now we've looked at what we're sending, we'll look at how to configure automated synchronization triggers. These set the conditions for sending items from each platform to the other.



If you aren't already there from the previous step, log in to Azure DevOps, and click the 'edit' icon on your connection. Make sure the 'Triggers' section is selected. Click it if not. You can also click 'Triggers' in the left-hand menu.

Now, click the 'Create trigger' button. The 'Add trigger' dialogue will appear. You can choose the entity type to work with from the drop-down box. In this case, we only have one option, 'Work item'. In other platforms or projects, you may have other options available to select from.

In the 'If' text field, we can enter a search query that will be used to select matching items. Our query could be used to exchange all items of a particular type or from a particular user. We could send urgent queries or queries that have comments. We can also set up multiple triggers to send items that match different conditions.

The search query uses Work Item Query Language or WIQL for short. Check [here for a guide](#) to it. Look for the "Filter Conditions (WHERE)" section of the guide, which is most relevant to what we are doing here.

Add trigger ✕

Trigger will apply to selected entity type ⓘ

Work Item ▾

Specify a search query using Azure DevOps advanced search syntax to synchronize work items automatically. All entities that fit the query will be triggered for synchronization. [Find more details.](#)

**If** ⓘ

[Work Item Type] = 'Task'

**Notes**

Synchronize tasks

**Active?**

Add

For this example, let's create a query that syncs items of the 'task' type. To do that, we type the code `[Work Item Type] = 'task'`, into the 'if' field. This trigger will apply to any item where 'task' is set as the type. You can substitute 'task' for any other type, or 'Type' for other fields.




You can also add a description. It is best to add as much information about what you are doing and why you are doing it. That makes it easier to work with if you come back to it later, and also enables others to understand what is going on.

There's also a switch to activate the trigger. You need to click this, or the trigger won't do anything. Later, this can be used to quickly switch existing triggers on or off.

**Triggers** Rules Statistics Info

Create a trigger to synchronize issues automatically.  
All issues that fit the query will be triggered for synchronization.

[+ Create trigger](#)

When	If	Status	Action
Issue Events: create/update	[Work Item Type] = 'task'	<input checked="" type="checkbox"/>	  

< 1 >

Click the green 'Add' button and you'll see your trigger listed, and active. Items that meet these criteria will now be synchronized automatically.

## Common Pitfalls to Avoid after Setting up the Azure DevOps ServiceNow Integration

There are a few things to be aware of when setting up an Azure DevOps ServiceNow integration that will help you ensure things go as smoothly as possible.

### Role Clarification

As discussed above, Azure DevOps is intended for use by developers. ServiceNow is for support teams. While it is highly beneficial to exchange information, there is plenty that you don't need to share. Support teams won't want to know the technical details, and developers don't need to know the full history of interactions with customers.

When choosing what fields to sync, be careful to make sure teams get the information they want. You might want to have a dedicated field where the support team summarizes each issue, rather than pass on all the customer comments.

Similarly, developer comments may not be useful to the customer support team, but they are likely to want to know when an issue has been resolved and what the solution is. Try to set your sync rules up to reflect each team's needs.

### Too Many Messages

Teams may get notifications when new tickets are created or comments are added. When synchronizing platforms, make sure notifications are tuned to stop people from being bombarded with messages. That increases engagement and makes synchronization more useful to team members.

## Conclusion

As we've seen, there are many benefits to an Azure DevOps ServiceNow integration such as letting teams work more smoothly and efficiently. There are many things to consider when connecting teams, and taking careful account of the issues raised here will help your organization get the best value possible from their integration.

With care, you can ensure you craft a process that delivers exactly what you need, making the best use of the information your teams gather while minimizing the need for maintenance and avoiding problems with duplication and wasted effort.

We've seen how [Exalate](#) can bridge the gap between different groups and help them share information while retaining their autonomy and giving them enough flexibility to evolve. It does so reliably, allowing everyone to focus on what they do best.

### **Recommended Reads:**

- [ServiceNow to ServiceNow Integration: Set up a Two-Way Sync](#)
- [Jira ServiceNow Integration: How to Set up an Integration in 6 Steps](#)
- [Jira Azure DevOps Integration: The Complete Step-by-Step Guide](#)
- [How to Set up a Salesforce ServiceNow Integration](#)
- [How to Set up an Azure DevOps Salesforce Integration](#)
- [Zendesk ServiceNow Integration: The Complete Guide](#)
- [How to Set up a Zendesk Azure DevOps Integration](#)
- [ServiceNow Integrations: Integrate ServiceNow and Other Systems Bidirectionally](#)