



How to Set up an Azure DevOps Salesforce Integration: The Comprehensive 2023 Guide

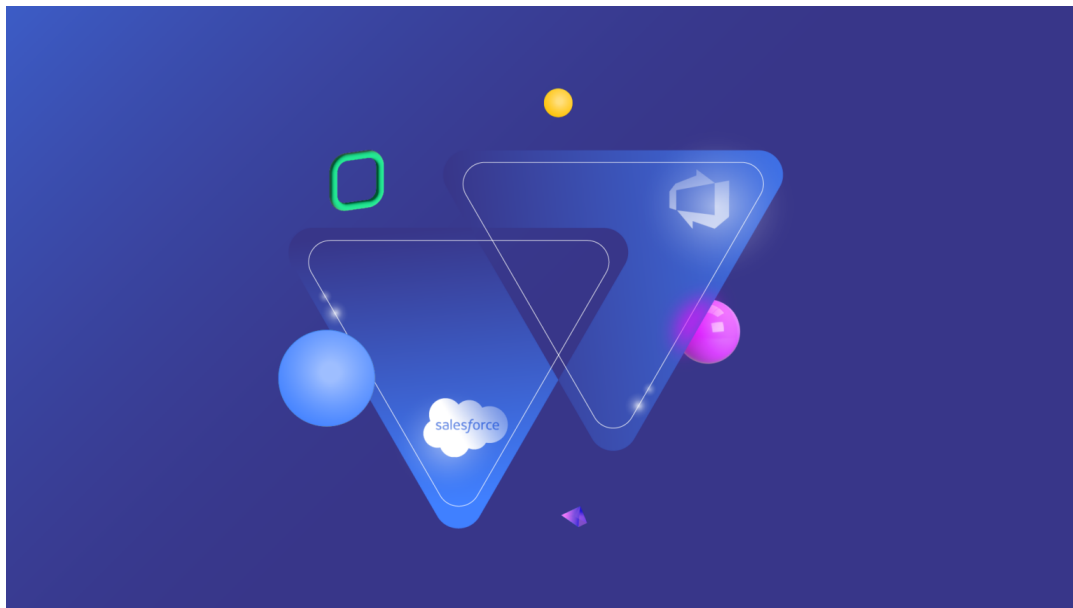


Table of contents

Why Integrate Azure DevOps and Salesforce

How to Choose the Right Solution for an Azure DevOps Salesforce Integration

How to Set up an Azure DevOps Salesforce Integration in 6 Steps

- Step 1: Install Exalate on Azure DevOps
- Step 2: Install Exalate on Salesforce
- Step 3: Connect Azure DevOps and Salesforce
- Step 4: Customize the Connection to Decide What Information Gets Shared
- Step 5: Set Conditions for Automatic Synchronization: Triggers
- Step 6: Synchronize Information between Azure DevOps and Salesforce

Common Use Case

- Development and Sales Teams

Conclusion



Information continuously evolves and grows in organizations. This untapped information, if shared between teams using different applications like Azure DevOps and Salesforce, can increase business revenues and leave you with more satisfied customers. For this purpose, an Azure DevOps Salesforce integration seems like the best choice.

Such integration would mean the information exchange between Azure DevOps and Salesforce is automatic, bi-directional, and in real-time, so data is consistently accessible and up-to-date.

This guide will introduce you to a step-by-step Azure DevOps Salesforce integration. It will also help you understand the benefits of such integration and present the key drivers for choosing the correct integration technology.

Note: *In this guide, we have used an integration solution called [Exalate](#) for implementing an Azure DevOps Salesforce integration. We will learn more about it on the way!*

Here is what we're going to cover in this article:

- [Why Integrate Azure DevOps and Salesforce](#)
- [How to Choose the Right Solution for an Azure DevOps Salesforce Integration](#)
- [How to Set up an Azure DevOps Salesforce Integration in 6 Steps](#)
 - [Continue with the Basic Mode](#)
 - [Continue with the Script Mode](#)
- [Common Use Case](#)

Why Integrate Azure DevOps and Salesforce

Azure DevOps has been a game-changer in bringing the development and operation teams together. It facilitates an end-to-end DevOps toolchain for developing and deploying software programs. It aims to bridge the gap between the once-siloed development, IT operations, engineering, and security teams, to help them collaborate and work together harmoniously.



It has a gamut of applications supported through its [marketplace](#).

Salesforce is a popular Customer relationship management (CRM) tool used by sales, marketing, service, and IT teams to strengthen customer bonds and relationships. It aims to provide a complete customer overview through a single Salesforce interface. It is a robust, customizable, and scalable cloud-based platform that can fulfill most of your business needs.



It too has a variety of best-in-class apps provided through [AppExchange](#).

A common trait you can observe here is that both these applications are built to bridge the gap between teams and bring them closer, be it development and operations or sales and marketing. Therefore, they can share information and build on each other's expertise to

achieve common business improvement goals.

But this collaboration and communication between teams if not handled correctly can lead to a waste of time and effort of valuable human resources and will eventually reduce productivity.

This is because information exchange between teams is generally carried out manually via phone calls and emails or meeting notes or by switching between applications. Then it is further absorbed and updated in respective platforms manually.

This causes costly manual data-entry errors in addition to misplaced, unused, redundant, and scattered information across different platforms.

So how do we let these teams exchange data automatically without having to leave their familiar environment?

An Azure DevOps Salesforce integration would make perfect sense here. It will allow for real-time, automatic two-way synchronization of information between Azure DevOps and Salesforce, leaving teams in a more streamlined and collaborative environment.

Choosing the right integration tool here is the next logical step. So let's understand how we can do that.

How to Choose the Right Solution for an Azure DevOps Salesforce Integration

The possibilities and checklists for a suitable integration tool can be overwhelming. But to bring down the list to the bare minimum, here's what you must consider.

Note: As mentioned above, we have chosen [Exalate](#) for implementing an Azure DevOps Salesforce integration and we will look at what it has to offer.

Security

Perhaps, this should be your top criterion for an integration tool, since it involves disparate applications across companies sending and receiving business information. It's important then, to secure this critical business data from being accidentally misused or shared with unauthorized parties. Using secure transport protocols like HTTPS, encrypted file transfers, or controlling access to the data to authorized users only are important security mechanisms.

You can read more about how Exalate implements security measures by reading its [security and architecture whitepaper](#).

Decentralized Integration

It's important for an integration tool to support decentralized integration. This means that both sides of the integration have complete control over what data is sent and how incoming data is received.

They can share or receive data they want independently without having to configure or inform the other side. This guarantees that both sides maintain autonomy.

Exalate achieves this with the help of its scripting engine that supports outgoing and incoming sync rules on both integration sides. These rules help control the information shared, thus ensuring decentralized integration.

Reliability

The way your integration tool handles downtime and system failures is also important. Sync updates or changes must be queued in the proper manner and applied in the same order as their initiation. All this must be done automatically without any manual intervention.

Exalate achieves this using an advanced transactional sync engine which breaks down all the synchronization requests in single atomic steps that are applied in the same order and retried once the system resumes from a downtime even when a system is being upgraded to a new version or a firewall is being reconfigured.

Flexibility

Your integration and synchronization requirements are bound to change with time. It's natural for you to expect your integration tool to adapt to these changing requirements with minimal configuration.

Exalate, for example, supports basic synchronization scenarios straight out of the box, so it works on a low-code non-configuration mode for simple use cases that can be set up in very little time and is also easy for business users to understand. And you can also build custom synchronizations using its intuitive scripting engine for unique or advanced integration cases.

Another aspect of a tool being flexible is that it supports integrations with other popular applications like Jira, GitHub, Zendesk, ServiceNow, and HP QC/ALM as well. Since Exalate already supports these integrations, it will be easy for you to integrate with another of your partner, vendor, customer, or supplier who uses one of these applications.

Having chosen the right tool for an Azure DevOps Salesforce integration, it's now time we learn how to implement the integration.

How to Set up an Azure DevOps Salesforce Integration in 6 Steps

To begin with, Exalate needs to be installed on both Azure DevOps and Salesforce instances. This is an important part of Exalate's distributed architecture feature. Installing it on both sides will mean that the shared information can be controlled from either end independently.

You then establish a connection between them, it's like an initial handshake to know and acknowledge what instance you are integrating with.

And finally, you can configure the connection to control what is sent and received and then set automatic triggers to start syncing data.

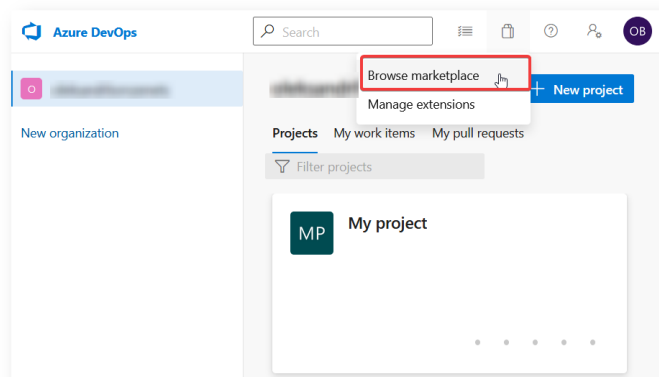
So let's dive right into it!

Step 1: Install Exalate on Azure DevOps

You can install Exalate on Azure DevOps through its marketplace or on a docker. (Check out this [guide](#) for a detailed procedure on installing Exalate on Azure DevOps).

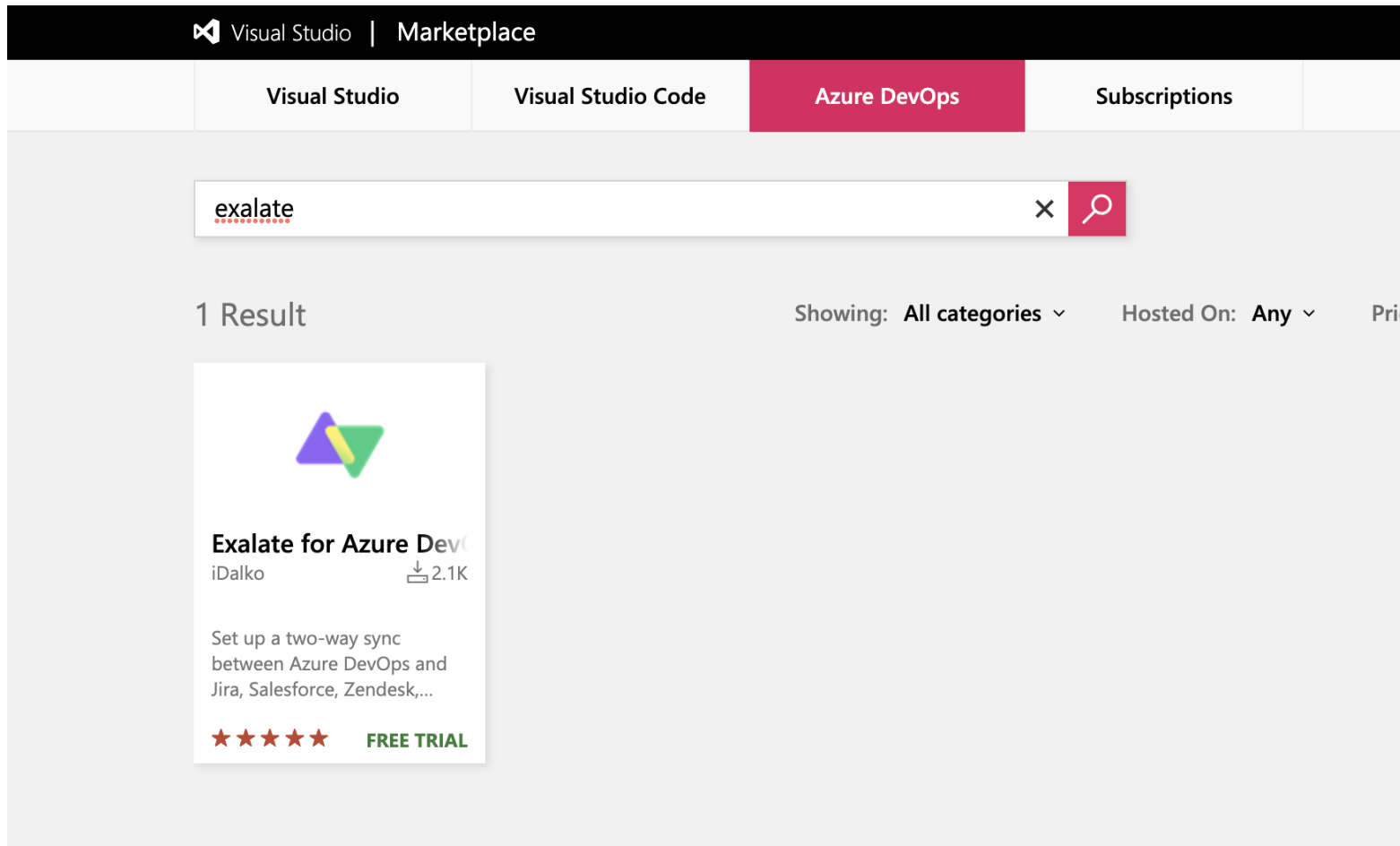
We are going to install it from the marketplace in this guide.

You can find [Exalate on the Azure DevOps marketplace](#). Or you can first log in to your instance and navigate to “Browse marketplace” from within it.



If you can't find it, then navigate to Organization Settings - Extensions and then click "Browse marketplace".

Type "Exalate" in the search box.



On the next screen, click the green “Get” button to start the 30-day free trial.

Exalate for Azure DevOps Integration

iDalko | 2,049 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.

Secure, Reliable, Real-time Synchronization Process

Exalate has been built to ensure that all changes are applied in case of failure or upgrades through the advanced

Categories

Azure Boards

Tags

almmicrofocus azuredevops github hpqc integration jcloud jira jiraserver servicenow zendesk

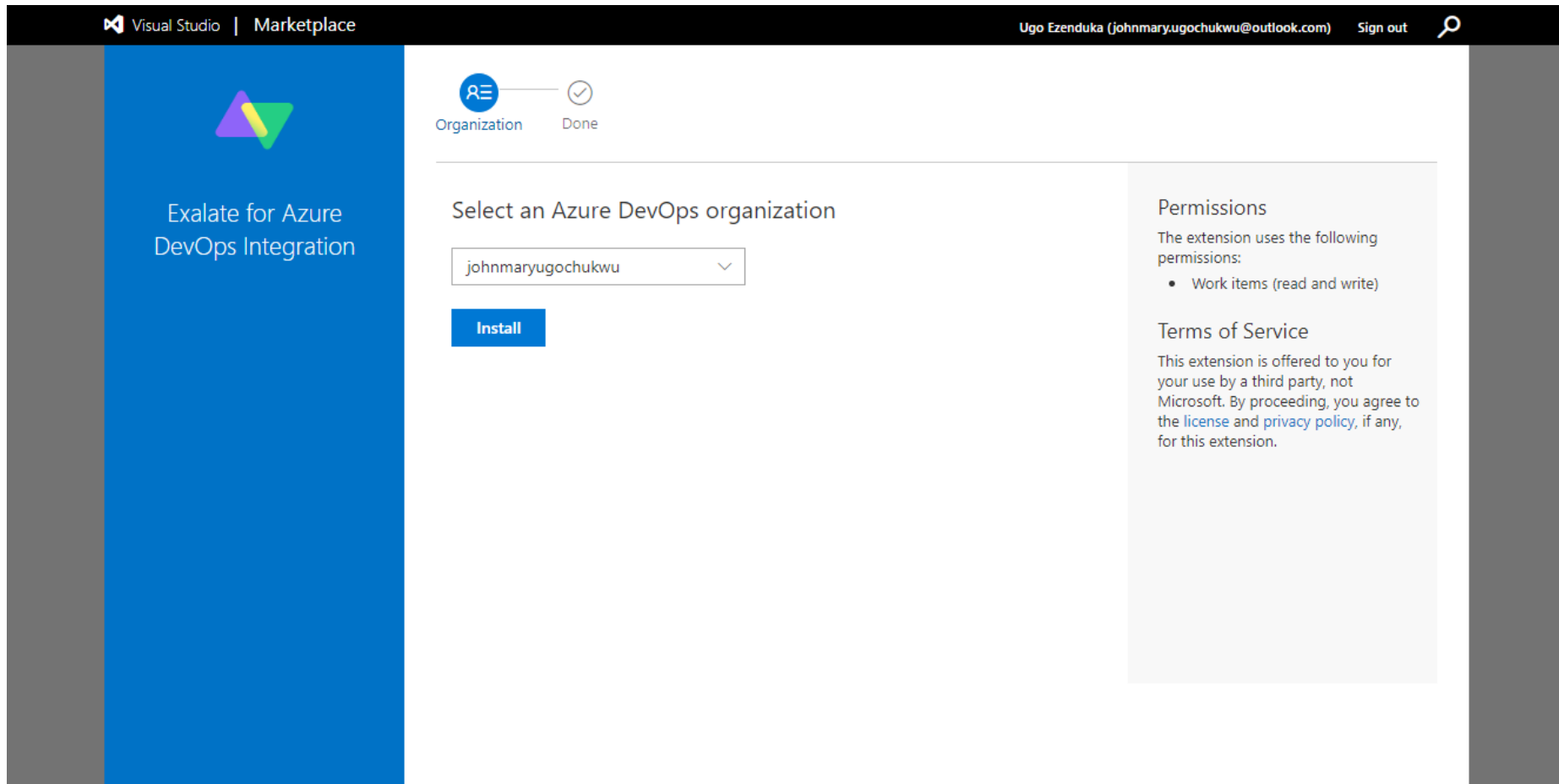
Works with

Azure DevOps Services

Resources

[Technical Support](#)
[Get Started](#)

You will now be taken through the installation wizard. You need to select your organization type from the drop-down list. After that, click “Install”.



The screenshot shows the Visual Studio Marketplace interface for the 'Exalate for Azure DevOps Integration' extension. The left sidebar features the extension's logo and name. The main content area is titled 'Select an Azure DevOps organization' and includes a dropdown menu with 'johnmaryugochukwu' selected and an 'Install' button. On the right, there are sections for 'Permissions' (listing 'Work items (read and write)') and 'Terms of Service' (stating the extension is offered by a third party, not Microsoft).

Follow the wizard next as it's pretty straightforward.

Once done, you can access Exalate by going to the “Organization Settings” and clicking “Extensions”.

Then proceed to [verify Exalate on your Azure DevOps instance](#).

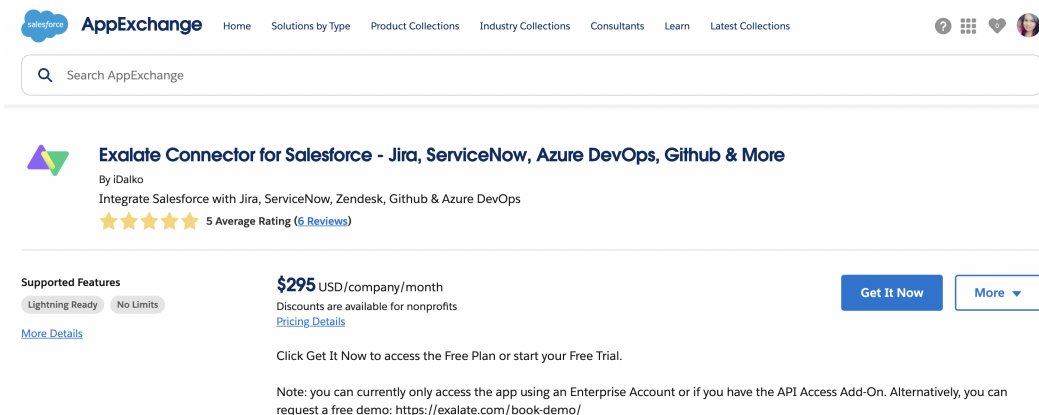
Once you finish this, install it on the Salesforce side. You can skip this if you have already installed it and move to the next step.

Step 2: Install Exalate on Salesforce

To start installing Exalate from Salesforce AppExchange, head over to its [website](#).

Note: Exalate accesses Salesforce through APIs. Salesforce has its own [guidelines](#) for API Access add-ons. For example, API access is provided by default in Enterprise accounts, while it is not the case with other accounts like Professional. Visit this documentation [page](#) to learn about the different Salesforce editions Exalate supports.

On the "Search AppExchange" bar type "Exalate". At this point, you will be asked to log in if you still haven't. Then click on the "Get It Now" green button. Go through the app details if you want before.



The screenshot shows the Salesforce AppExchange interface. At the top, there is a navigation bar with the Salesforce logo, "AppExchange", and links for Home, Solutions by Type, Product Collections, Industry Collections, Consultants, Learn, and Latest Collections. A search bar is prominently displayed with the text "Search AppExchange". Below the search bar, the app "Exalate Connector for Salesforce - Jira, ServiceNow, Azure DevOps, Github & More" is featured. It is developed by iDalko and has a 5-star average rating from 6 reviews. The pricing is listed as \$295 USD/company/month, with a note that discounts are available for nonprofits. There are "Get It Now" and "More" buttons. A note at the bottom of the app card states: "Click Get It Now to access the Free Plan or start your Free Trial. Note: you can currently only access the app using an Enterprise Account or if you have the API Access Add-On. Alternatively, you can request a free demo: https://exalate.com/book-demo/".

Choose the "Install in Production" environment, but you can choose to install it on Sandbox either.

Where do you want to install this package?

Install in a Production Environment

Install this package in the org where you or your users work, including Developer Edition orgs.

* Connected Salesforce Accounts ⓘ

teja.bhutada@idalko.com



Don't see your account? [More Info](#)

Install in Production

Install in a Sandbox

Test this package in a copy of a production org.

Install in Sandbox

Cancel

A screen will prompt you with the installation details. Go through them once, agree to the terms and conditions and click "Confirm and Install" at the bottom of the page.

Confirm Installation Details

Free	idalko
Duration	Number of Subscribers
Does Not Expire	Site-wide
Username teja.bhutada@idalko.com	

Here are the details we'll share from your profile [Edit Profile](#)

* First Name	Teja	* Company	Exalate
* Last Name	Bhutada	* Country	India
Job Title		State/Province	
* Email	teja.bhutada@idalko.com		
Phone			

* I have read and agree to the [terms and conditions](#).


Salesforce.com Inc. is not the provider of this application but has conducted a limited security review. [Learn More about the AppExchange Security Review](#)

Cancel Confirm and Install

Login to your Salesforce instance now if not already logged in.


You now need to choose the profiles that will have permission to use the Exalate app. You can choose it to be used only by admins, for all users, or for specific profiles only. Don't worry if you can't come to an immediate conclusion. You can choose whichever option you want and then choose to edit the permissions later. Click [here](#) to know how to grant user permissions for Salesforce.

Click "Install" after selecting the option.




Install ExalateBridgeApp


By Exalate



Install for Admins Only



Install for All Users




Install for Specific Profiles...

App Name	Publisher	Version Name	Version Number
ExalateBridgeApp	Exalate	1.1.0	1.1


Additional Details [View Components](#)

Next, you need to approve access to third-party websites. So give access and click "Continue". After a successful installation message, you will be redirected to your Salesforce instance by clicking "Done".



Install ExalateBridgeApp

By Exalate

**Installation Complete!**

Done

App Name	Publisher	Version Name	Version Number
ExalateBridgeApp	Exalate	1.1.0	1.1

After this, you need to "Request Node" to enable Exalate to be used on Salesforce. So start by searching for "Exalate" in the "Apps" section of your Salesforce instance and request the node from there.

Click "Allow" to give the necessary permissions to the app.

Then fill in a basic form so that your Exalate for Salesforce instance is verified and the evaluation license activated. On the form, click on "Agree and Submit" and accept the EULA.

An email will arrive in your inbox. Click on "Verify Exalate instance" in the email.

Upon successful verification, you can proceed to the next step.

Anytime you are logged out of your Salesforce instance, follow these [steps](#) to log in again.

Note: To know more about the detailed installation procedure click [here](#).

Step 3: Connect Azure DevOps and Salesforce

On your Salesforce instance, you will be on the welcome screen. Click on the "Connections" tab on the left-hand menu.

Go to the "Connections" tab. Connections in Exalate define your scope of synchronization and specify how your synchronization must behave. A unique connection for each instance is created.

On the screen, you can see existing connections between Salesforce - ServiceNow, Salesforce - Zendesk, and Salesforce - GitHub. However, this screen will be empty if you haven't worked with Exalate before and this is your first time creating connections.

Either way, click on "Initiate connection".

With Exalate, one side initiates the connection and the other side accepts it. Exalate has a uniform interface so you will come across similar screens on either end.

Here, we start from the Salesforce side. But you can start from Azure DevOps as well.

The screenshot shows the Exalate Console interface. At the top, there is a search bar and navigation icons. The main area is titled "Exalate Console" and contains a sidebar with navigation links: Getting Started, General Settings, Connections, Entity Sync status, Triggers, Errors, Exalate Notifications, License Details, and Bulk Connect. The main content area displays a message: "Connection defines synchronization behavior, including communication details, sync rules, and scope." Below this message are three buttons: "Initiate connection", "Accept invitation", and a refresh icon. A table lists existing connections with columns for Connection, Entities under sync, Last sync, and Status.

Connection	Entities under sync	Last sync	Status
SF_CHATTER_THREADS Do not delete please!	3	Case 5007Q... 2 months ago	Deactivated
Suman 2	0		Deactivated
jcloud_to_SF_commnet_with_chatter	3	Case 5007Q... 1 month ago	Active
POCMathieu	6	Contact 0037Q... 3 months ago	Deactivated

You will then be asked to enter the destination instance URL. This will be the link to your Azure DevOps instance.

In case you are confused about which link to enter, you can navigate to the left-hand "General Settings" Exalate menu and copy the URL from there.


After a brief pause, new fields will appear. These will prompt you to choose the configuration type.

Initiate connection ✕


Destination instance URL ⓘ

✓ I don't have a URL

Choose the configuration type

 **Basic**

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

 **Script**

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

Next

Exalate comes in two configuration modes: **Basic** and **Script**.

The Basic mode allows you to work with default mappings and configurations that can't be modified. It comes with the [Free Plan](#) that allows you to experience Exalate firsthand and offers up to 1000 free syncs per month. It is suitable for basic synchronization use cases.

The Script mode, on the other hand, offers the full functionality of Exalate with advanced configurations and scripting capabilities. With this mode, you can implement unique, complex, or advanced synchronization use cases.

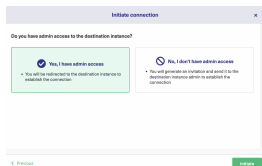
You can also choose to upgrade your existing Basic connection in [Azure DevOps](#) or [Salesforce](#) to a Scripted one if you want to use Exalate with all its features and configuration modes.

Both these modes have slightly different screens, so we will take a look at both of them one by one.

Continue with the Basic Mode

If you select “Basic” on the screen above and click “Next”, you will be asked to verify admin access to the destination instance, Azure DevOps in our case.

Click the “Yes, I have admin access” button if you have access. Then click “Initiate”. If you don't have access, click “No, I don't have admin access”. Exalate then generates a unique invitation code for you. You need to copy and paste this code manually on the Azure DevOps side.



After a quick verification, you will be redirected to the Azure DevOps side.

Select a project on the Azure DevOps side from a drop-down list. This would be the project whose work items you want to sync or the one in which you will receive incoming Salesforce entities.

Click “Next” after selecting the one you want.

Accept invitation ×

Select a project for the incoming sync

After accepting an invitation for a Basic mode connection, Exalate will be able to sync basic work item fields. The sync rules cannot be updated. Exalate will sync the following work item data: summary, description, comments, and attachments.

Work item types are mirrored. This means that tasks will be synced as tasks, bugs as bugs, and improvements as improvements. If a work item type is not included in the basic set, it will be synced as a task.

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

DOPS | ▾

Confirm

A message stating that your connection has been established will be displayed.

You now need to enter the work item key you want to synchronize on the Salesforce side.

Click “Exalate” once you have entered it.

Exalate ✕

Connection established successfully ✓

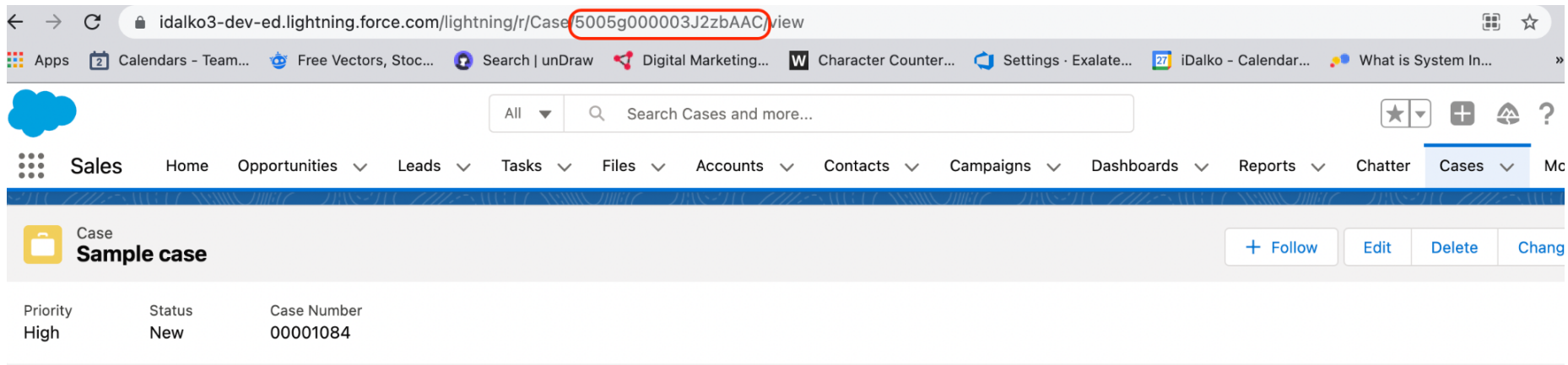
Sync your first work item to see how it works.

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

Exalate

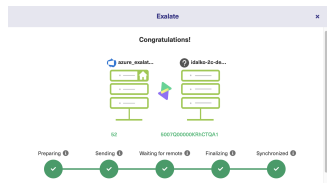
If you now navigate to the Salesforce instance, you will be shown the same screen asking you to enter the Case number you want to synchronize on the Azure DevOps side.

Note: The Case number must be copied from the URL of the Case you select to synchronize. This is shown in the image below.



Sit back and let Exalate do its work. Have a look at the status messages to know what's happening.

After a while, see the successful synchronization status as shown below.



In the Basic mode, you can sync

- individual work items or Salesforce entities as shown above.
- by creating [triggers](#) for automatic synchronizations.
- work items using the [Connect](#) operation on the Azure DevOps side.
- the existing work items and Salesforce entities in bulk using the [“Bulk Connect”](#) operation.

We will see how to create triggers a bit further down.

Continue with the Script Mode

Moving on with the script mode, choose “Script” on the screen below and click “Next”.

Note: We are initiating the connection from the Azure DevOps side now so you know how it works on both ends. But the same screens will appear if you start from the Salesforce side.

Initiate connection [x]

Destination instance URL ⓘ
https://salesforcenode-zsnr-qgor-kyvn-fhmv-exalate.cloud ✓ I don't have a URL

Choose the configuration type

Basic <ul style="list-style-type: none">• Automatic configuration of basic fields• Sync rules cannot be edited• Only work items can be synced• Recommended for use cases of basic complexity	Visual ⓘ <ul style="list-style-type: none">• Low-code, visual interface• Configure both sides of the connection using a single interface• Recommended for use cases of basic to intermediate complexity	</> Script <ul style="list-style-type: none">• Groovy-based scripting• Configure each side of the connection separately• Recommended for use cases of basic to advanced complexity
--	--	---

Next

Enter the local instance short name, Azure DevOps since that's our local instance, and also the remote instance short name, Salesforce in our case.

A connection name joining the two names and appended with a "to" will be generated. You can choose to change it if you want.

Spend some time describing the connection. This will come in handy when you have a lot of them and don't remember why you created a particular one in the first place.

Initiate connection ✕

Connection information

Local instance short name*

Remote instance short name*

Connection name*

Description

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

Click “Next” when you are ready to move ahead.

Then select the project on the Azure DevOps side as you did for the Basic mode.

Initiate connection ✕

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.*

DOPS ▾

[← Previous](#) [Initiate](#)

A unique invitation code will be generated. This code is one of Exalate's security mechanisms, such that each connection established ensures that data is sent and received from the correct source and destination.

Click on "Copy invitation code" to copy it. Keep it somewhere safe.

Click on "Done" once you have copied it.

Initiate connection x

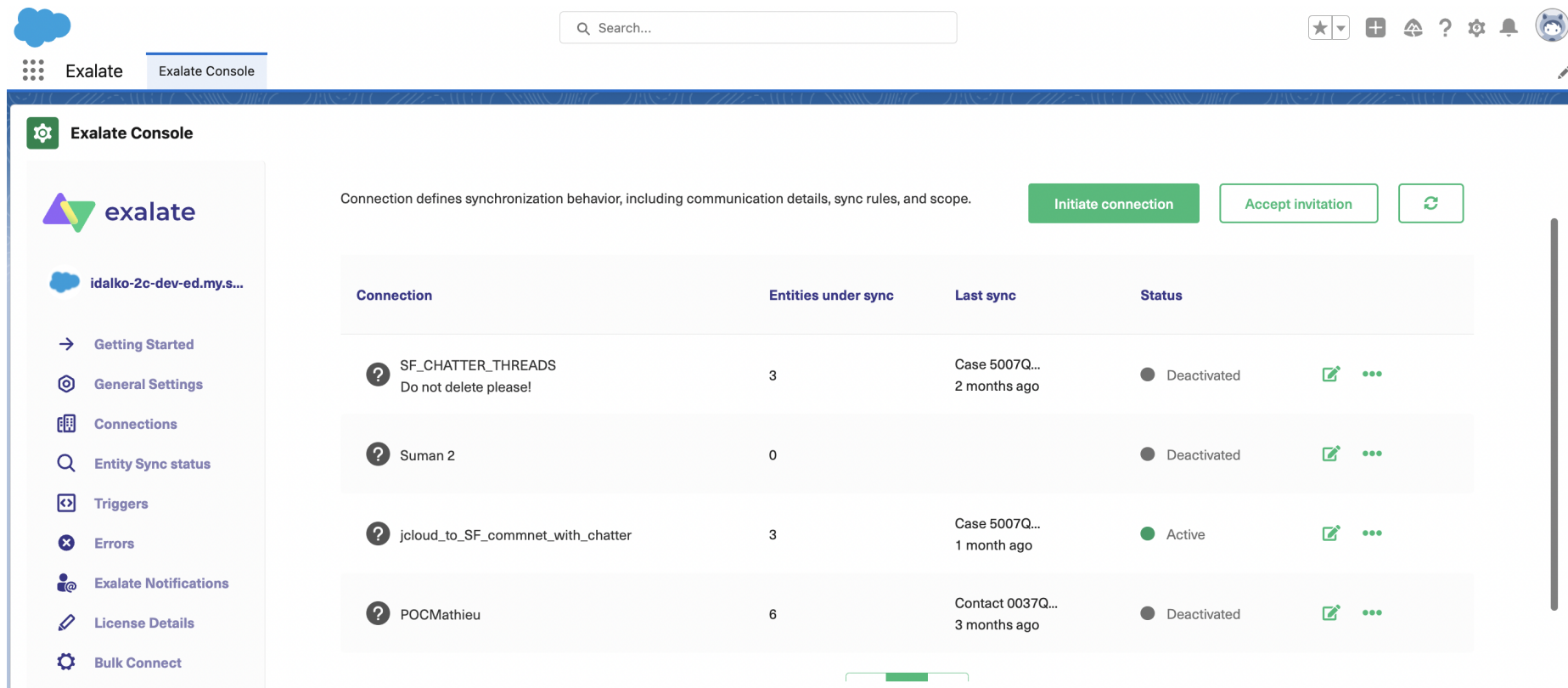
On the "Salesforce" side, you (or their application administrator) need to **Accept the Invitation.**

Use the following invitation code:

Copy invitation code

Done

On the Salesforce “Connections” tab, click on “Accept invitation”.

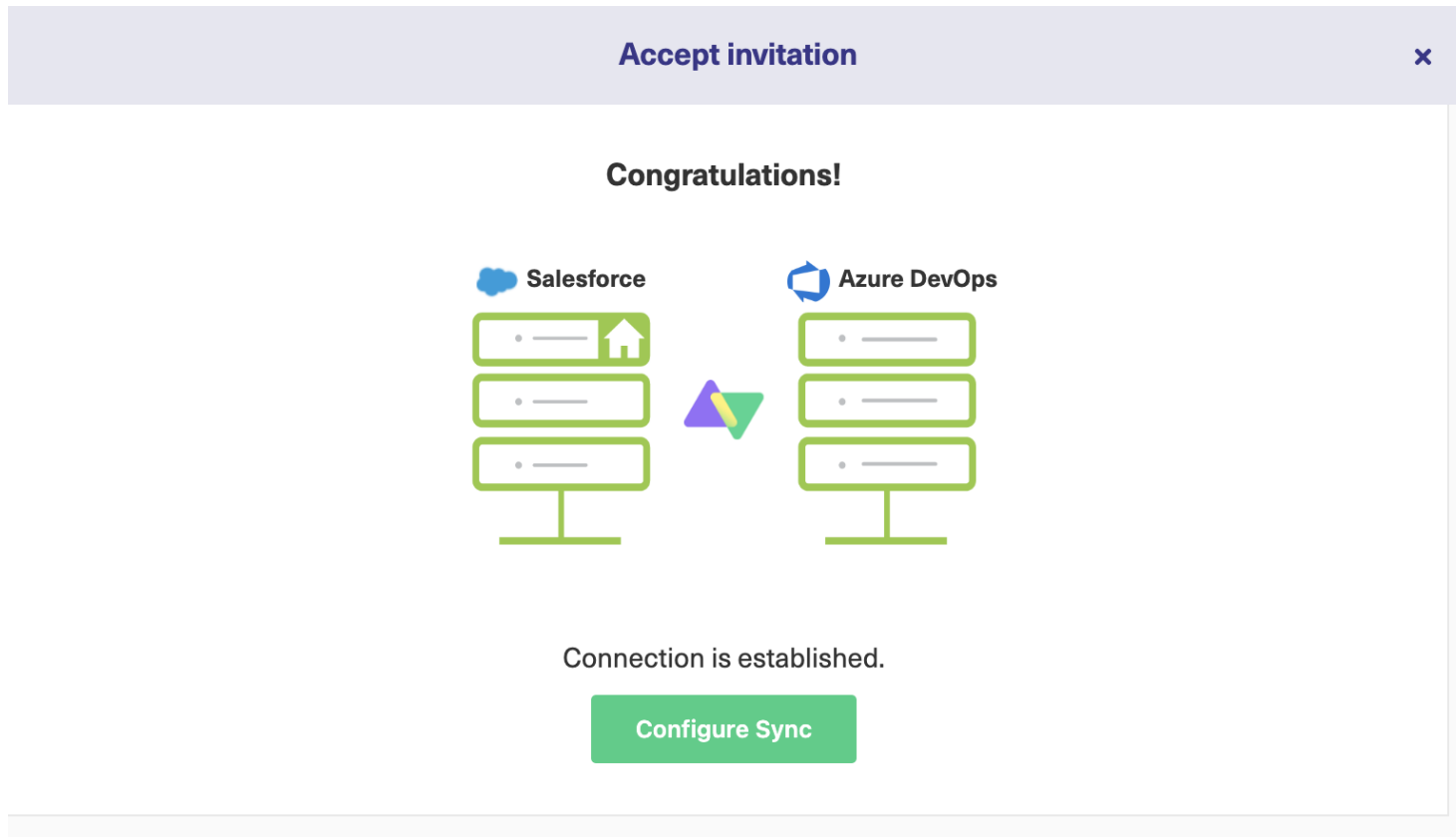


After this, a multi-line text box will appear. Here, you paste the invitation code you had kept safe and click “Next”.



This will complete your connection process in Script mode. You can now proceed to configure the connection.

For this, you click on the “Configure Sync” button or edit the connection if you want to configure it later by closing this window.



Step 4: Customize the Connection to Decide What Information Gets Shared

Customizing or configuring the connection is necessary to control what information you want to send to the other side and how you want to deal with information coming from that side.

It also involves creating automatic synchronization triggers, so you set them once or as needed and experience a two-way synchronization that happens according to the conditions you have set.

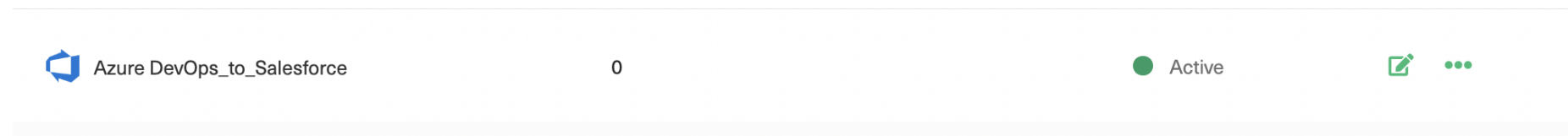
As mentioned before, you can choose to configure using the “Configure Sync” button or choose to edit it.

To edit the connection, navigate to the “Connections” tab and click on the edit icon in front of your connection name as shown below.

Editing can happen at both ends, but the outgoing and incoming information you set or triggers you create will be specific to your platform.

You can also go to the other side of the connection by clicking on the remote antenna icon shown below.

By clicking the 3 dots, you can either activate, deactivate, or delete the connection.



Any way you decide to go ahead, you get similar screens for configuring the connection.

It has 4 tabs: “Rules”, “Triggers”, “Statistics” and “Info”.

We will see rules here and triggers in the next step.

“Statistics” gives you an overview of your synchronizations like the number of issues under sync, the date of your last sync, etc.

The “Info” tab gives you information about your connection like the name, description, and type of connection. You can edit the description here if you want.

» **Azure DevOps_to_Salesforce** ● Active [← Back to Connections](#) [Publish](#)

Rules Triggers Statistics Info

▼ **Outgoing sync** ⓘ

```
1 if(entity.entityType == "Case") {
2   replica.key           = entity.Id
3   replica.summary       = entity.Subject
4   replica.description   = entity.Description
5   replica.comments      = entity.comments
6   replica.attachments   = entity.attachments
7   replica.Status        = entity.Status
8 }
9 if(entity.entityType == "Opportunity") {
10  replica.key            = entity.Id
11  replica.summary        = entity.Name
12  replica.description    = entity.Description
13 }
```

[Copy outgoing sync processor to clipboard](#)

▼ **Incoming sync** ⓘ

```
1 if(firstSync){
2   entity.entityType = "Case"
3 }
4 if(entity.entityType == "Case"){
5   entity.Subject     = replica.summary
6   entity.Description = replica.description
```

Let's discuss the "Rules" tab now.

The screen above shows the rules on Salesforce and the one below shows them on Azure DevOps.

As seen, each side has a set of incoming and outgoing sync rules written in Groovy scripting language. It is pretty intuitive and easy to follow.

Incoming sync rules specify how you want to map the incoming information.

The Outgoing sync rules specify what information you want to sync with the other side.

You can edit the sync rules by adding scripts to send (Outgoing sync) or receive (Incoming sync) new information. Or by deleting the existing rules to stop sending (Outgoing sync) or receiving (Incoming sync) data, depending on whether you are editing it on the Azure DevOps side or the Salesforce side.

For sending or receiving new information, we add a script and make changes in the outgoing/incoming sync rules.

So in the screen shown above, if we want to synchronize "Opportunity" in addition to "Case", we add the following code:

```
if(entity.entityType == "Opportunity") {  
  
    replica.key      = entity.Id  
  
    replica.summary  = entity.Name  
  
    replica.description = entity.Description  
  
}
```


» **Azure DevOps_to_Salesforce**

● Active

< [Back to Connections](#)

[Publish](#)

Rules

Triggers

Statistics

Info

▼ **Outgoing sync** ⓘ

```

1 replica.key = workItem.key
2 replica.assignee = workItem.assignee
3 replica.summary = workItem.summary
4 replica.description = nodeHelper.stripHtml(workItem.description)
5 replica.type = workItem.type
6 replica.status = workItem.status
7 replica.labels = workItem.labels
8 replica.priority = workItem.priority
9 replica.comments = nodeHelper.stripHtmlFromComments(workItem.comments)
10 replica.attachments = workItem.attachments
11 replica.project = workItem.project
12 replica.areaPath = workItem.areaPath
13 replica.iterationPath = workItem.iterationPath
14
15 //Send a Custom Field value
16 //replica.customFields."CF Name" = workItem.customFields."CF Name"
    
```

[Copy outgoing sync processor to clipboard](#)

▼ **Incoming sync** ⓘ

```

1 if(firstSync){
2 // Set type name from source entity, if not found set a default
3   workItem.projectKey = "DOPS"
4   workItem.typeName = nodeHelper.getIssueType(replica.type?.name)?.name ?: "Task";
5 }
6
7 workItem.summary = replica.summary
    
```

Here, *entityType* is the Salesforce entity you want to sync. The ID, name, and description of “Opportunity” are synced with key, summary, and description of work items.

Now if you want to stop the data exchange, delete those lines in the outgoing and incoming sync rules respectively.

If you prefer to keep the rules so that you can use them later, you can simply comment them as shown on the screen below.

To comment a single line, add “//” at the start of the line. To add a comment to multiple lines, add a “/*” at the start of the line and a “*/” wherever you want the comment to end. The commented lines will be ignored while synchronizing information.

For instance, we don’t want to sync the summary of work items, we simply comment the line: `//replica.summary = workItem.summary`.

▼ **Outgoing sync** ⓘ

```
1 replica.key = workItem.key
2 replica.assignee = workItem.assignee
3 //replica.summary = workItem.summary
4 replica.description = nodeHelper.stripHtml(workItem.description)
5 replica.type = workItem.type
6 replica.status = workItem.status
7 replica.labels = workItem.labels
8 replica.priority = workItem.priority
9 replica.comments = nodeHelper.stripHtmlFromComments(workItem.comments)
10 replica.attachments = workItem.attachments
11 replica.project = workItem.project
12 replica.areaPath = workItem.areaPath
13 replica.iterationPath = workItem.iterationPath
14
15 //Send a Custom Field value
16 //replica.customFields."CF Name" = workItem.customFields."CF Name"
```

Proceed to create automatic synchronization triggers now.

Step 5: Set Conditions for Automatic Synchronization: Triggers

Sync rules control what information you send and receive. To start the synchronization, you need to create triggers. Triggers are conditions you set, which when met, exchange information based on the outgoing and incoming sync rules you have set.

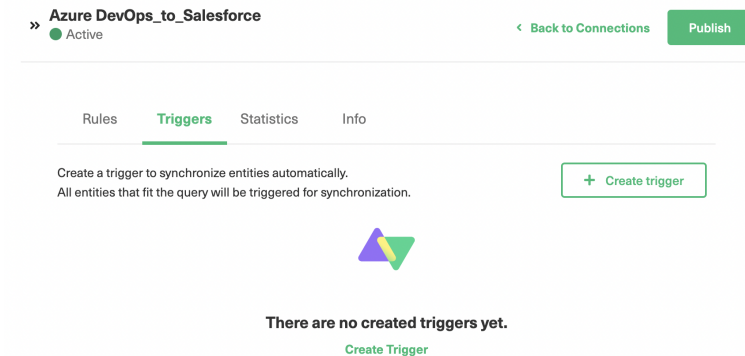
To create triggers, click on the “Triggers” tab shown below.

Alternatively, you can also create triggers by clicking on “Triggers” in the left-hand menu of “Exalate”.

Both approaches send you to similar screens, but the latter one asks you to select the connection you want to create a trigger from since it is from the general tab.

You will see a list of triggers here. It will be empty if you haven’t created any.

Click on the “Create trigger” button.



An “Add trigger” pop-up will be displayed.

The first drop-down will ask you to select the entity type you want to create the trigger for.

On the Azure DevOps side, it is “Work Item”.

On the Salesforce side, there are many entity types you can sync, but the most popular ones are **Accounts, Products, Opportunities, Tasks, and Cases.**

Add trigger ×

the query will be triggered for synchronization. [Find more details.](#)

Trigger will apply to selected entity type* !

Work Item ▾

If* !

[Work Item Type] = 'Task'

Notes

The work items of the type task are automatically synced.

Active?

Add

The “If” field is the condition of the trigger. In the above example, we created a trigger for the work item type “Task” i.e *[Work Item Type] = ‘Task’*.

You can use platform-specific query language in the “If” section.

[WIQL](#) (Work Item Query Language) in the case of Azure DevOps and [SOQL](#) (Salesforce Object Query Language) in the case of Salesforce.

On the Salesforce side, you first select the Salesforce entity. We select “Opportunity” as shown below.

Create Trigger

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

Trigger will apply to selected entity type* ⓘ

Opportunity ▼

Use search query ✕

Select conditions to filter Opportunity for synchronization:

Name	Description	Next Step	Fiscal Period
<input type="text" value="Name"/>	<input type="text" value="Description"/>	<input type="text" value="Next Step"/>	<input type="text" value="Fiscal Period"/>
Tracking Number	Order Number	Current Generator(s)	Main Competitor(s)
<input type="text" value="Tracking Number"/>	<input type="text" value="Order Number"/>	<input type="text" value="Current Generator(s)"/>	<input type="text" value="Main Competitor(s)"/>

For Salesforce, there are 2 ways you can deal with the “If” section. The first is to fill in the text boxes with proper information about the Opportunity you want to filter, as shown on the screen above.

Or toggle the “Use search query” switch to enter a SOQL query as shown below.

```
StageName= 'Prospecting' AND Name like '%demo%'
```

Notes

This search query defines Opportunity stage as 'Prospecting' and Opportunity name that partially matches the word 'demo'

To learn more about how to create them for specific platforms, refer to this [one](#) for Azure DevOps or [this](#) one for Salesforce.

Leave “Notes” so you know the purpose of your trigger.

There is a toggle switch that lets you activate or deactivate a trigger. This is useful when you don’t need the trigger and don’t want to create it from scratch later. Click the “Add” button once you are done.

The trigger you have created will be listed as shown below.

You can choose to edit or delete the trigger from here. To synchronize existing work items matching your trigger criteria click on the 3 dots and select “Bulk Exalate”.

» Azure DevOps_to_Salesforce

● Active

[← Back to Connections](#)

[Publish](#)

Rules

Triggers




Statistics

Info

Create a trigger to synchronize entities automatically.

All entities 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"/>	  

- Bulk Exalate
- Bulk Unexalate

< 1 >

Once you are done, click on “Publish” to apply the changes to your connection.

Step 6: Synchronize Information between Azure DevOps and Salesforce

For the Basic mode connection, individual work items and Cases can be synced by following the procedure mentioned in step 2.

You can also create triggers or use the “[Bulk Connect](#)” operation for syncing information with the Basic mode. There is also a “[Connect](#)” operation on the Azure DevOps side that you can use to synchronize work items.

Information in the Script mode connection can be shared by using the “[Connect](#)” operation in Azure DevOps, by creating triggers, or by using the “[Bulk Connect](#)” operation in both Salesforce and Azure DevOps. In addition to this, you can always edit what information leaves your system and how you interpret incoming information using the sync rules.

Exalate periodically checks for new synchronization requests or existing sync updates. So if you don’t see your sync result immediately, wait for some time and try again.

Common Use Case

Development and Sales Teams

An Azure DevOps Salesforce integration can be useful for both development and sales teams.

Suppose your sales team uses Salesforce and your development team uses Azure DevOps.

The sales team always has access to customer feedback, queries, and new feature requests. Integration between these applications would mean that they can request bug fixes and dev updates right from within Salesforce. This in turn would create a new work item in

Azure DevOps.

The development team would then work on it and update the statuses while doing so. An automatic bi-directional synchronization here would mean the sales team having real-time updates about the work items they have raised, and always having the correct answers for their customers.

This would also hold true in the opposite case. Suppose the development team is working on a frequently requested feature update. They send this across to the sales team on Salesforce, maybe as a “Task”. The comments, statuses, and other relevant information can then be exchanged between Salesforce and Azure DevOps, helping the sales teams to close deals that are pending due to that particular feature request faster and in a more organized manner.

Conclusion

This guide served to explain why an Azure DevOps integration is essential to bridge the gap between sales and development teams. How it helps alleviate problems these teams face due to manual data entry mistakes, thus wasting a lot of valuable time and effort.

We also saw how an automatic and real-time information exchange between Azure DevOps and Salesforce can help us have a consistent and reliable view of revenue-driving information.

We also discussed how Exalate offers features suitable for such an integration. And then finally had a look at how an Azure DevOps Salesforce integration can be implemented by following a few straightforward steps.

Recommended Reads:

- [How to set up a Jira Salesforce Integration](#)
- [Salesforce to Salesforce Integration: Sync Multiple Salesforce Instances Bidirectionally](#)
- [Jira Azure DevOps Integration](#)

- [GitHub Salesforce Integration: How to Set up a Sync in 6 Steps](#)
- [How to Set Up a Salesforce ServiceNow Integration](#)
- [Salesforce Zendesk Integration](#)
- [How to Set up a Zendesk Azure DevOps Integration](#)
- [How to Set Up an Azure DevOps ServiceNow Integration](#)
- [How to Set up an Azure DevOps GitHub Integration](#)