# Salesforce Integrations: Integrate Salesforce and other Tools in 2023

## Table of contents

Almost every sales and marketing team has used Salesforce at some point in their work life. The sheer popularity of this platform is evident in the increasing user base, enhanced customer experience, better sales cycles, and immense customizability.

When such a powerful CRM platform needs to be integrated with other tools like GitHub, Jira, or Azure DevOps, there is no way that can be done out of the box.

To give the user the comfort of staying in the tool they already use and still get the information they are looking for, it's necessary to integrate Salesforce and other platforms.

So without further ado, let me make clear the intention of this blog.

It is to introduce why Salesforce integrations are needed and how they can be achieved using the native capabilities of Salesforce. We will also see how 3-rd party integration solutions might be a better alternative. And then we move on to see an actual implementation using an integration solution called Exalate.

What's covered in this article:

## What you Need to Know about Salesforce Integrations

Salesforce is widely used as a CRM application, and different teams like sales, marketing, or customer service use it on a daily basis.

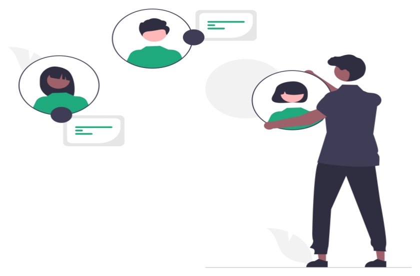
And then there are other applications like Jira, GitHub, Azure DevOps, Zendesk, etc which have their own set of customers.

Information needs to flow between all these different platforms. But if done manually, it can be time-consuming and prone to errors. So we must resort to automatic ways to do so.

Salesforce third-party integration solutions can streamline collaboration between teams.

There are many ways such an integration can be pictured:

- Data can be converted to a common format and be synced or exchanged between applications.
- Business processes, aka business logic, can be automated end-to-end. This helps in streamlining collaboration across different teams. This is also known as B2B integration.
- Provide a centralized user interface to teams using different applications so that they can access or manage the information through it.



These integrations can be either synchronous or asynchronous depending on the use case in question.

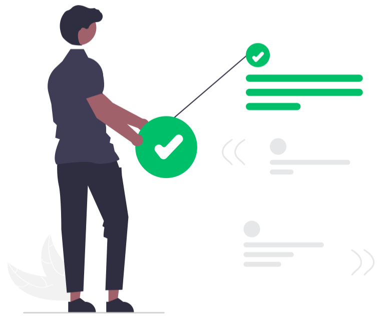
Another way these Salesforce integrations can be classified is based on the timing of the information to be exchanged:

- Sometimes you need to integrate information that changes less frequently. It changes at a certain time period: monthly, daily, hourly, etc. So at this specific time interval information is either brought into Salesforce or retrieved from Salesforce and sent to some other application.
  In short, the information exchanged is not in real-time. You can also use triggers to make this happen. Say an Opportunity becomes 'Won', only then transfer the details to another application say Jira, to be taken up by the license management team for further processing.
- Nearly most of the use cases depend on real-time integration between Salesforce and other platforms. This kind of exchange gives insights to the sales team since 'time' is of the essence to them.
  A spin-off of this can also be a near real-time integration where a few moments of delay, maybe a couple of minutes, in information exchange does not harm anything. It is a better option since it is more predictable than real-time exchanges. For instance, when there is a confirmation about a purchase order that needs to be passed between 2 systems, then near real-time can be a safe bet.
- As discussed a while ago, customized UIs can also be created in Visualforce to implement an interface that is used to quickly evaluate some information coming in from an external system.
- And then we come back to square one, Salesforce, and how we can make it our single source of truth. We head over to the AppExchange and see a variety of apps purpose-built for integrations. They are there because they recognize the customizability Salesforce offers, and make integrations with it easier and quicker to set up. We will come back to this point in a while.

## Benefits of Salesforce Integrations

Moving forward, let's have a quick look at what benefits a Salesforce integration would bring to the table:

- Avoid context switching or toggling between applications to look for information. Have it handy in the platform you already are familiar with.
- Increase team productivity and reduce friction. This is a byproduct of automating business processes since you bid adieu to manual ways of doing things.
- Accurate information is exchanged through integrations, which in turn provides teams with better decision-making insights.
- Efficient communication across teams as diverse tools start interacting and collaborating with each other.
- Integrations are as much for business teams as they are for technical teams, so a range of audiences can benefit and get aligned on common business goals.
- Transparency between teams and visibility of critical information comes naturally to integrated systems.

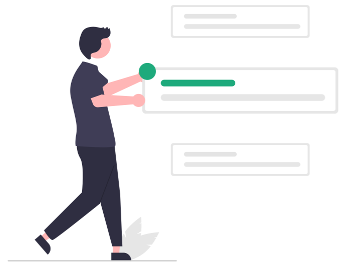


After understanding the what and why let's see a few practical use cases of Salesforce integrations.

## Common Use Cases

We will see these use cases in the form of popular patterns that are often demanded from an integration.

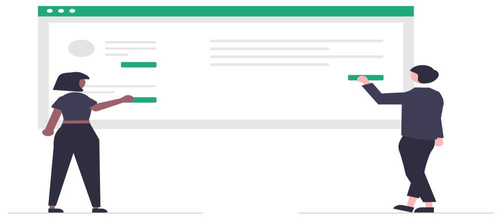
## Use Case 1: Migrate Customer Data from External Systems to Salesforce



Sometimes information residing in external systems needs to be migrated to Salesforce. In such a case, the scope of the migration needs to be planned, mappings need to be defined and then the actual exchange of data takes place.

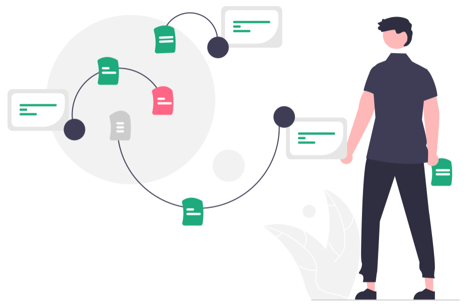
Often such a use case is suitable for migrating from a legacy system to Salesforce or consolidating different CRM systems (a Salesforce to Salesforce integration) or a common customer inventory is needed.

## Use Case 2: Transfer Customer Information between Salesforce and Other Systems (One to many)



The sales team, for example, is keeping track of customer information in their Salesforce instance in the form of Opportunities, Cases, or Accounts. There can be a need to reflect this information into systems like Jira or GitHub, or Azure DevOps. This can then be taken up by the back office and development teams respectively. The back office teams using Jira might handle the licensing and contract management tasks whereas the dev team can look into customer cases raised as issues in GitHub or Azure DevOps. And all this information can then flow back and forth smoothly between these systems increasing visibility for the teams involved.

## Use Case 3: Present Multiple Related Salesforce Objects into External Systems (Many to One)

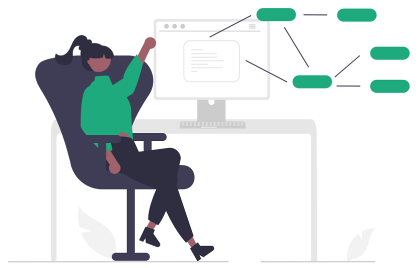


Sometimes you need to sync multiple related Salesforce objects to Jira.

Salesforce has related objects like Cases belonging to a particular Account or Contact. And then these need to be reflected onto other systems, maybe under Jira as issues. Since these 2 systems have different fields and entities, information coming from Salesforce can be reflected in custom fields.

Here, maintaining the Case, and its related Account and Contact information under the correct Jira issue is important. Changes or updates on the Salesforce data must reflect correctly on the Jira side.

## Use Case 4: Sync Salesforce Records to the Development Team



Customer queries, feedback, or issues are often logged into Salesforce Cases. When these useful customer insights are converted into Jira or GitHub issues or Azure DevOps work items, then they can be taken up by the development team to be worked on further.

Comments, statuses, and attachments (if any) can be bi-directionally synced between all these systems. This can give visibility to the sales teams so they can communicate correctly to the customer about the status of their query and enhance their overall experience.

You can also envision a workflow orchestration between Salesforce and other systems. For instance, when a customer case belonging to a particular Account, under the State = 'New' is created, then it is sent over to Jira. When the dev team starts working on it, an 'In Progress' status in Jira puts the case under 'Working' in Salesforce. Comments and attachments are synced. And then when the dev team finally resolves the issue and puts it under 'Done' they pass a comment: 'Issue is fixed, marked for release' over to the sales team. Then the case is marked 'Solved' in Salesforce and the comment from Jira is reflected too.

## Use Case 5: Correlate Data between Salesforce and External Systems

A one-to-one connection between entities residing in Salesforce and other systems is sometimes needed. Say incidents, problems, RITMs, etc in ServiceNow need to be connected to a particular case in Salesforce. Then the proper status updates, work notes, and other custom fields need to be synced. This can bring useful insights for the customer support and sales teams.

As you might have already guessed, the possibilities are endless. So keeping them in mind, we move forward with the ways in which Salesforce integrations can be implemented.

## How to Implement Salesforce Integrations

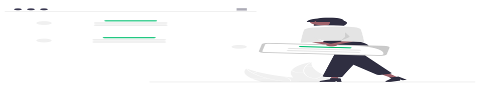
We have talked enough about how Salesforce is a great CRM tool, boasting of customizability. So while we are at it, let me point to the native ways in which Salesforce allows you to enhance its connectivity and reach, by allowing integrations with external systems.

### Native Ways to Implement Salesforce Integrations

Of course, this seems like the most logical choice when it comes to implementing Salesforce integrations. But before we dive deeper into the nuances, let's first see how it can be achieved.

The way Salesforce categorizes integrations with other 3rd party applications is through 'Inbound' and 'Outbound' messages. Simply put, inbound implies data coming into Salesforce from another system and outbound means data leaving Salesforce and being fed into an external system.

#### What are inbound and outbound messages?

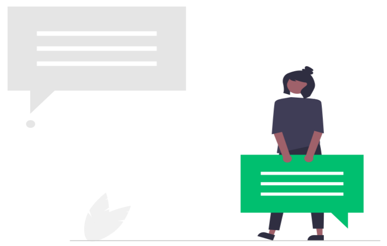


Inbound messages work on data-based integrations. Generally used when Java, .Net or PHP code needs to interact with Salesforce. Here, the external system, acting as a client,  prepares and submits a request to fetch data from Salesforce, acting as a server. The data is fetched from the server and a response is generated. This response is then handled by the client, i.e the external system.

Outbound messages work on business logic integration. Here, Salesforce prepares and submits a request to send to an external system. The external system receives the request and sends an acknowledgment, but the request isn't processed just yet. Once it is, a callback is provided that contains the result of the initial request. The result is then processed by Salesforce.

For Inbound Salesforce integrations:

- **REST API**: This is used for different operations on the data. Works in both JSON and XML format and is lightweight. It supports synchronous operations.
- **SOAP API**: It uses XML for structured payloads and supports asynchronous operations.
- **Bulk API**: This is used for high-volume transactions. It is best used if you want to migrate data to Salesforce and it supports asynchronous operations.
- **Salesforce Connect**: This allows you to view or edit remote files or data within the Salesforce environment as external objects. So they can be searched in the global search or added to the record feeds. But this data does not persist in Salesforce. It supports real-time data access. It has a point-and-click interface.



For Outbound Salesforce integrations:

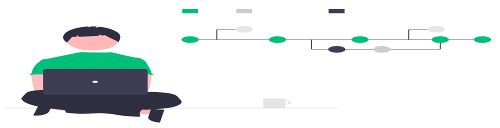
- **Apex callouts**: Apex is a powerful medium that Salesforce provides to enhance its functionalities. It can be used to transfer data from Salesforce to an external system. For instance, when an Account record is updated, then the update must be reflected on the external system as well.

There are also others like Streaming API, Apex Web services, etc.

We have seen a lot of ways Salesforce integrates with other systems. So let's have a look at the challenges some of these alternatives pose.

## Challenges of Native Salesforce Integrations

- Salesforce connect works best for Salesforce to Salesforce integrations, but Salesforce is in the driving seat of such an integration. The external objects viewed within Salesforce are read-only and additional configuration efforts are required to make them editable. There is also an additional price on it for certain Salesforce editions. Since it is a point-and-click interface, integrations are bound by the native UI capability.
- Apex is an alternative if deeper integrations are required, but honestly, it is neither designed nor optimal for providing integration capabilities. To perform integration using Apex, we need technical resources, which might not be readily available. It is also bound by governor limits.
- The other pitfalls for the alternatives we discussed above are limited data durability, lack of strong security mechanisms, and limited read/ write capabilities.



You can still build your own apps using the Salesforce App Cloud platform. But it comes down to additional development efforts and still no guarantee that most integration scenarios will be covered. They also sometimes require to be complemented by middleware applications on AppExchange.

## 3rd-Party Integrations

So the best alternative will be to leverage the breadth of the apps that Salesforce has to offer on AppExchange. They are professional, provide ready-made integrations to get you up and running in no time, and support a lot of different deployments and pricing models.

Now I understand you are caught between a rock and a hard place here, so let me make this easier for you.

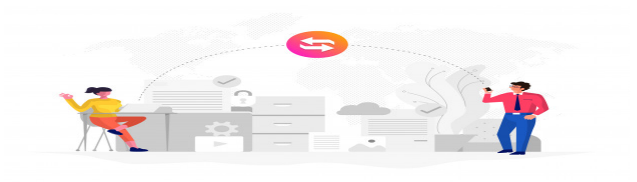
Let's first set the criteria for the right 3rd-party solution.

### How to Choose the Right 3rd-Party Integration Solution

#### Decentralized Integration

Almost all the solutions out there in the market are centralized. The major problem with a centralized solution is that the integration requirements need to be mapped out over a central system, so there is an increased dependency on it.

Also, the centralized solution becomes the driving force for carrying forward the integration and thus becomes a single point of failure.



As opposed to this, a decentralized integration will offer the benefit of independent control at either end of the integration. This makes the systems less dependent on each other, and such loosely-coupled systems then offer greater security, flexibility, and scalability. This brings us to our next few points.

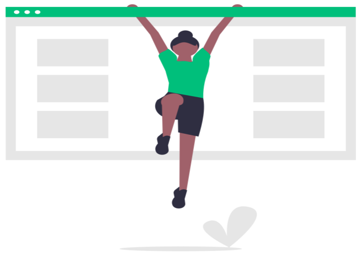
### Security

When 2 systems are not dependent on a central UI to manage their integration, they get to decide what information must be sent and received independently of one another. So in essence, you do not send the information you don't want the other party to access, and receive only the information you need. This avoids the possibility of unauthorized access to information.

Also in addition to this, necessary security measures like token-based authentication, use of HTTPS, role-based access, etc are mandatory.

### Flexibility

Support for basic to advanced integrations is what these tools are made for. So the more flexible the solution, the more is the possibility of assimilating it into your ever-growing integration requirements. This acknowledges the fact that businesses grow and so do integration requirements.



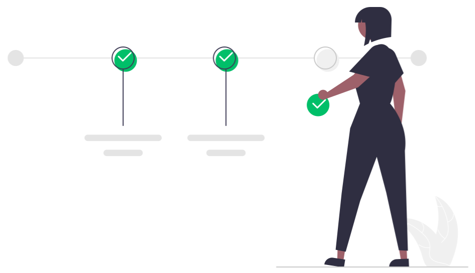
Another aspect to consider is the kind of users the solution caters to. It's common that the integration is run by business and technical teams, so it should be equally useful to both.

### Reliability

Integration is to replace manual interactions and thus avoid errors. It also ensures the data exchanged is accurate and consistent. So reliability plays an important role to determine better results for your integration. An automated mechanism to retrieve changes from the last downtime or system failure and apply them in the correct order of their initiation must be at the core of the solution.

### Number of Supported Integrations

After all, you want a solution that supports a number of diverse platforms. Think Jira, Azure DevOps, GitHub, and the like. This will give you the liberty to connect to just another of your customer, partner, supplier, or vendor without you having to build or configure a completely new integration by yourself.



A major point I would like to stress here is that any integration solution you consider needs to allow the teams to grow the integration into what is required by the business. This includes functionality, security, governance, and other concerns. This can only be achieved by a tool flexible, reliable, and secure enough to handle this for you.

The tool I will use here is called Exalate.

Let’s see what it has to offer:

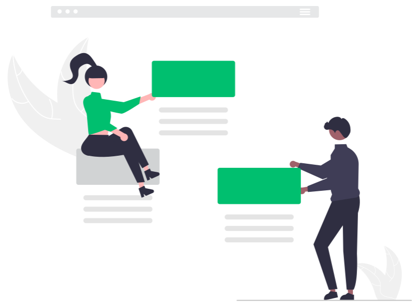
- It supports decentralized integration. As already discussed, this means that each integrating side has complete, independent control over incoming and outgoing information. So it increases the inherent security within your integration. Exalate achieves this with the help of a scripting engine that has Sync rules, that can be configured at either end. We will see how that’s done in a while.
- This scripting engine offers flexible and deeper integrations. It also comes with a drag-and-drop interface for common integration scenarios, but even this can be configured for advanced use cases.
- It uses a JWT-based token mechanism, HTTPS, and also provides role-based access control. All this can be explored in detail in its security and architecture whitepaper.
- It supports integrations for many popular platforms like Jira, Zendesk, Azure DevOps, GitHub, HP ALM, etc. So you can simply set up a Jira Salesforce integration or you can also get the same platform integrations using Exalate like a Salesforce to Salesforce integration.
- With the help of an integrated retry mechanism, changes or updates can be applied automatically in case of downtimes.

It’s time we pack a punch and deliver an integration.

Let’s see how.

## How to Set up Salesforce Integrations Using a 3rd-Party Solution

In this section, we are going to set up a Salesforce integration using a 3rd party solution. In case you have jumped to this section directly, the tool in action is Exalate.



Exalate is a 3rd-party integration solution that allows you to set up a flexible 2-way sync between different tools like Jira, Salesforce, GitHub, etc. It helps unlock collaboration between teams across company borders, or even between different departments or projects within a single company.

But as a general rule, before starting with the implementation, it is important to lay the foundations of integration well.

Let us now walk through the step-by-step approach for a Salesforce integration using Exalate.

## Step 1: Lay the Groundwork

Start with extensive planning of technical/ business resources and docs, map the data flow,  orchestrate the workflows end-to-end, and set the data exchange requirements. Consider your integration as just another project, having specific roadmaps, guidelines, goals, roles, and responsibilities.

Chalking out a detailed integration plan either within a company (intra-company integration) or across different companies (cross-company integration) is important to deliver the benefits sooner.

When it comes to Salesforce integrations, it is also important to consider the type of integration desired (as discussed above) and review the API limits.

Proper authorizations and authentications must also be in place.

A major point here is also to ensure that the difference in the architecture of the 2 integrating systems is taken into account.

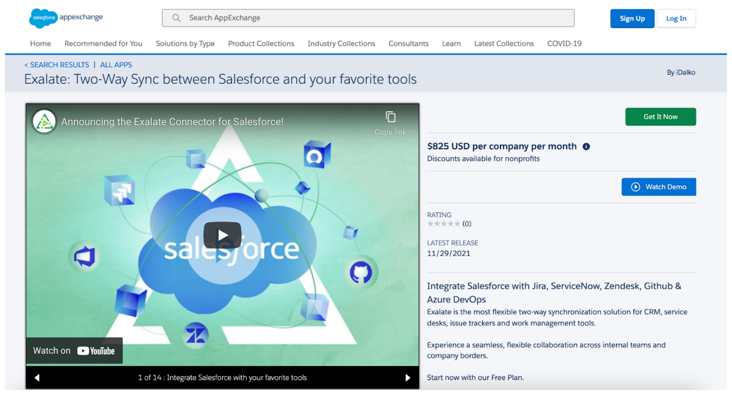
Having undertaken this, prepare to get started with Exalate and move to the next step.

## Step 2: Install Exalate on Salesforce and the Other Application

Exalate first needs to be installed as a dedicated app on Salesforce and the other application you want to integrate it with. It can be any one of the different integrations Exalate supports.

You can install it on Salesforce from AppExchange.

Search for the app, and click on ‘Get it Now’ to start the installation process.



Exalate: Two-Way Sync between Salesforce and your favorite tools

By iDalko

$825 USD per company per month
Discounts available for nonprofits

RATING
★★★★★ (0)

LATEST RELEASE
11/29/2021

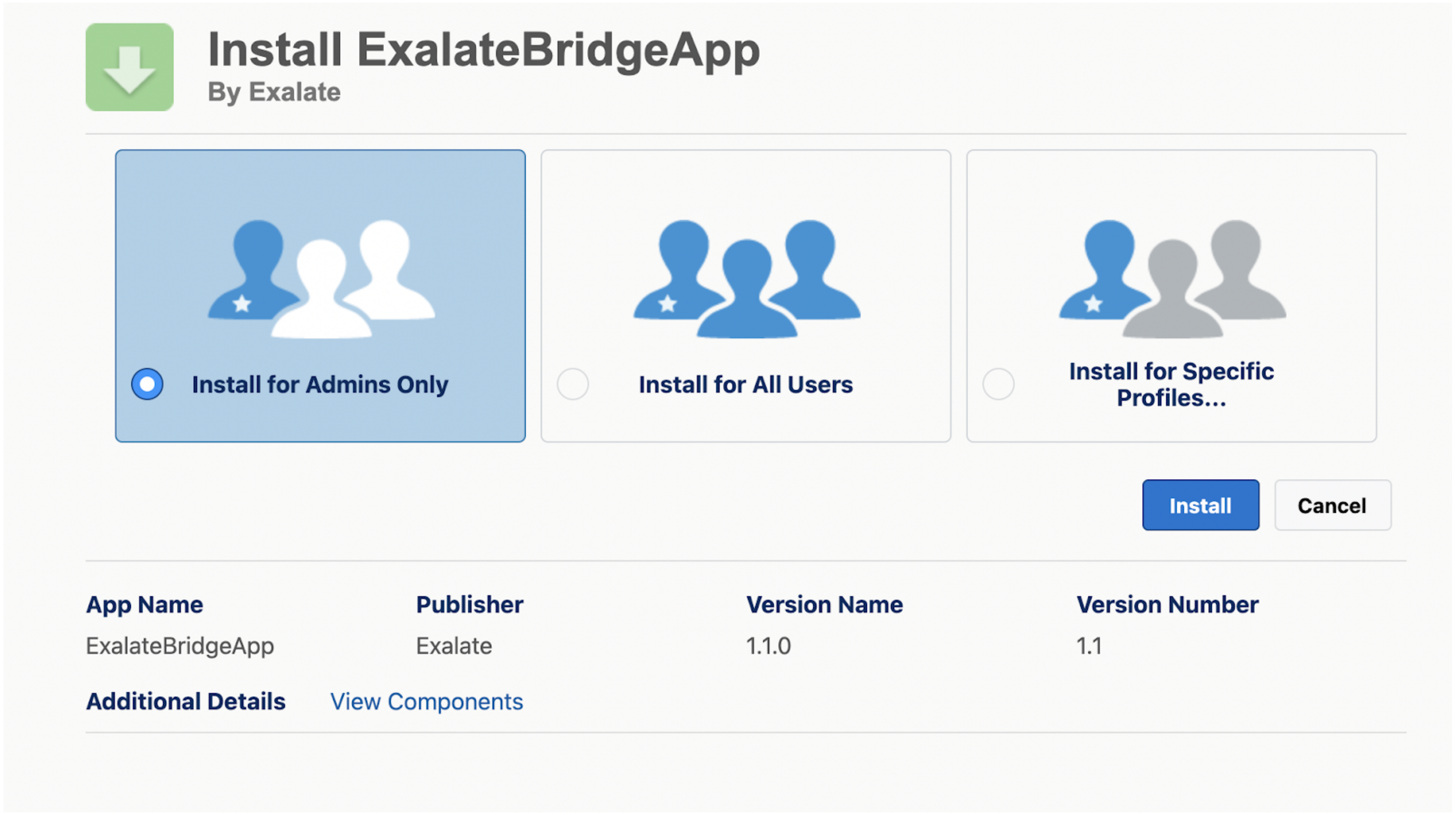
Integrate Salesforce with Jira, ServiceNow, Zendesk, Github & Azure DevOps
Exalate is the most flexible two-way synchronization solution for CRM, service desks, issue trackers and work management tools.

Experience a seamless, flexible collaboration across internal teams and company borders.

Start now with our Free Plan.

You then get to choose the production environment: Production or Sandbox.

Next, choose the users who will have access to the Exalate app.



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

After a few more steps, you can request an Exalate node by filling in your details like email address, etc.

And that's it from the Salesforce side.

As stated earlier, you can go to the other instance and install Exalate on it. Head over to the documentation, choose the correct platform, and then move ahead to the next step.

***Note**: If you prefer, you can even have a look at the installation videos of different platforms here.*

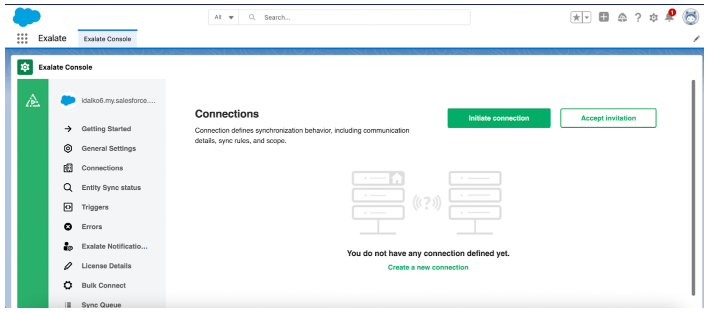
## Step 3: Connect Salesforce and the Other Application

Once Exalate is installed on both sides, you need to go to the 'Connections' tab in its console. A connection needs to be created.

It first authenticates both the source and the destination platforms and then creates a secure passage through which data can be transferred back and forth according to the rules you have set.

One side initiates the connection and the other side accepts the connection invitation.

The Exalate UI is the same for all platforms, so it doesn't matter which side you start from.



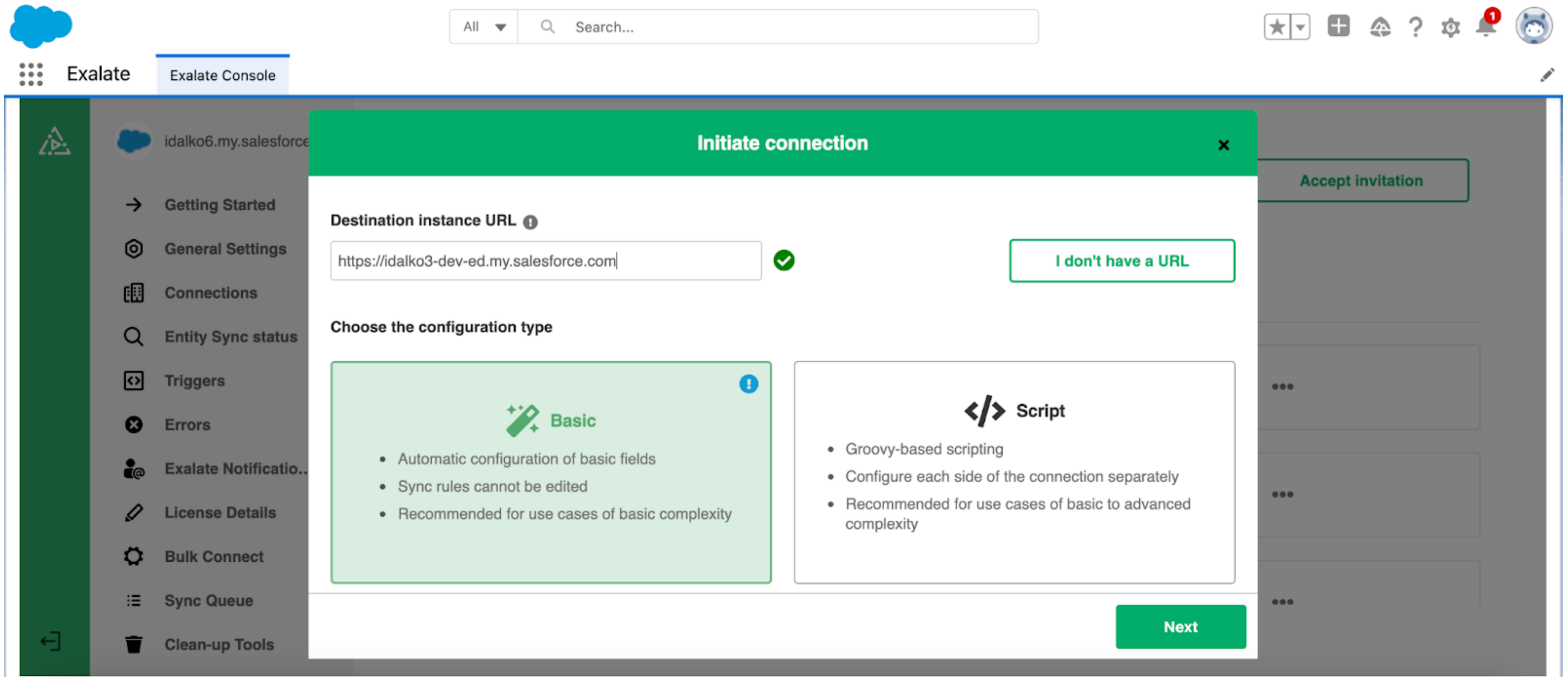
Connections

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

You do not have any connection defined yet.

From your Salesforce instance, click ‘Initiate connection’ under the ‘Connections’ tab.

On the next screen, enter the destination instance URL. This can either be another Salesforce instance or Jira, GitHub, Azure DevOps, etc.



Initiate connection

Destination instance URL

https://idalko3-dev-ed.my.salesforce.com

Choose the configuration type

**Basic**
- Automatic configuration of basic fields
- Sync rules cannot be edited
- 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

Once done, a quick check is performed to check if Exalate is installed on the destination side.

Based on the destination instance you will be prompted to choose between 2 different modes.

The **Basic mode** and the **Script mode**.

The **Basic mode** is for simple integrations like when you want to sync a Case from Salesforce to Azure DevOps, or if you want to sync an issue (issue type, summary, description, comments, and attachments) from Jira to Salesforce.

The sync cannot be configured in this mode, so it works best if you want to give Exalate a try for the first time. But it does come with a Free Plan that supports up to 1000 free syncs per month.

What makes Exalate stand out is the **Script mode**.

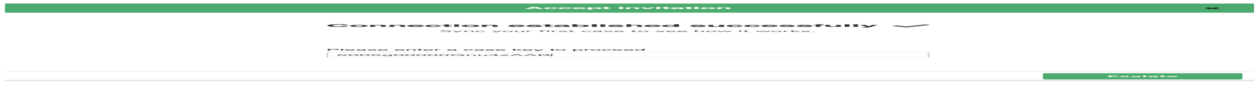
It has an intuitive Groovy-based scripting engine that allows you to control your sync from either end independently with the help of outgoing and incoming processors. That controls information coming in and going out of Salesforce or the other application. These processors can be configured with the help of Sync rules. We will see how it can be done.

Let's first see how both these modes work.

## The Basic Mode

Choose the Basic mode on the screen above and hit 'Next'. You will be redirected to a screen asking you if you have admin access to the other side.

In case you do, depending on the destination application, you will be either prompted to choose a project (for instance, in Jira or Azure DevOps) or a repository in the case of GitHub or a connection will be established straight away.


Accept invitation

Connection established successfully

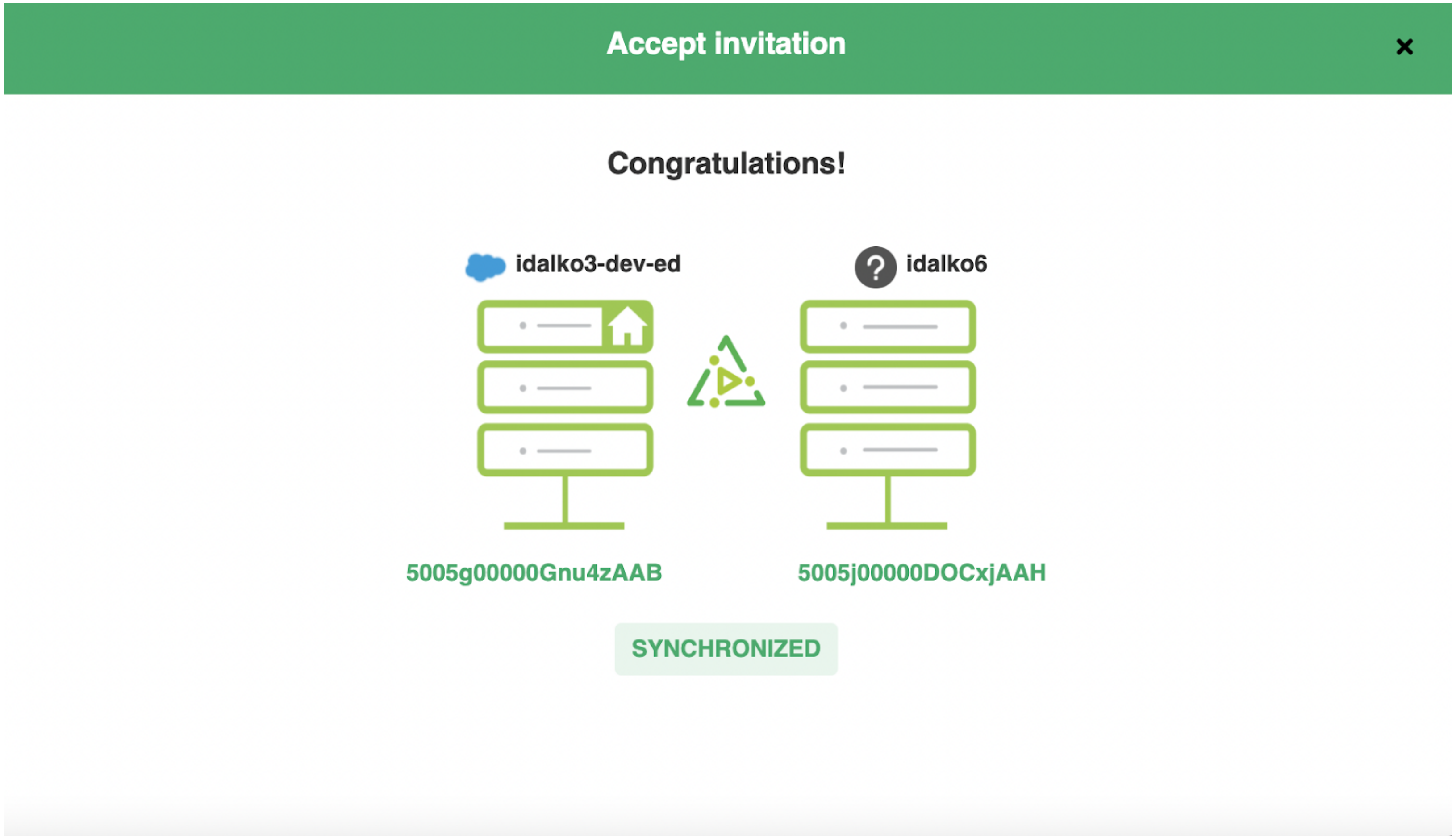
Sync your first case to see how it works.

Please enter a case key to proceed

5005g000000Gnu4zAAB

You can start by entering the Case key directly and get it synced over to the other side.

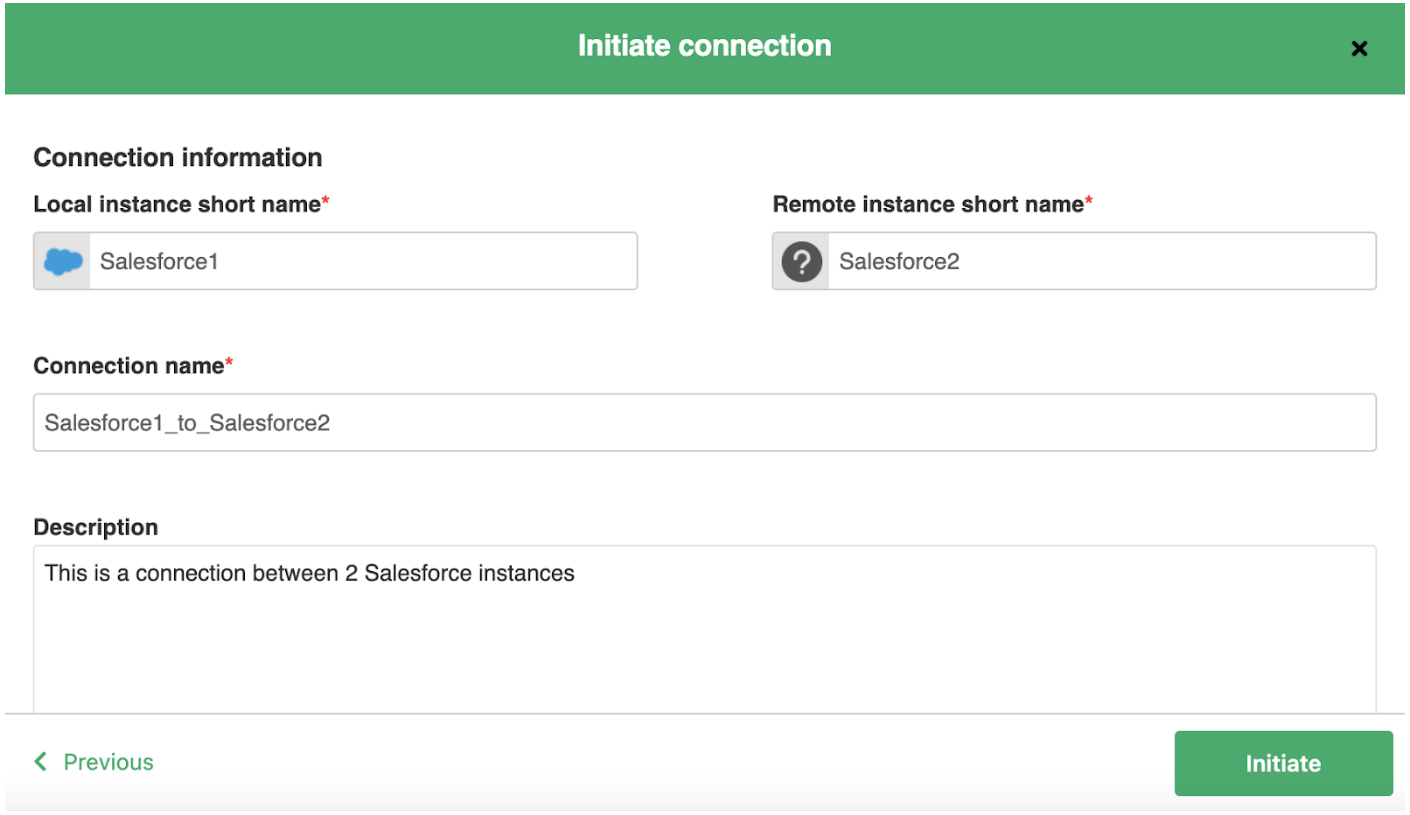
You can even navigate to the other side and enter the unique identifier of the entity you want to sync.



## The Script Mode

Let's try out the full functionality of Exalate. So choose Script and click 'Next'.

Enter the details for the remote and the local instance. You can change the connection name if you want. Enter a description too. And click 'Initiate'.



Initiate connection

Connection information

Local instance short name*: Salesforce1

Remote instance short name*: Salesforce2

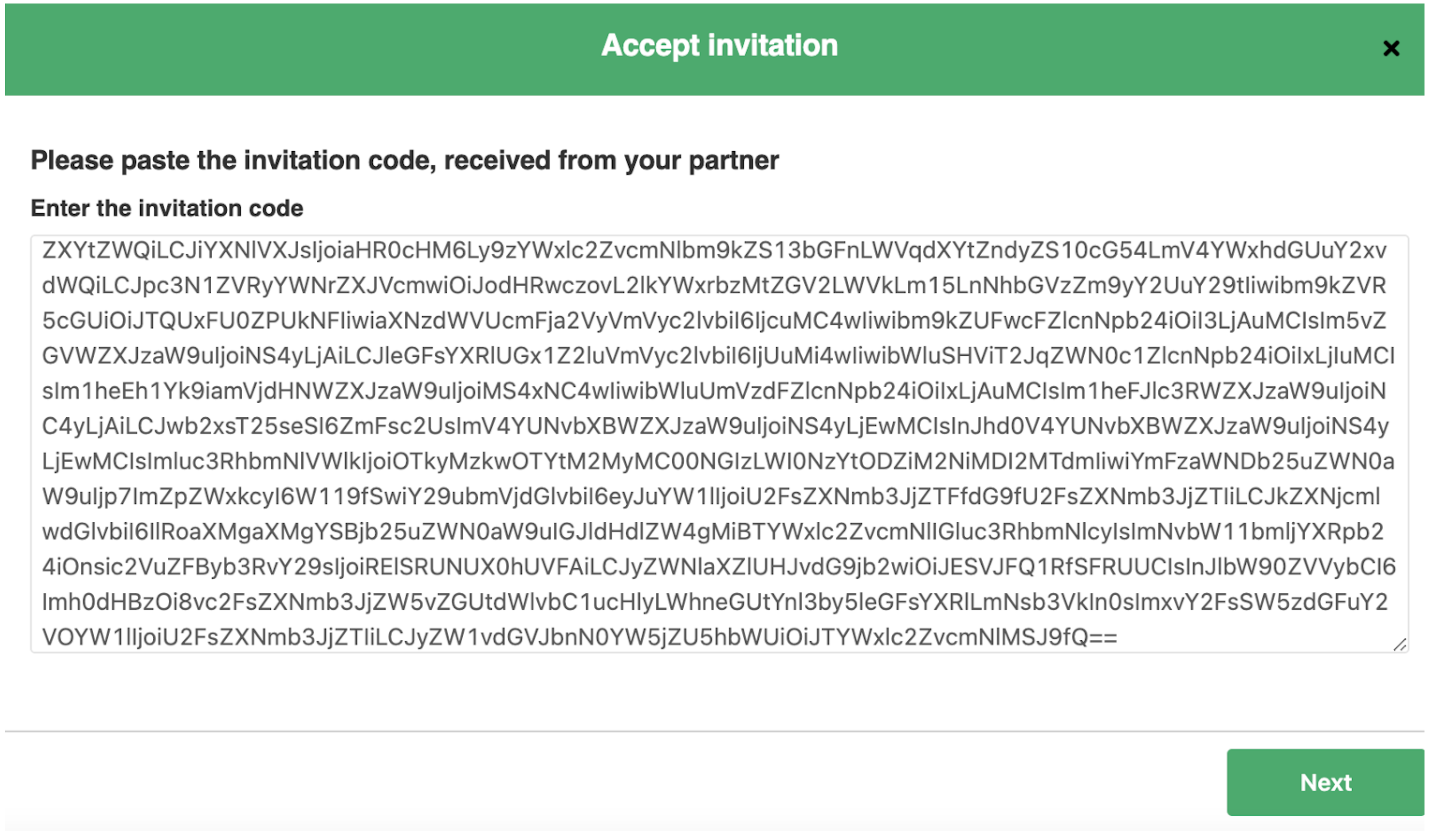
Connection name*: Salesforce1_to_Salesforce2

Description: This is a connection between 2 Salesforce instances

Next, an invitation code will be generated. This works like a shard secret that authenticates both integrating parties. Click on ‘Copy invitation code’.

Go to the other side, and now click ‘Accept Invitation’.

Paste the copied code in the text area and click ‘Next’.



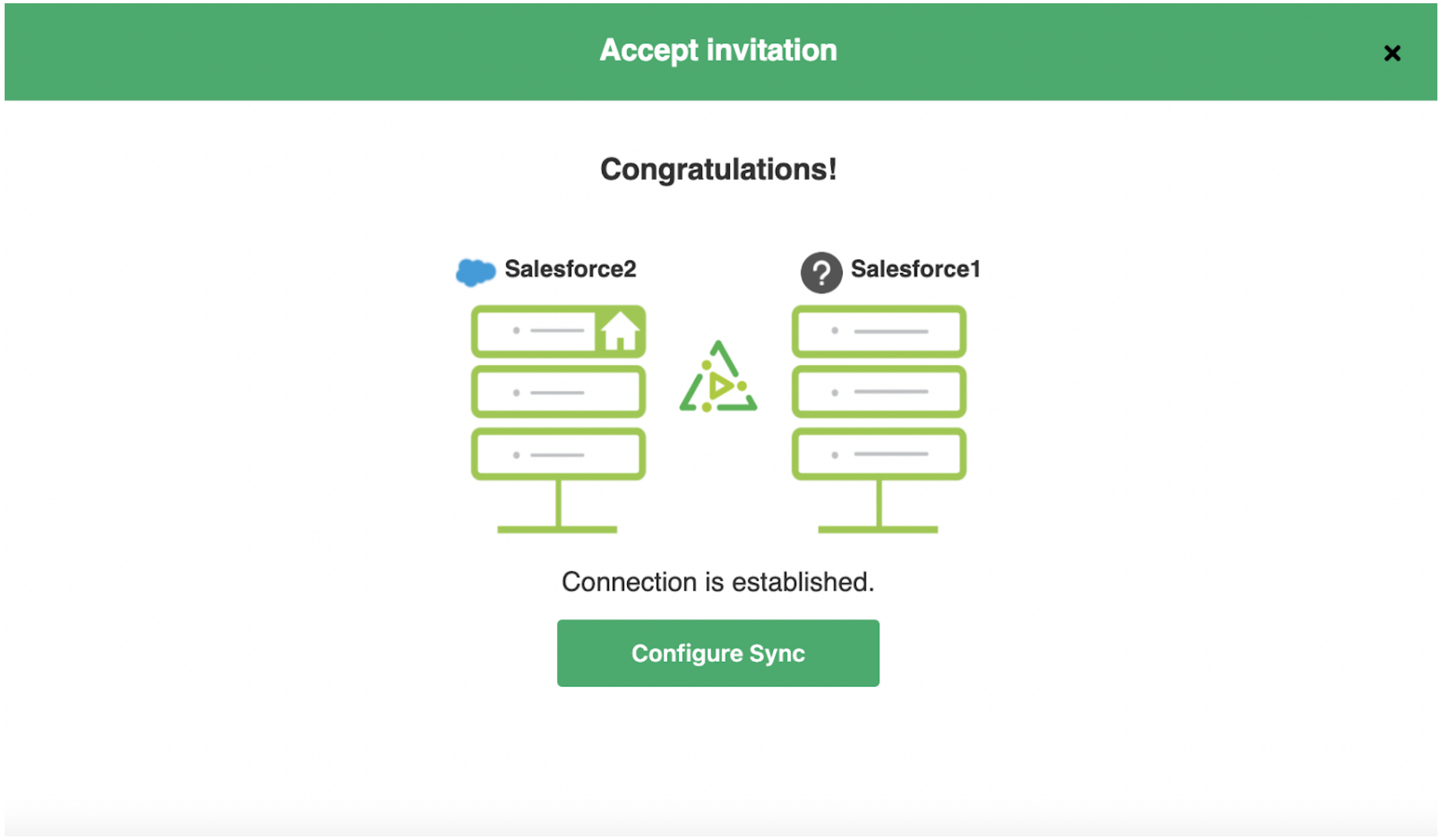
**Please paste the invitation code, received from your partner**

**Enter the invitation code**

ZXYtZWQiLCJiYXNlVXJsIjoiaHR0cHM6Ly9zYWxlc2ZvcmNlbm9kZS13bGFnLWVqdXYtZndyZS10cG54LmV4YWxhdGUuY2xv
dWQiLCJpc3N1ZVRyYWNrZXJVcmwiOiJodHRwczovL2lkYWxrbzMtZGV2LWVkLm15LnNhbGVzZm9yY2UuY29tIiwibm9kZVR
5cGUiOiJTQUxFU0ZPUkNFIiwiaXNzdWVUcmFja2VyVmVyc2lvbiI6IjcuMC4wIiwibm9kZUFwcFZlcnNpb24iOiI3LjAuMCIsIm5vZ
GVWZXJzaW9uIjoiNS4yLjAiLCJleGFsYXRlUGx1Z2luVmVyc2lvbiI6IjUuMi4wIiwibWluSHViT2JqZWN0c1ZlcnNpb24iOiIxLjIuMCI
sIm1heEh1Yk9iamVjdHNWZXJzaW9uIjoiMS4xNC4wIiwibWluUmVzdFZlcnNpb24iOiIxLjAuMCIsIm1heFJlc3RWZXJzaW9uIjoiN
C4yLjAiLCJwb2xsT25seSI6ZmFsc2UsImV4YUNvbXBWZXJzaW9uIjoiNS4yLjEwMCIsInJhd0V4YUNvbXBWZXJzaW9uIjoiNS4y
LjEwMCIsImluc3RhbmNlVWlkIjoiOTkyMzkwOTYtM2MyMC00NGIzLWI0NzYtODZiM2NiMDI2MTdmIiwiYmFzaWNDb25uZWN0a
W9uIjp7ImZpZWxkcyI6W119fSwiY29ubmVjdGlvbiI6eyJuYW1lIjoiU2FsZXNmb3JjZTFfdG9fU2FsZXNmb3JjZTIiLCJkZXNjcml
wdGlvbiI6IlRoaXMgaXMgYSBjb25uZWN0aW9uIGJldHdlZW4gMiBTYWxlc2ZvcmNlIGluc3RhbmNlcyIsImNvbW11bmljYXRpb2
4iOnsic2VuZFByb3RvY29sIjoiRElSRUNUX0hUVFAiLCJyZWNlaXZlUHJvdG9jb2wiOiJESVJFQ1RfSFRUUCIsInJlbW90ZVVybCI6
Imh0dHBzOi8vc2FsZXNmb3JjZW5vZGUtdWlvbC1ucHlyLWhneGUtYnl3by5leGFsYXRlLmNsb3VkIn0sImxvY2FsSW5zdGFuY2
VOYW1lIjoiU2FsZXNmb3JjZTIiLCJyZW1vdGVJbnN0YW5jZU5hbWUiOiJTYWxlc2ZvcmNlMSJ9fQ==

Here again, depending on the other application you will be required to either choose a project or a repository like in the Basic mode.

A connection then gets successfully established. We have shown a Salesforce to Salesforce connection here, but it can just be any other application you want.

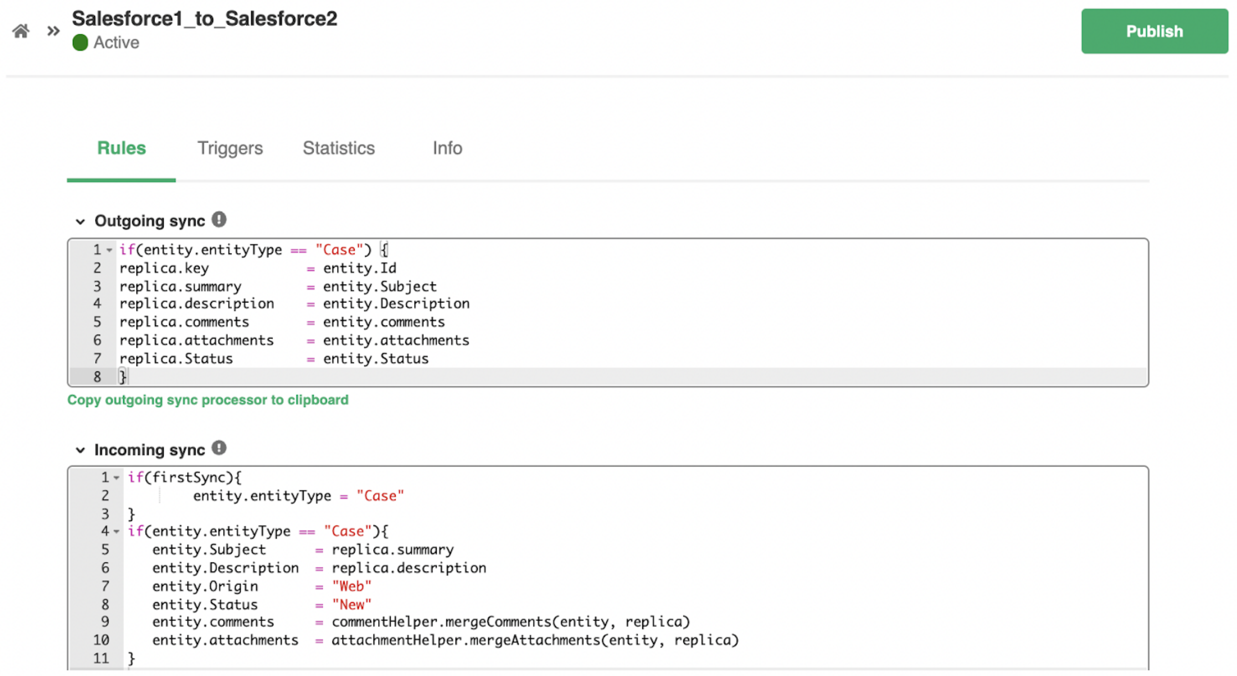


Move to the next step.

## Step 3: Configure the Connection to Control Information Exchange

To configure the connection, click the ‘Configure Sync’ button. Or you can even go to the ‘Connections’ tab and click the edit connection icon in front of the connection name.

Both approaches will take you to a similar screen.


Salesforce1_to_Salesforce2
Active
Publish
Rules
Triggers
Statistics
Info
Outgoing sync
Copy outgoing sync processor to clipboard
Incoming sync

Outgoing sync

```
if(entity.entityType == "Case") {
replica.key            = entity.Id
replica.summary        = entity.Subject
replica.description    = entity.Description
replica.comments       = entity.comments
replica.attachments    = entity.attachments
replica.Status         = entity.Status
}
```

Incoming sync

```
if(firstSync){
        entity.entityType = "Case"
}
if(entity.entityType == "Case"){
   entity.Subject      = replica.summary
   entity.Description  = replica.description
   entity.Origin       = "Web"
   entity.Status       = "New"
   entity.comments     = commentHelper.mergeComments(entity, replica)
   entity.attachments  = attachmentHelper.mergeAttachments(entity, replica)
}
```

You can see 4 tabs: ‘Rules’, ‘Triggers’, ‘Statistics’, and ‘Info’.

We will explore the ‘Rules’ tab here and the ‘Triggers’ in the next section.

Statistics and Info simply provide more information about the connection.

Rules are the Sync Rules we discussed a moment ago. They have both an ‘Incoming sync’ and ‘Outgoing sync’ to control information flow.

For instance, ‘Outgoing sync’ on Salesforce means what information must be sent out from Salesforce and ‘Incoming sync’ means how information coming from the destination platform is interpreted. The same rules are present on the other side too.

This information is passed back and forth in something called ‘Replica’. It works like a message or payload between the integrating applications.

For instance, *replica.summary=entity.Subject* in the ‘Outgoing Sync’ of Salesforce means that the subject of the entity (Case, Product, Account, Opportunity, Task, etc) is copied into *replica.summary.*

This *replica.summary* is copied into the summary on the other side, for instance, in Jira’s ‘Incoming Sync’ we will write *issue.summary=replica.summary.*

So if you want to send additional information, you can simply add it to the replica and send it over to the other side. There we extract the information from replica and apply it locally in any way that makes sense.

If there is something you don’t want to send or receive, you can simply delete the line or comment it, so it will be ignored at the time of synchronization.

Once you have decided *what* information must be sent and received, move to the next step where you decide *when* to send this information.

## Step 4: Set up Automatic Synchronization Triggers

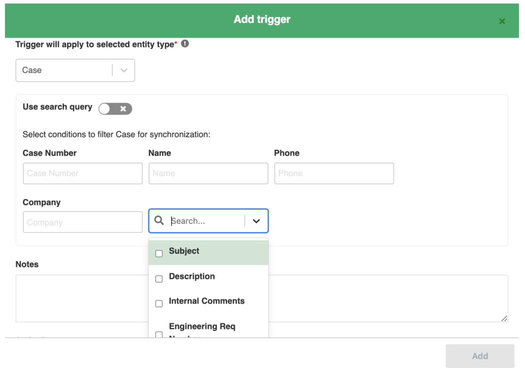
Triggers control when information exchange must happen. Say when a new Case is created i.e Status = 'New', then send it over to the other side.

In short, when the condition set for the trigger is met, then send and receive information according to the 'Sync rules'.

Remember that triggers are platform-specific, so on the other side, they will be either in JQL(Jira Query Language) for Jira or WIQL (Work Item Query Language) for Azure DevOps and so on.

Click the 'Triggers' tab from the last screen we saw. There is also a 'Triggers' screen on the left-hand side of the Exalate console. This is a global screen to manage triggers. You can create one from here too, but you need to select the connection name while doing so.

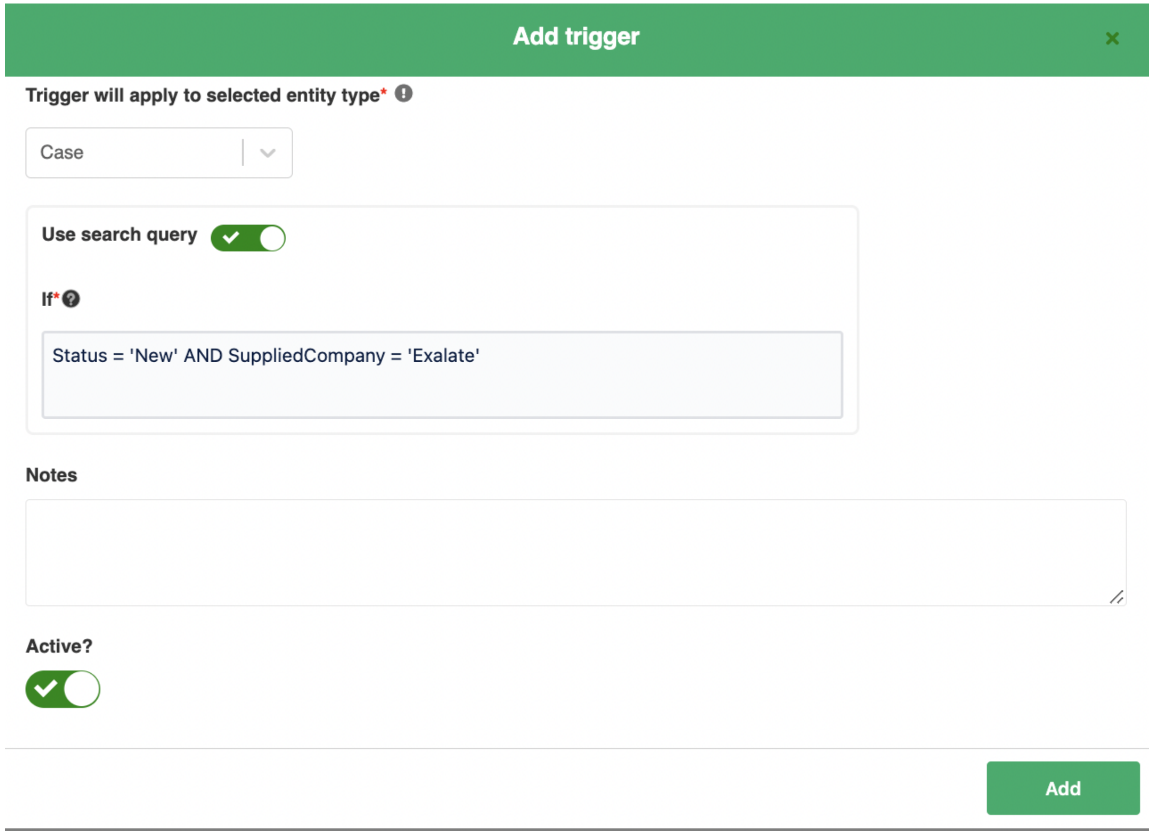
Click the 'Create Trigger' button to open the 'Add Trigger' screen.



Under the first drop-down, select the Salesforce object to which the trigger applies. It can be a Case, Account, Opportunity, or Product.

You can directly enter some values in the text boxes provided to filter what conditions to set for the trigger.

Or you can simply toggle the 'Use search query' button. This will allow you to enter triggers in the SOQL (Salesforce Object Query Language).


Add trigger

Trigger will apply to selected entity type*

Case

Use search query ✓

If*

Status = 'New' AND SuppliedCompany = 'Exalate'

Notes

Active? ✓

Add

For instance, if this is the query: *Status = 'New' AND SuppliedCompany = 'Exalate',* then it means that newly-created Cases having the Web Company = ‘Exalate’ need to be synced over to the other side.

Leave some notes for the trigger, activate it, and click ‘Add’.

You can view the newly created trigger on the previous screen. You can click the 3 dots in front of the trigger name and choose to edit or delete it. You can also sync the existing entities that satisfy the trigger condition using the ‘Bulk Exalate’ option.

Don’t forget to ‘Publish’ the changes to the connection.

### Step 5: Start Synchronization

⍰⍰There are multiple ways in which you can start synchronization.

- For the Basic connection, directly enter the Case number.
- Manually sync the Salesforce object through the Exalate panel under the object view. This option is also available for other platforms like Jira, Zendesk, etc. So just select the name of the connection you want to sync the particular object with.
- Create automatic sync triggers like we just saw.
- Use the ‘Bulk Exalate’ or ‘Bulk Connect’ option to sync objects that are already existing in Salesforce or other application.

## Things to Consider When Using Exalate

### Start with a Clear Integration Scope

With a flexible tool like Exalate that allows deeper integrations with the help of scripts, it is important to have clear integration goals and requirements set.

This also includes understanding what information must be sent and received and how it will impact security. Since decentralized integration that Exalate provides needs a deeper thought into the information exchange scope.

## API Limits

Exalate accesses Salesforce through APIs. Salesforce has its own guidelines for API Access add-ons. For instance, API access is provided by default in Enterprise accounts, while it is not the case with other accounts like Professional. This must be considered beforehand.

***Note**: Visit this documentation page to learn about the different Salesforce editions Exalate supports.*

## Access to the Exalate App

Managing access to the Exalate app is an important consideration since you don't want the information to be in the wrong hands or violate any data regulatory requirements.

This is possible by giving appropriate permissions while installing the app itself, i.e you can choose to permit either the admins, all users, or specific users. This can also be changed later on.

## Conclusion

In this blog post, we discussed what Salesforce integrations are and went over the benefits of integrating the Salesforce application with other applications like Jira, GitHub, Azure DevOps, Zendesk, or HP ALM by having a look at a few real-life use cases.

And then we looked at the native ways in which to achieve such integration. Of course, there are challenges there, but we addressed them by exploring a third-party integration solution called Exalate..

And then we saw a step-by-step approach to our integration by seeing how it can actually be set up using Exalate.

All said and done, Salesforce integrations can prove to be an effective decision for your business and we saw just that in our blog post.

***Recommended Reads***:

- Salesforce Third-Party Integration: Set up a Bidirectional Sync between Salesforce and Other Tools
- Jira Salesforce Integration: How to Set up a Two-Way Sync between Different Teams
- Salesforce to Salesforce Integration: Sync Multiple Salesforce Instances Bidirectionally
- How to Set up an Azure DevOps Salesforce Integration
- How to Set up a Salesforce ServiceNow Integration
- GitHub Salesforce Integration: How to Set up a Sync in 6 Steps
- Salesforce Zendesk Integration (Step-by-Step Guide)
- How to Sync Multiple Related Salesforce Objects (Contact & Account Linked to a Case) to Jira