exalate

eBonding Integration: The Ultimate 2024 Guide to Flexible Data Sync

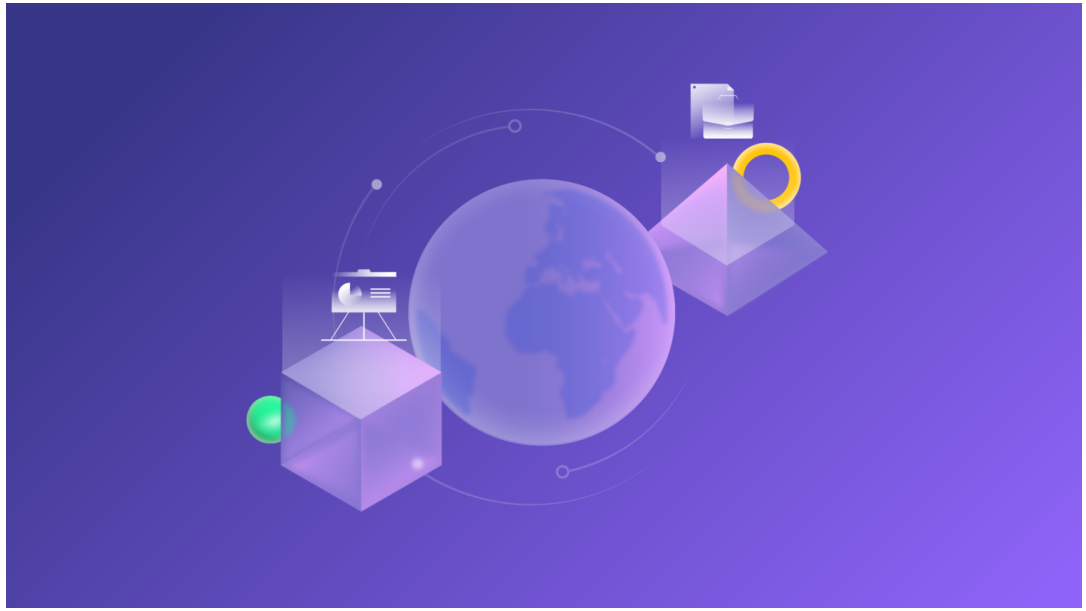


Table of contents

- IntegrationHub: eBonding Spoke
- Exalate

So what's my point?
Conclusion

In the material world, a bond is used to join 2 things together by means of an adhesive. But this bond can be extended to the digital world as well. That's what eBonding, literally electronic-Bonding, is here for.

And what exactly do we want to bond together? "Inherently disparate business applications" through integration.

In this article, we will have a look at how eBonding makes such integration a reality. We will also see why it is needed in the first place with a few real-life scenarios, then we will help you choose the right solution for it before finally seeing how it can be implemented.

So let's eBond!

What's covered in this blog post?

What is eBonding?

eBonding (also ebonding or e-bonding) is the use of automated connectors to synchronize information between different business applications so they always have matching data.

Changes in one system are reflected in the other, so they both deliver an end-to-end business process. Yes, eBonding doesn't just mean synchronizing data bi-directionally. It is a well-thought-out, well-planned process of integrating business applications to deliver an end-to-end automatic workflow, resulting in better value and service to customers.

For this reason, people sometimes also prefer to call it a B2B software integration methodology.

A little too much? Well, I will make it simpler. Let's start by going over the *why*.

Where did it all start?

eBonding isn't brand new. It traces its origins to the telecommunications industry where large customers were looking for solutions to enable their different ticketing systems to communicate automatically with one another so they didn't have to use emails to pass information every time.

Getting inspired by this early on, other industries began to acknowledge its power and importance. Further, as companies started providing managed services (MSPs) and outsourcing grew, so did the number of applications people used. With these different applications having different workflows and processes, it became difficult for information to be passed between them.

People surely found a way to deal with it, but it was manual, through emails and phone calls. They often switched between different applications to dig for the information they needed, giving rise to what's known as a "swivel chair approach".

But this way of information exchange was annoying and caused friction between them (because they were actually swiveling their chairs around). Naturally, these manual interventions also led to duplicated, misplaced, wrong, or altered data.

So they searched for automatic ways of "synchronizing information". This is when they found eBonding as a life-saving solution.

Why do you need it?

- Since the data exchanged through eBonding is automatic and in real-time, you don't need to get all familiar with someone else's application anymore. Enjoy consistent and coherent data in the comfort of your own familiar application without even having to lift a finger, literally.
- Inferring from the above point, you can expect automation and simplification of your business processes and workflows, leaving your teams to focus on what really counts.
- Implementing eBonding in the correct way can help you foresee a complete "digital transformation" of your business. It can help you connect with your customers, vendors, partners, or suppliers in a digital, secure, automatic, and reliable manner.

Are these reasons good enough for you to continue reading? If you are still debating it, I suggest you have a look at some of these practical use cases.

eBonding Scenarios

Here are the most common eBonding scenarios:

Intra-Company and Cross-Company

Outsourcing/ multi-sourcing or MSPs is a norm nowadays. It helps deliver projects faster and more efficiently by leveraging the right expertise.

So the scenarios that I discuss in this section will cover both: intra-company (within one company; maybe different teams, departments, or projects) and across company borders (let's call it cross-company) to integrate with their suppliers, vendors, customers, or partners.

I have specifically made this distinction because cross-company scenarios have different sets of challenges when it comes to eBonding, since we are talking about bonds extended outside companies; think of security, reliability, scalability challenges multiplied.

So let's delve a little deeper.

Intra-Company- the Case

A software company has its development team using GitHub to create and manage dev issues and the quality assurance team uses Jira to handle issues related to test cases, test plans, and their executions.

At the outset, everything should look fine, right? But not really! Problems have begun to creep in.

Every time a GitHub issue is worked on, it needs to be passed over as a new issue in Jira for the QA team to handle. If all the test cases work fine, an update via an email/ phone call needs to be given to the dev team, so they can process it further.

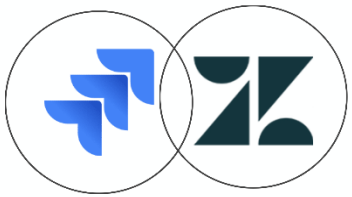
If bugs are discovered, then the QA team again manually sends over the status to the respective team members, who then work on the issue in GitHub. Then there is some back and forth until the situation is resolved.

Alright, hold that thought.

Cross-Company - the Case

An investment company (let's call it Alpha), uses Jira internally to manage issues and plan projects. It has outsourced software development to another company (Beta) that uses Jira for project management. It has also outsourced its ticketing system to another company (Theta) that uses Zendesk.

A classic example of multi-sourcing.



To keep track of dev issues handled by Beta, Alpha sends endless emails to them and gathers the relevant information. It then keeps track of all this in its own Jira by creating appropriate issues and updating its statuses.

Yet again it needs visibility of tickets that are raised in Theta's Zendesk, right?

We'll get there in a little bit.

Servicenow on One Side

Are you wondering why I have considered ServiceNow specifically? Well because eBonding for a long time has been associated with it. It has become a popular solution in the form of an eBonding spoke (we will cover this shortly) such that when you search for it, ServiceNow appears along.

We are also going to consider another tool called Exalate (we will also see the why part of it) in the same section.

For now, let's consider a case when ServiceNow exists at one side of our eBonding.

ServiceNow Jira - the Case

So a company uses ServiceNow to track incidents customers raise. Imagine a certain incident has presented itself. The support agents analyze it and then provide a workaround, maybe after spending a few hours searching for its solution.

On a tight SLA schedule, they come to know that it needs technical expertise.

So they manually forward (yes, yet again!) it to their service provider (basically the dev team) using Jira. The team then takes up the issue and starts working on it.

It is worth mentioning here that all the time, the issue is worked on in Jira, the support agents have no visibility on its current status and are clueless about what needs to be communicated to the customer. Remember, if all this goes even a tad bit wrong, the overall customer experience is going to get hampered, not to mention the effect it is going to have on the SLA.

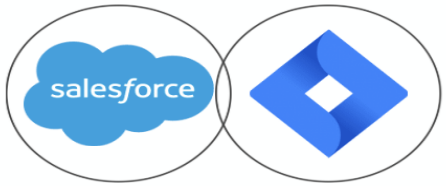
Integrating Multiple Platforms

We just talked about ServiceNow being on one side of an eBonding integration. But it is not uncommon for companies to use different work management systems like Jira, ServiceNow, Azure DevOps, Salesforce, GitHub, Zendesk, and the like.

So of course, if teams in your organization are using them internally or your eBonding partner uses any of them, you would want integration between them. For instance, your team uses Salesforce as their CRM and you want to eBond it with Jira, Zendesk, GitHub, ServiceNow, or Azure DevOps.

Let's look at an example.

Jira-Salesforce - the Case



Modern teams are digital, global, and not always co-located. So what do they do when they need information and expertise from other teams?

For instance, your sales team using Salesforce will definitely benefit from exchanging information with the development team using Jira. Cases in Salesforce have valuable customer feedback, queries, and questions. If such information is percolated to Jira for the development team to work on, the overall customer experience will just get better.

Both teams will then always have the most up-to-date and consistent information in Jira and Salesforce. But again how does it get passed? Manually? Er...

Have you observed a pattern here?

Yeah right, the information is being passed manually in all the cases we have seen above. This has created manual data entry errors and has increased friction amongst teams.

eBonding here can help business-critical information get synchronized in both directions in the respective applications automatically and in real-time. This will help automate processes end-to-end, help drive transparency, and keep everyone on the same page.

But it isn't a one-off effort.

Challenges of eBonding:

There are a lot of things that can go wrong.

This is because every organization and its applications follow different processes, have different schemas, naming conventions, different ways of handling tickets or issues, and so on.

For instance, Jira issues have “Summary” while there is “Short-description” in ServiceNow. **Priorities** in one application can be **Severity** in the other.

So finding a common ground to agree upon information exchange is a task in itself. This can sound easy for 1 or 2 eBonded integrations but becomes a gigantic mess when implemented for growing integrations.

Since processes are different, the way people handle tasks also differs. For instance, when an incident is put on hold, is resolved, or is automatically closed after the resolution is going to be different for every team, or in some applications, you cannot add any new information to closed tickets. Handling such scenarios means outlining detailed steps to accommodate these differences.

Typical integration errors are hard to find and detect. If not worked early on, it’s possible they get lost in the general error log (containing every error or warning there ever was) or go completely undetected till they become a blocker.

So take a deep breath, spend some time, and chalk out:

Why do you need eBonding in the first place? Identify the current problems or challenges you are facing in your interactions with other businesses and applications. How are these problems affecting SLAs or the overall customer experience?	How is information being exchanged, absorbed, and processed in different applications? What is the visibility of information for different applications or processes?
How is consolidated information (this includes direct and indirect information, for instance, issue along with comments, attachments, configuration items, etc.) viewed by the eBonding parties? Do they have access to this shared cross-application referential information when they need it?	How are the processes defined on both sides? Who is authorized to change or update the issue status or when can tickets be modified? Count in who takes ownership of these things to avoid friction later on.
What integration scenarios, use cases or processes do you wish to automate?	Should information exchange be synchronous or asynchronous?
How will eBonding errors be handled? Will there be a detailed error log that can be referred to?	How are the network infrastructure/ deployment models of different applications you want to eBond?

These are just a sample set of driving questions, but you get my point.

Get Started with eBonding

Strategize, plan, prepare a team of developers, testers, project managers, software architects, etc, build the solution and roll it out.

While it sounds easy at the outset, building an eBonding solution in-house is not always the best way out.

The amount of time, effort, and money you put into building a solution is not always proportionate to future business requirements. Such an eBonded system will work initially, but over time, it becomes a nightmare.

- Maintenance costs
- difficulty to scale
- the rigidity of an in-house solution

These only add to the woes of the team members who need to spend endless hours on it to rebuild and repurpose. And every time you want to sync with a new customer or supplier, you have to go through the process of setting up a new integration all over again. What a waste of money and resources.

Also, many times companies push their developers off their limits and bring the eBonding integration out as soon as possible. This approach doesn't give them the breathing time to think about the strategy, the technology, the systems they are going to use, etc, leading to an inadequate, clumsy, and ineffective solution.

Thankfully tech geeks have come up with commercial off-the-shelf eBonding solutions. And they come in a variety of different flavors: deployment models, pricing, number of supported integrations, code or no-code configurations, and architectures.

You also get to experience out-of-the-box integrations with such tools. It's way easier to set up customizations through them because after all, they are the experts. Not to forget, you don't need technical people all the time to run your integrations for you. Business users can make it happen with equal ease.

After a while, you will realize that an eBonding tool is surely going to save a lot of your man-hours and money and will get you a faster time to value.

Since they are readily available and are gaining momentum, it's time we have a look at the endless possibilities such tools offer.

But before we actually see how they are implemented, let's first understand what you need to look out for in such solutions.

Here's what.

What to Consider When Choosing an eBonding Tool

Decentralized integration (Autonomy)

What would your reaction be, if the following notification pops up: "Please contact your integration partner if you want to make changes to your sync", or "Please verify if changes in your sync are saved in the central hub"? Frankly, I would be frustrated and I am sure you would too.

Well, having a solution that supports decentralized integration means each party has complete control (or autonomous control) over what is sent and received, without having to bother the other side. Such control can be achieved through a distributed architecture rather than a centralized one.

With such a distributed architecture, you don't need to configure changes to a central hub every time your integration requirements change, but independently control your information flow at either end.

Such a set-up also ensures that your systems are loosely coupled. So let's touch base with this point next.

Loosely-Coupled Systems

The distributed architecture as discussed above points towards an often overlooked aspect of an integration tool, i.e. it should allow the systems to be loosely coupled to keep both eBonding parties less dependent on each other.

However, being less dependent in no way affects the quality of the integration. In fact, it means that changes on one side do not affect the other.

In essence, this can force you to think that it might create data synchronization issues. Because it is all the more difficult to maintain the correct synchronization sequence and apply it locally if the systems are loosely coupled and there is no central entity into which the changes are easier to track and apply.

To avoid this, additional coordination mechanisms like transactional sync queues must be in place. Only then can it be ensured that all data synced is applied in the correct order in which it is initiated.

So considering these points, if you choose a tool that supports such mechanisms, then in the long run it can make the integration even more manageable and scalable.

Security

Drawing an inference from the point above, since each side controls what it sends and receives, you no longer have to be skeptical about your information being accessed by someone who shouldn't, you just don't send it.

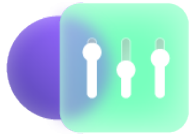
Automatically, only authorized people will be able to access and process information passed between applications.

Adopting proper security measures is also an important aspect of an eBonding tool. Companies are always privy to sharing information with others, especially if it's outside their borders (cross-company). So token-based authentications, encrypted file transfers, VPN connections (if necessary), and secure transfer protocols like HTTPS are what you should be looking for.

Flexibility

Whoever said “Flexibility is the key to stability” clearly did not exaggerate (if you’re wondering, it was John Wooden, the American basketball coach).

Businesses today revolve around flexibility, especially for IT integrations (aka eBonding).



Because they are already comfortable with their own ecosystem of applications and don’t want to make the switch, they are on the lookout for ways to integrate and collaborate with other software applications without leaving their own.

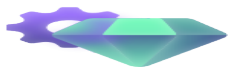
There is also a realization that change is a part of IT, so are changing eBonding integration requirements. Information passed today can become obsolete tomorrow, or you might want to share something completely new the day after.

You might have a simple eBonding use case or a really complex one. Having your tool rapidly adapt to these changes is important. So keeping flexibility higher up your checklist is the key to a great performing solution.

Reliability

All systems break down at some point, so does your eBonded integration, right? Wrong!

True that downtimes and system failures exist, but that should not tumble your tool down. It should be up and running gracefully within the least amount of time without anyone meddling in between. Such that no one should even notice the outage.



Scalability

At the core, eBonding is integrating different applications of your partners, customers, or suppliers. Meaning it's always going to be more than one. More than one eBonded integration, application, and company.

So it must be scalable to be able to cater to a variety of different audiences and applications: Jira, ServiceNow, Salesforce, Azure DevOps, and the like. The more the merrier! The tool should easily be able to adapt to growing integrations so that a new application or a new partner can be onboarded with minimal effort and tweaking.

So keep this in mind while doing your research.

All these points sound good, now let's see how we can implement them.

How to Implement it?

I consider 2 tools here, IntegrationHub's eBonding spoke and Exalate.

Why these?

Well, because like I already said, the first tool is a popular eBonding solution and the second one (if you haven't already heard of it) is climbing the charts for supporting decentralized integration and loosely-coupled systems and being flexible.

So you can explore both of them. Ideally, after you are at the end of this blog.

IntegrationHub: eBonding Spoke

If you have directly jumped to this section, let me start by telling you a little about ServiceNow, in case you are new to it.

ServiceNow is a leading tool for managing IT services and is popular among its customers.

IntegrationHub is a result of ServiceNow's effort towards IT automation. It has pre-defined integration design patterns that can help synchronize information (uni, bi-directionally) between 2 ServiceNow instances (eBonding spoke) or between a ServiceNow and another instance (for example, Jira, Slack, Remedy, Salesforce, GitHub, etc) using their respective spokes along with the flow-designer.

There are 2 reasons why we are only discussing the eBonding spoke here: first, it doesn't require an IntegrationHub or Orchestration subscription and is available by default in your ServiceNow instance. And second, because this article is all about eBonding, right?

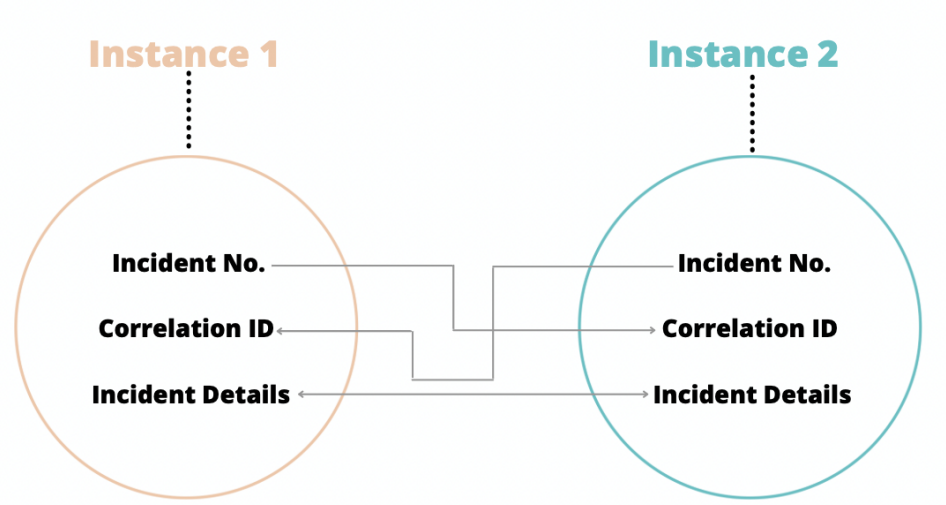
eBonding spoke allows bidirectional integration between 2 ServiceNow instances. That means you can synchronize incidents, problems, change requests, etc with your other ServiceNow instance.

This is very useful when you and your eBonding partner have 2 production environments for ServiceNow. Maybe one at the back-end (source system) and the other at the front-end (the target system). You want to create matching incidents on both these instances, so different teams can handle them as per their roles and responsibilities.

For this, ServiceNow eBonding spoke contains OOB actions like the following:

Create Remote Incident action:

This takes Incident details from the source instance to create one on the target instance. It has a Correlation ID on both systems that have each other's Incident numbers. So the source Incident number is passed to the target instance in the Correlation ID and vice versa.



Lookup Remote Incident action:

This action is used for looking up the details of the remote incident like the short description, summary, priority, etc.

Update Remote Incident action:

This is used to update the incident on the remote instance from the details looked up from the source instance. The Correlation ID we saw above is used for this.

So effectively both the ServiceNow instances will always have matching Incident data. Changes to one instance are reflected in the other using the Actions mentioned above.

As you just saw, the eBonding spoke works well with 2 ServiceNow instances and can be used without much technical expertise. It is also easy to set up and configure.

But if you need integrations between ServiceNow and other non-ServiceNow applications you need to add particular spokes (e.g.: Jira spoke, GitHub spoke, etc). These spokes are provided in IntegrationHub but at an additional price.

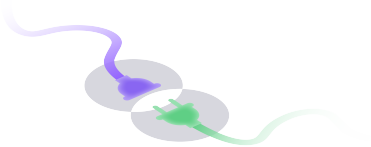
Also, these are 3rd party APIs that have OOB actions included in them. They are pretty extensive but become rigid after a point. So integrations are limited to the Actions supported. If you need to fit in new advanced integration logic, then a request for the updated version to the creator of the spoke needs to be made.

But we are diverting from the topic here. Getting back to eBonding, let's see what Exalate has in store for us.

Exalate

Exalate is a cross-company (or B2B) integration solution that helps organizations and teams to close their gaps. Gaps of having to deal with inconsistent, incoherent, and scattered information spread across applications and teams, and especially across company borders.

It helps streamline collaboration across work management systems like Jira, Azure DevOps, ServiceNow, Salesforce, GitHub, HP ALM, and other trackers.



So for instance, if you want Zendesk-Salesforce, GitHub-ServiceNow, or simply ServiceNow-ServiceNow (just like eBonding spoke) integrations, Exalate can be explored.

It allows information to be synced bi-directionally so business processes are automated end-to-end. Secure information exchange, decentralized integration with the help of a distributed architecture, reliability, and flexibility are some of its primary feature offerings.

Suppose your customer support team is using Zendesk and your development team has Jira as its go-to app.

The tickets raised need to be handed over to the development team (clearly a Jira-Zendesk integration scenario). They start working on it and update the issue status in Jira. Support agents are clueless about the issue the customer has raised and what to inform them, and, yes, panicking!

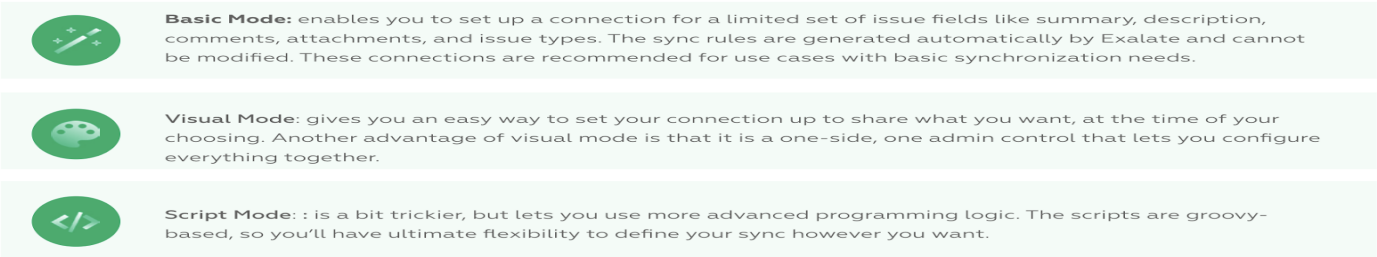
With Exalate, you can help this situation by following the steps below:

Step 1: Install Exalate on both ends of the eBonding applications i.e Jira (cloud or on-premise) and Zendesk.

Step 2: Secure a connection between Jira and Zendesk.

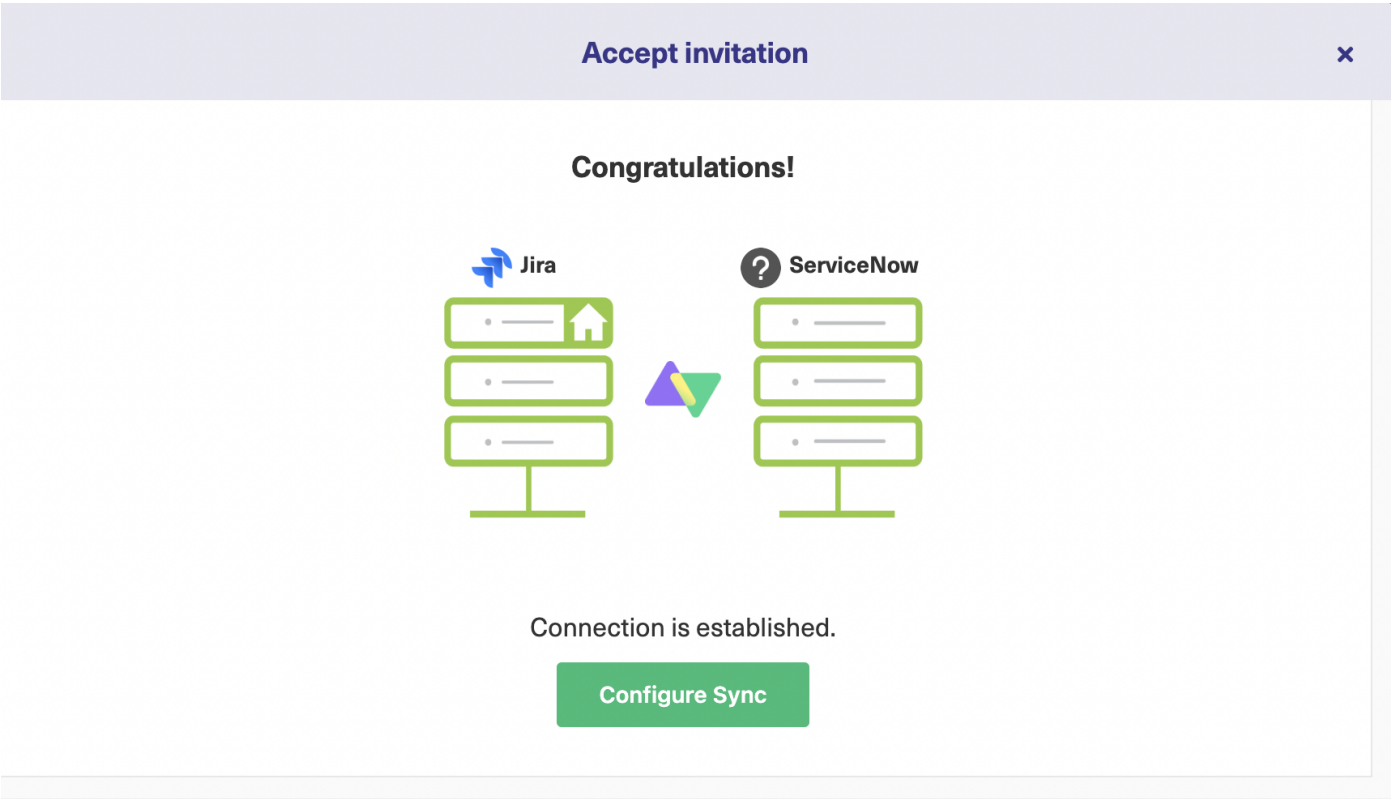
A connection in Exalate defines the synchronization behavior. It is used to uniquely identify each end of the eBonding integration.

While setting it up, you are asked to choose between 3 configuration modes.



If you are one of those tech geeks, then Script mode is the most interesting offering Exalate has in store for you. It is highly customizable and allows you to sync almost any kind of information between software applications.

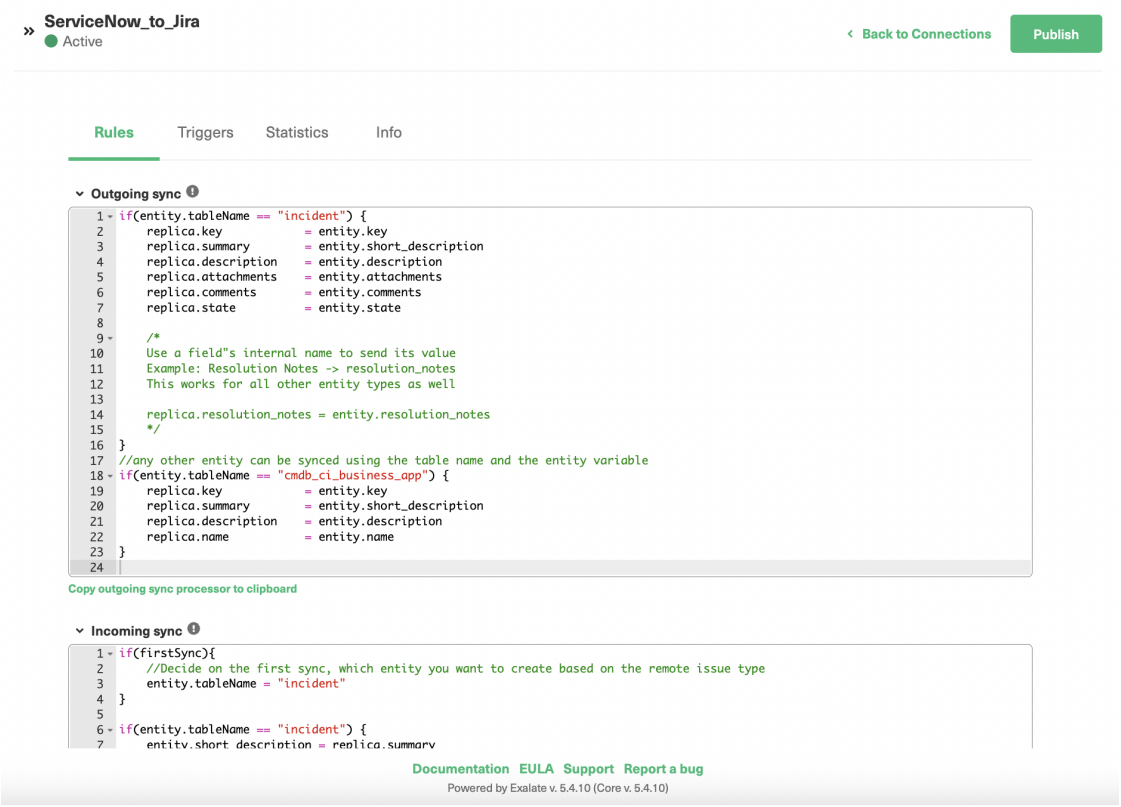
It uses the “Groovy Scripting” language for adding scripts that allow you to control what information is sent and received. Let me elaborate on this a little in the next step.



Step 3: Configure the sync to control how information flows.

There are Outgoing and Incoming sync rules on both sides of the eBonding integration.

So Jira's outgoing sync will decide what information will be passed from Jira to Zendesk i.e development to customer support. And the Incoming sync will decide what information will be received from Zendesk to Jira. The same on the Zendesk side as well. This makes Exalate's integration decentralized, secure, and flexible.



Step 4: Create automatic syncs through triggers.

Once you have decided what to send to and receive from the other side, you can create triggers to start your sync automatically. Triggers are conditions that when satisfied sync according to the sync rules you have set in Step 3.

Step 5: Take a break or have a cup of coffee.

So what's my point?

Phew! That was a lot of information.

Long story short, I'm not comparing the 2 tools in any way, simply because they are not direct competitors. But there are certain facts that you must consider before making a decision.

eBonding spoke works best if you are a company using ServiceNow, and wanting to eBond with other teams using ServiceNow. It allows customization, but that can be limited by ServiceNow's own infrastructure.

For non-ServiceNow eBonds, Exalate is the better choice. It works with multiple ServiceNow instances as well as other work management systems.

Of course, you can integrate ServiceNow and other applications using IntegrationHub, but that would be less intuitive. You can have a look at this comparison between Exalate and IntegrationHub.

So make an informed decision, read this again if you want, and eBond!

Conclusion

This article explained what eBonding is in the first place. We also saw how it all started with the telecommunications industry and then was acknowledged by others. Then we saw a few real-world scenarios.

Moving forward, we saw how implementing an eBonding solution is not always a good option and why you should rather opt for a commercial one. Then we went over a few factors you must consider while choosing such solutions.

Finally, we saw a glimpse of what eBonding spoke (IntegrationHub) and Exalate have to offer.

Recommended Reads:

- ServiceNow eBonding: The Complete Guide
- B2B Integration: The Comprehensive Guide
- The Definitive Guide to Cross-Company Integrations for IT Professionals
- ServiceNow to ServiceNow Integration: The Step-by-Step Guide to Setting up a Two-Way Sync
- Service Integration and Management (SIAM): The Complete Guide
- How to Build an Effective SIAM Operating Model