Jira and Salesforce Integration

Last Modified on 01/31/2025 4:53 am EST

A sales team looking to collaborate with other teams (internally or externally) must find a way to ensure that both sides share data based on pre-approved mappings.

If the sales team uses Salesforce and the developers use Jira Service Management, they can get progress updates and improve the speed of service delivery.

To get both systems to interact, admins would need to install Exalate on both sides to handle the transformations and field mappings.

Exalate supports Jira Service Management, Jira Software, and Jira Work Management.

You can install it on Jira Cloud or Jira On-premise or even set up a private connection in case any of your instances are behind a firewall.

You can also deploy Exalate for Salesforce and Jira on Docker.

Exalate Installation and Configuration Steps

Here is a brief overview of a Jira Salesforce configuration.

1. Install Exalate for Salesforce via AppExchange or from our integrations page.
2. Install the Exalate app on Jira. You can start from the Atlassian marketplace.
3. Choose the connection type: Basic or Script.

To learn how to set up a connection between Jira and Salesforce using the Script mode, go to the Getting Started guide for a detailed visual breakdown.

Just choose the two platforms you want to configure, and you'll get a comprehensive explanation of how to proceed.

Advanced Jira and Salesforce Integration Use Cases (+ using AI Assist)

With Exalate, your team can set up different advanced use cases. The AI-powered scripting engine that is embedded in the Exalate UI makes it easier to generate scripts based on human prompts.

You can also use Script Helpers to reduce the effort of scripting connections from scratch. For a step-by-step breakdown, check out a detailed Jira Salesforce integration guide.

Let's go through some Jira Salesforce integration use cases.

Use Case 1: Map Salesforce Cases to Multiple Jira Issue Types

Whenever the sales team creates a case for a customer on Salesforce, the contents of the case should be mapped to a Jira issue (task, epic, or feature).

To get this to work, enter the script mapping or use AI Assist to generate the code by typing in a detailed prompt describing your use case.

> *"I want to sync the Salesforce case with different issue types in Jira. For example, when a Salesforce case type is a problem it is mapped to Jira issue type Bugs, when the case type is question it is mapped to a Bug in Jira, and when the case type is feature request it is mapped to feature request in Jira."*

Go through the generated output to confirm if it aligns with your expectations. You can continue refining the prompt until you get what you need.

Here is a snippet from the generated code for the Jira incoming side:

```
if (entity.entityType = "Case"){

issue.summary      = replica.summary
issue.description  = replica.description
issue.comments     = commentHelper.mergeComments(issue, replica)
issue.attachments  = attachmentHelper.mergeAttachments(issue, replica)
issue.labels       = replica.labels
}
```

Click Discard if the generated code is incorrect. If the generated script is correct, click Insert Changes. Once you're satisfied with the scripts, click Publish to save and implement changes.

Note: Review the AI Assist prompting guidelines to improve your prompts and, thus, the output.

Exalate also allows the mapping of multiple Salesforce objects to a Jira issue. So, tweak the script and prompt to get accurate results for your particular use case.

Use Case 2: Map Statuses Between Jira and Salesforce

All collaborations can work if both the team using Jira and Salesforce are aligned on goals. So to keep both sides updated, you must sync the status of the issue or case to be updated instantly across both systems.

Your browser does not support HTML5 video.

Here is a sample prompt for this use case in the Jira incoming sync textbox:

> *Map the status of the incoming Salesforce case to match the status of the associated Jira issue so that "New" is mapped to "Open" and "Done" is mapped to "Resolved".*

Here is the code snippet generated by AI Assist.

```
def statusMap = [
   "New"   : "Open",
   "Done"  : "Resolved"
]

def remoteStatusName = replica.status.name
issue.setStatus(statusMap[remoteStatusName] ?: remoteStatusName)
```

Use Case 3: Map Private Comments and Preserve User Mentions

Comments are synced by default. But if you want user mentions to appear in the comments with the user's name, write a script for it.

Here is a sample prompt to provide in the Jira Outgoing sync textbox:

> *"I want to send only private comments from Jira to Salesforce. Also, I want to send user mentions with usernames from Jira to Salesforce"*

Here is the code snippet generated by AI Assist.

```
replica.comments = issue.comments.collect { comment ->
    if (comment.internal) { // Check if the comment is private
        // Replace user mentions with user names
        def userMapping = [
            "[~accountid:a1b2c3]": "User A",
            "[~accountid:d4e5f6]": "User B"
            // Add more mappings as needed
        ]
        userMapping.each { key, value ->
            comment.body = comment.body.replace(key, value
)
        }
    }
    return comment
}
```

If you want to sync comment threads, the same principles apply. It is worth noting that Salesforce supports a “chatter feed” functionality that allows threaded replies to comments. Jira does not have similar functionality.

For threaded replies, the code snippet generated by AI Assist for the Salesforce Outgoing sync should be as such:

```
replica.comments = entity.comments.inject([]) { result, comment ->
    def res = httpClient.get("/services/data/v54.0/query/?q=SELECT+Name+from+User+where+id=%27${comment.author.key}%27"
)
    comment.body = nodeHelper.stripHtml(res.records.Name[0] + " commented: " + comment.body)
    result += comment

    def feedResponse = httpClient.getResponse("/services/data/v54.0/chatter/feed-elements/${comment.idStr}")
    def js = new groovy.json.JsonSlurper()
    def feedJson = groovy.json.JsonOutput.toJson(feedResponse.body)
    feedResponse.body.capabilities.comments.page.items.collect {
        res = httpClient.get("/services/data/v54.0/query/?q=SELECT+Name+from+User+where+id=%27${it.user.id}%27")
        def c = new com.exalate.basic.domain.hubobject.v1.BasicHubComment()
        c.body = res.records.Name[0] + " commented: " + it.body.text
        c.id = it.id
        result += c
        }
    result
    }
}
```

This code snippet fetches the comment as well as the corresponding response from the httpClient to replicate the feeling of a thread in Jira.

Use Case 4: Update a Salesforce Account from a Jira Custom Field

Let’s say you make some of the content from a default field in a Salesforce Case appear in a custom field in the related Jira Cloud issue. This will help you update account information automatically with the correct user on both sides.

Your browser does not support HTML5 video.

The AI prompt for scripting this connection could look like this on the Jira Outgoing side:

> *“Map the value of the Salesforce field named ‘Account Name’ to a Jira custom field named “Salesforce Account’.”*

The generated results might contain this snippet:

```
replica.accountName = issue.customFields.'Salesforce Account'.value
```

The code for the incoming sync on the Salesforce side would look as follows:

```
if(!firstSync){
  def response =   httpClient.get("/services/data/v54.0/query/?q=SELECT+id+from+Account+where+name=%27${replica.customFields.'SF Account'.value}%27"
)


if(response){
  def accId = response.records.Id[0]
httpClient.patch("/services/data/v54.0/sobjects/Case/${entity.Id}",
"""{"AccountId" : "${accId}"}""")}}
}
```

You can discard, accept, or refine the prompt to nail down the specifics of your use case.

AI Assist, like any other AI, can make mistakes. So, try to be as precise and detailed as possible with your prompts.

Note: The code snippet might not work precisely as intended due to changes to the environment or other reasons. If you encounter any problems, reach out to us for clarification.

Automate Jira Salesforce Integration Using Triggers

Exalate uses platform-specific triggers to automate syncs based on the specified conditions.

Users can set trigger conditions on the Jira side using Jira Query Language (JQL).

```
project = BLUE AND labels = maintenanc
e
```

Any project with the name “BLUE” and the label “maintenance” will be synced automatically.

You can specify the trigger conditions and filter queries on the Salesforce side using Salesforce Object Query Language (SOQL). You can also use a visual interface to create queries in Salesforce instead of SOQL.

```
StageName= 'Prospecting'
```

This search query defines the Opportunity stage as 'Prospecting'. This will give the devs on the Jira side a better understanding of how to prioritize and address the issue.

Supported Jira and Salesforce Entities

Exalate works well for Jira and Salesforce integration because it supports the synchronization of several objects and entities between both systems. Check out the comprehensive list of supported Salesforce entities.

Check out the comprehensive list of supported Jira Cloud and On-premise entities.

This is a sample mapping between Salesforce cases and Jira issues:

Salesforce case ↔ Jira issue

- subject ↔ summary
- priority ↔ priority
- status ↔ status
- case owner ↔ assignee
- internal comments ↔ comments
- attachments ↔ attachments
- case origin ↔ labels
- custom fields ↔ custom fields

- any field available via REST APIs

Video Tutorials

- Watch the configuration tutorial for Jira and Salesforce.
- Watch the installation and integration tutorial videos for all connectors.

Other resources

- Download the Jira Salesforce integration eBook.
- For any help or support for your Jira Salesforce integration use case, reach out to our integration engineer.
- Talk to Aida, your AI-powered integration sidekick, and get answers to your questions faster.
- Check out the detailed security and architecture whitepaper.
- Visit the Exalate Academy to get access to learning materials.
- Subscribe to Exalate Hack to get email updates and expert tips about the product.