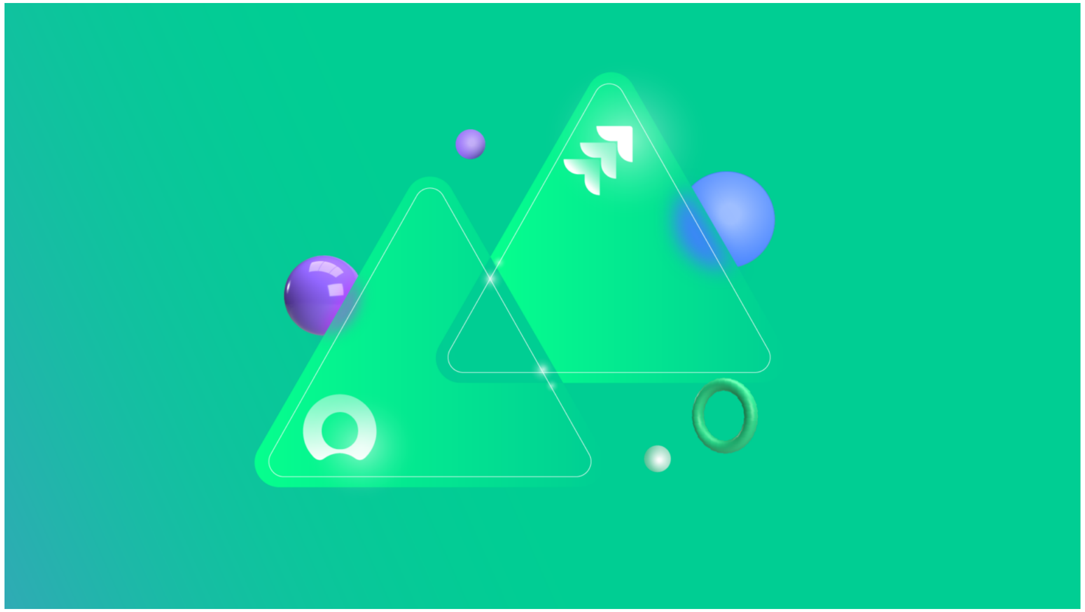
exalate

# How to Sync Jira Issue Fields as Work Notes in ServiceNow



## Table of contents

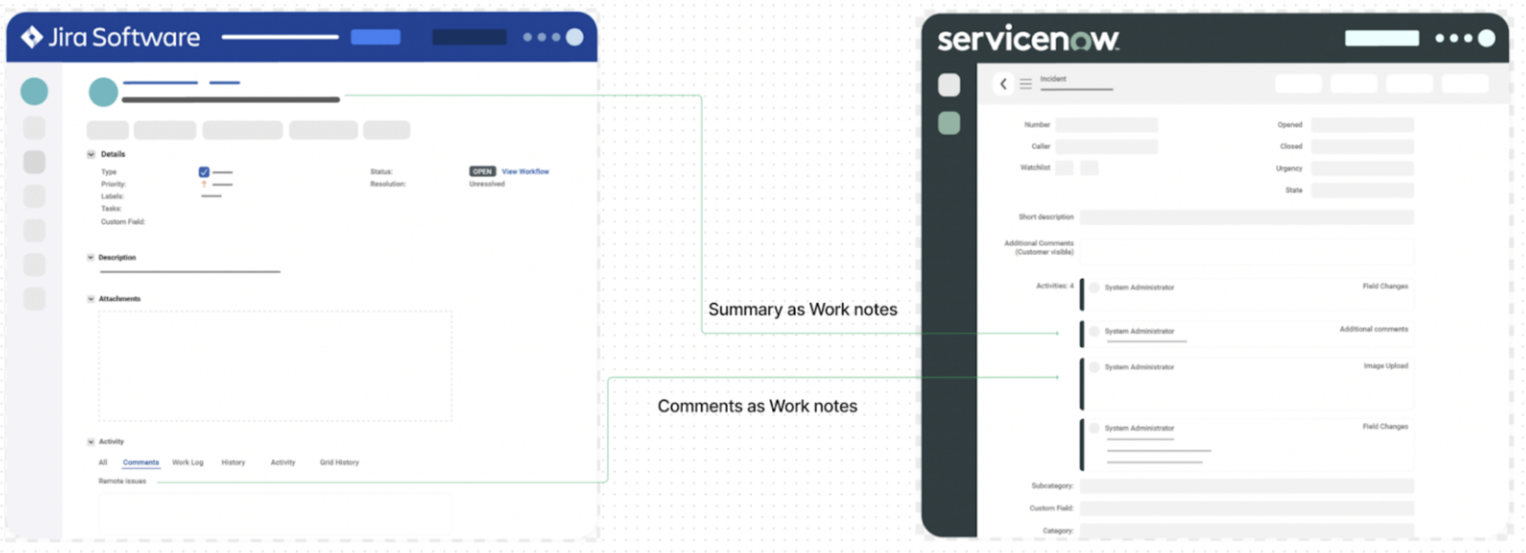
*This article was originally published in the Atlassian Community and the ServiceNow Community.*

We get a lot of customer requests to integrate Jira and ServiceNow at Exalate. And we come across a lot of interesting use cases in the process. Here is an advanced one.

## The Use Case

A Jira issue is synced with a ServiceNow incident.

2 advanced requirements are:



## First Requirement

When a comment is added to a synced issue in Jira, then it must appear as a Work note (or an internal note) on the ServiceNow side.

## Second Requirement:

If any change is made to a particular field on the Jira issue, that change must be logged as a private Work note on the ServiceNow side.

***Note:*** *We are syncing 'Summary' from Jira as a Work Note in ServiceNow for this example. But it can very well be any other field.*

## Challenges

Syncing summary and description in Jira as a summary and short description in ServiceNow is simple and can be implemented out-of-the-box.

But the real challenge lies in:

- The summary must be reflected as a short description on the ServiceNow side, the first time that the issue is synced over. Now, if any change is made to this summary, then it must be reflected as a work note in ServiceNow instead of updating the short description with the new summary.
- Syncing comments from Jira as internal work notes on ServiceNow also requires a little tweaking.

# The 3rd-Party Integration Solution: Exalate

Exalate works as a bidirectional integration solution for Jira, Zendesk, ServiceNow, Azure DevOps, Salesforce, and the like. We can implement the above use case using the Exalate API. It is made possible with an in-built Groovy scripting engine that allows you to implement complex use cases.

So you can configure incoming and outgoing information independently via processors present on both sides of the integration.

## Implementation with Exalate

Start by installing Exalate on both Jira and ServiceNow. Then a connection must be established between them.

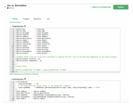
Exalate has 2 modes that serve different functions.

The Basic mode is for syncing only the issue type, description, summary, attachments, and comments. It cannot be configured and comes with a Free Plan.

The Script mode is where most of the complex processing happens. This mode consists of Groovy-based scripts called Sync rules that control information flow at both ends independently.

For this use case, we create a Script mode connection.

At the Jira end, the outgoing sync decides what information will go out from Jira to ServiceNow and the incoming sync decides what information will be received from ServiceNow. This is the same on the ServiceNow side, only the syncs are reversed.



We can modify these syncs under the “Rules” tab.

***Note:*** *The “Triggers” tab is used to automatically start syncing information when certain conditions are met.*

The out-of-the-box configuration on the Jira side remains unchanged since there is no additional tweaking required there.

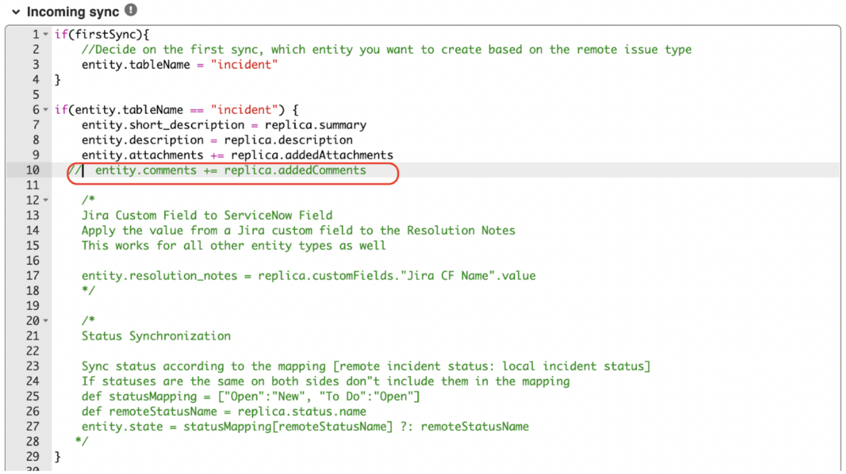
The changes made from the Jira end need to be reflected differently on the ServiceNow side, so we need to modify its “Incoming sync”.

Let us see how these requirements will be met one by one.

## Implementing the First Requirement

Here, the line that deals with comments on the ServiceNow side is: *entity. comments += replica.addedComments*

This line simply syncs the comments from Jira to ServiceNow as it is. We want to modify this behavior. So we comment it out.

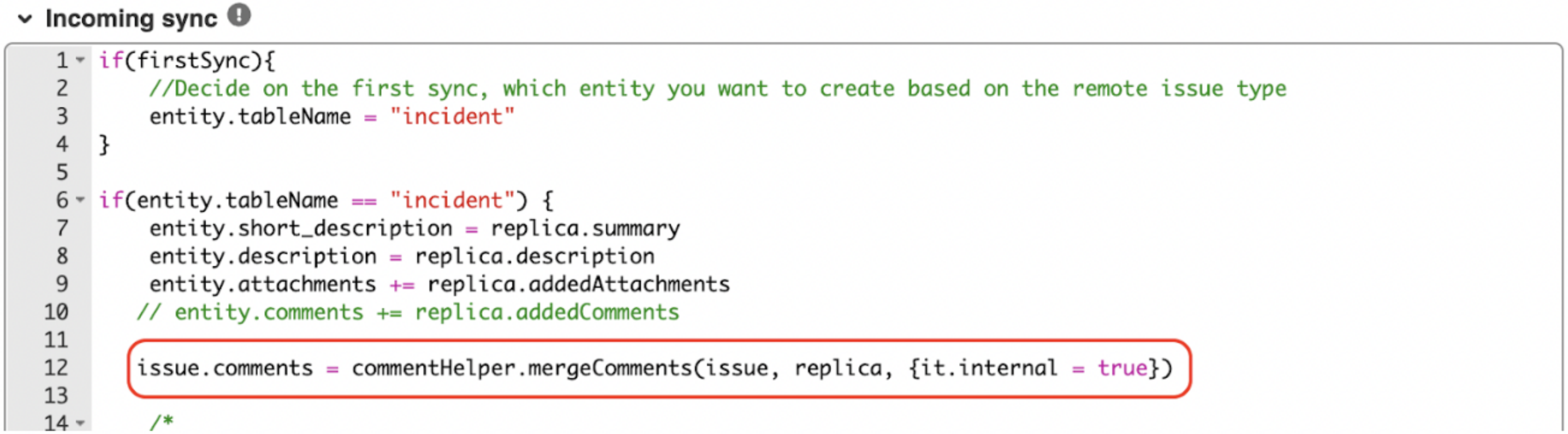


Incoming sync

```
if(firstSync){
    //Decide on the first sync, which entity you want to create based on the remote issue type
    entity.tableName = "incident"
}

if(entity.tableName == "incident") {
    entity.short_description = replica.summary
    entity.description = replica.description
    entity.attachments += replica.addedAttachments
  //  entity.comments += replica.addedComments

    /*
    Jira Custom Field to ServiceNow Field
    Apply the value from a Jira custom field to the Resolution Notes
    This works for all other entity types as well

    entity.resolution_notes = replica.customFields."Jira CF Name".value
    */

    /*
    Status Synchronization

    Sync status according to the mapping [remote incident status: local incident status]
    If statuses are the same on both sides don"t include them in the mapping
    def statusMapping = ["Open":"New", "To Do":"Open"]
    def remoteStatusName = replica.status.name
    entity.state = statusMapping[remoteStatusName] ?: remoteStatusName
   */
}
```

And add this line.

*issue.comments = commentHelper.mergeComments(issue, replica, {it.internal = true})*



```
if(firstSync){
    //Decide on the first sync, which entity you want to create based on the remote issue type
    entity.tableName = "incident"
}

if(entity.tableName == "incident") {
    entity.short_description = replica.summary
    entity.description = replica.description
    entity.attachments += replica.addedAttachments
   // entity.comments += replica.addedComments

   issue.comments = commentHelper.mergeComments(issue, replica, {it.internal = true})

    /*
```

We use the *mergeComments* method of the *commentHelper* class that Exalate provides to merge the comments on the ServiceNow side. The thing that does the magic is *it.internal=true* at the end. This allows the comments from the Jira side to be treated as internal comments on the ServiceNow side.

### Implementing the Second Requirement

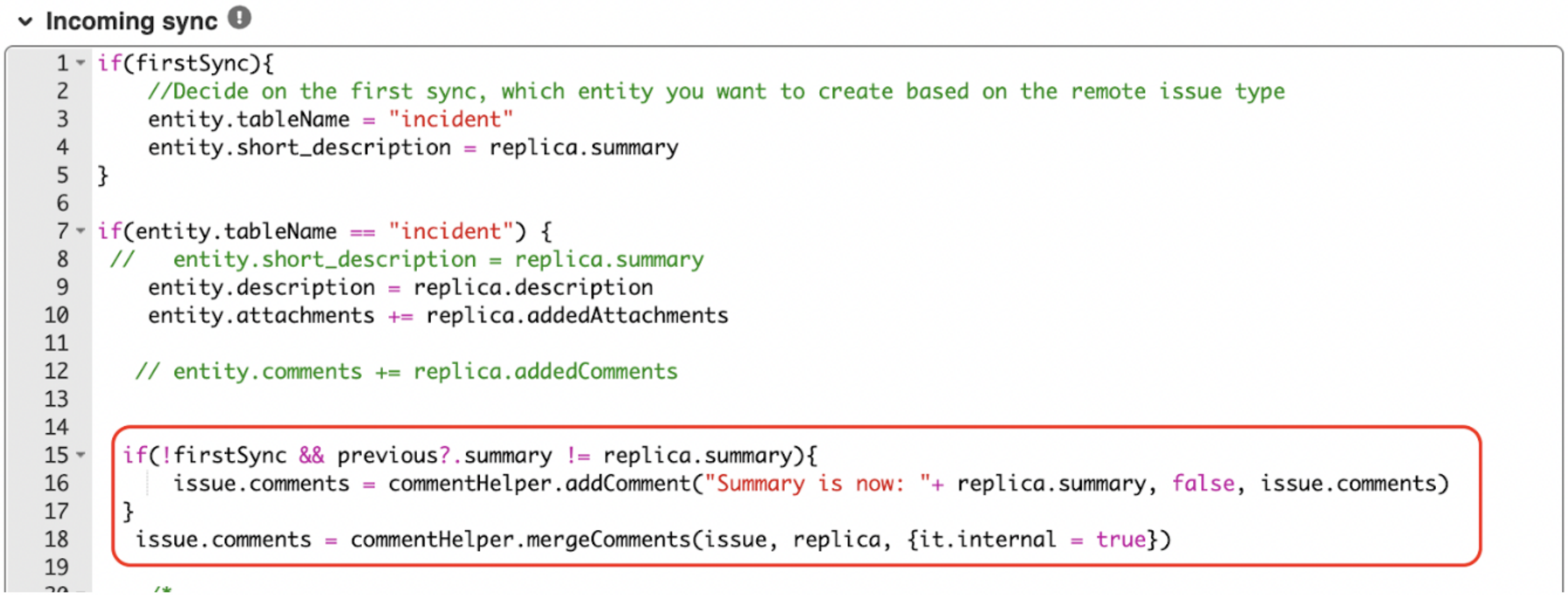
The second part is slightly different than just adding a line of code. Here, during the *firstSync* (the first if condition in the above code block) we need the summary of the issue to be reflected under a short description in ServiceNow.

So we put *entity.short_description = replica.summary* under the firstSync if condition.

Now upon subsequent syncs, we do not want the short description to be updated. So we remove that line.

To populate the work notes from the Jira summary there on, we add this code:

```
if (!firstSync && previous?.summary != replica.summary){
   isssue.comments = commentHelper.addComment("Summary is now: " + replica.summary, false, issue.comments)
}
```

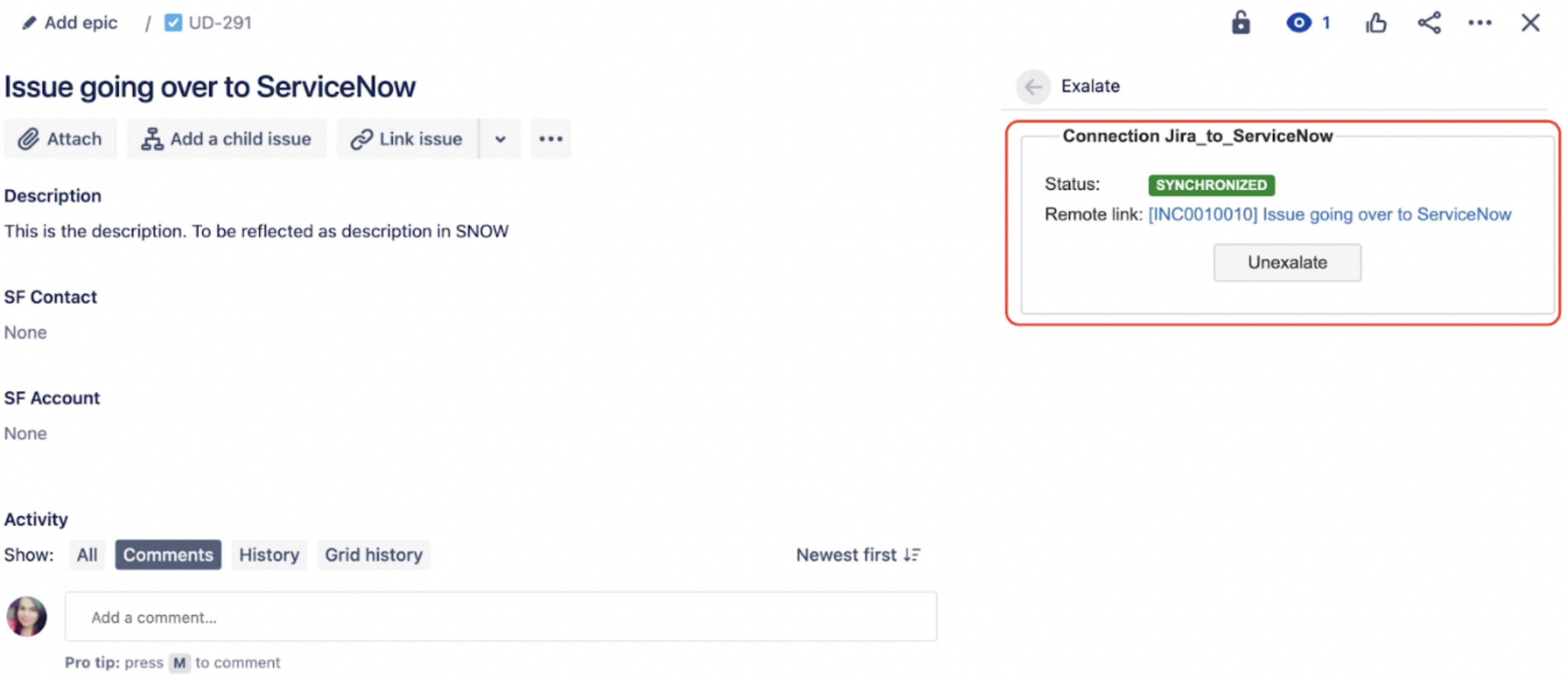


```
if(firstSync){
    //Decide on the first sync, which entity you want to create based on the remote issue type
    entity.tableName = "incident"
    entity.short_description = replica.summary
}

if(entity.tableName == "incident") {
 //   entity.short_description = replica.summary
    entity.description = replica.description
    entity.attachments += replica.addedAttachments

   // entity.comments += replica.addedComments


 if(!firstSync && previous?.summary != replica.summary){
     issue.comments = commentHelper.addComment("Summary is now: "+ replica.summary, false, issue.comments)
 }
  issue.comments = commentHelper.mergeComments(issue, replica, {it.internal = true})

```

The main object of interest here is “previous”. It allows us to capture the previous values of the sync. So if the previous summary is different from the current one coming in from Jira then add the necessary text you need to be followed by the actual new summary.

## Output

We start by creating an issue in Jira and syncing it over to ServiceNow.



Issue going over to ServiceNow

Description

This is the description. To be reflected as description in SNOW

SF Contact

None

SF Account

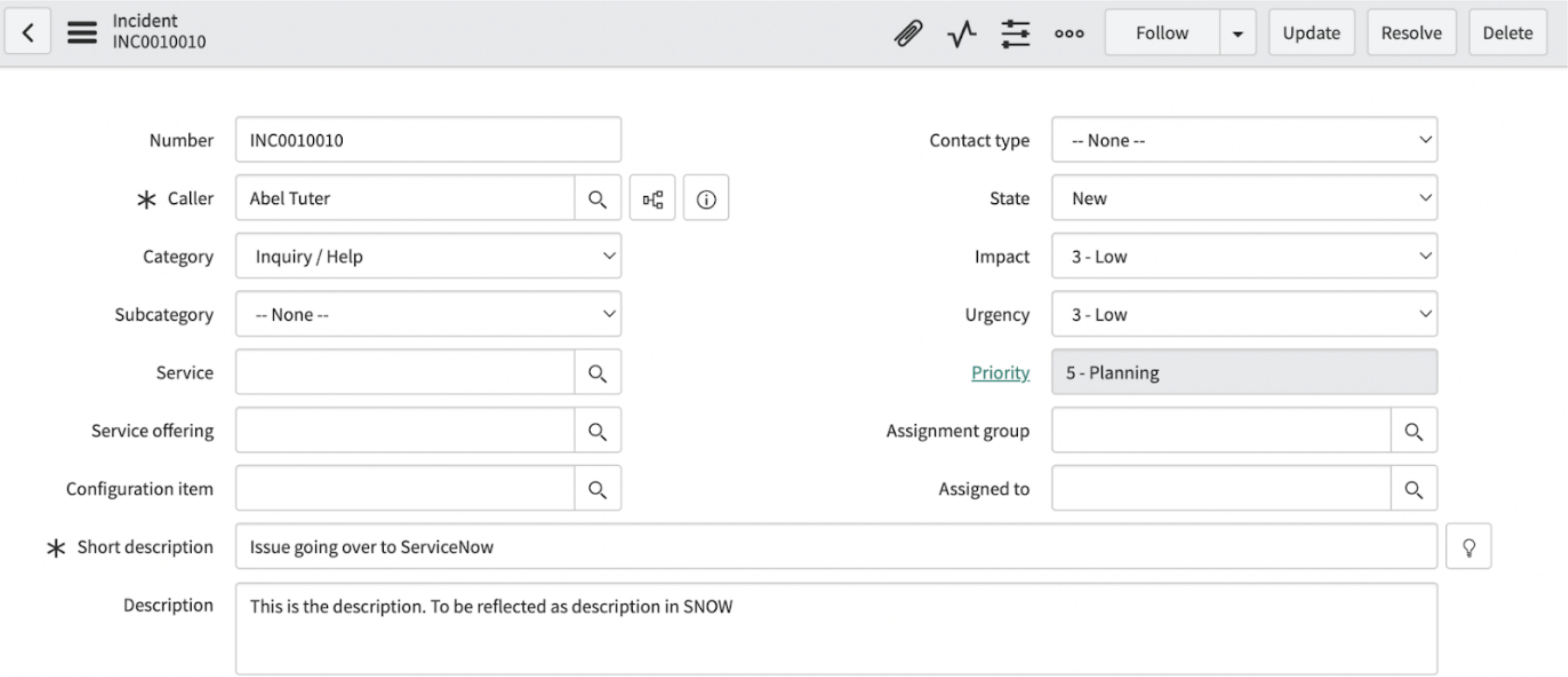
None

Connection Jira_to_ServiceNow

Status: SYNCHRONIZED

Remote link: [INC0010010] Issue going over to ServiceNow

The summary, short description, and other required fields have been synced.



| | | | |
|---|---|---|---|
| Number | INC0010010 | Contact type | -- None -- |
| * Caller | Abel Tuter | State | New |
| Category | Inquiry / Help | Impact | 3 - Low |
| Subcategory | -- None -- | Urgency | 3 - Low |
| Service | | Priority | 5 - Planning |
| Service offering | | Assignment group | |
| Configuration item | | Assigned to | |
| * Short description | Issue going over to ServiceNow | | |
| Description | This is the description. To be reflected as description in SNOW | | |

Leave a comment in Jira and see the comment reflected on the other side as a Work note.

Also, change the summary on the Jira issue and see it reflected on the ServiceNow side as a Work note.

Watch this video if you prefer so, or have a look at the detailed step-by-step approach to a Jira ServiceNow integration.

## Conclusion

There is a lot you can already plan to do using Exalate. Its advanced yet intuitive scripting engine allows you to accommodate a variety of use cases. The helper classes it provides, with a bulk load of methods, make your task a lot easier.

So give it a try if you still haven't, or book a demo with our experts to experience a smooth integration.