exalate

How to Sync Comment Threads and User Mentions in Comments between Jira and Salesforce

Table of contents

This article was originally published on the Atlassian Community.

A Jira Salesforce integration use case can be an interesting one, an amalgamation of business and technical teams.

In this article, we'll discuss an advanced comment use case that allows syncing threads of comments and user mentions between these 2 platforms so that they speak the same language.

The Use Case

The use case is implemented between a Jira Cloud and a Salesforce instance.

The following are the key requirements:

- An issue (or a ticket) created in Jira is synced over to the Salesforce instance as a Case. It might as well be any other Salesforce object.
- Basic fields like summary, description, and comments are synced between Jira Cloud and Salesforce.
- Threaded replies to comments (chatter feed capability) from Salesforce are synced to Jira. Comments from Jira are reflected in Salesforce.
- User mentions in Jira tags the correct corresponding user in Salesforce (if the user exists on both systems).
- Comment formatting is maintained.

The Challenges

- There is a transformation challenge between the 2 instances. Since Salesforce uses HTML internally to represent comments and Jira uses Wiki Markup, there are inherent differences in formatting between these 2 platforms. These must be addressed

correctly.
- There is a fully supported “chatter feed” functionality in Salesforce that allows threaded replies to comments. Jira does not have similar functionality. The challenge then is to reflect these replies back in Jira.

The Solution: Exalate

Exalate is a one-way or two-way synchronization solution that supports multiple platforms like Jira, Salesforce, Azure DevOps, GitHub, Zendesk, etc.

Its intuitive Groovy-based scripting engine allows you to implement advanced use cases. The Sync rules can be modified to set up deeper integrations. In addition to this, you can use Triggers and set up advanced automatic synchronization too.

***Note**: The Exalate Academy is a great way to learn more about Exalate. Give it a try!*

How to Implement Advanced Comment Sync Using Exalate

Prerequisites

- You need to install Exalate on both Jira Cloud and Salesforce instances.
- You should create in Script Mode a connection between the platforms.

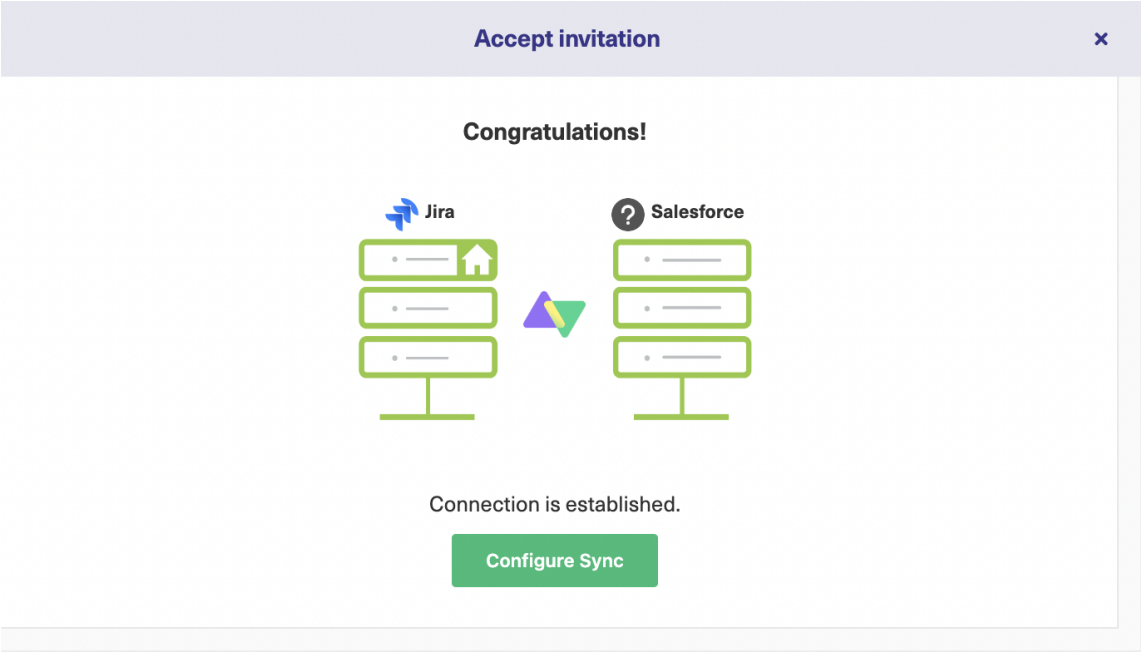
***Note**: You can learn more about setting up a connection between Jira and Salesforce through the Getting Started guide or read this complete Jira Salesforce Integration guide.*

The Implementation Using Exalate

Once the Connection has been created, you need to configure the Sync Rules.

These rules are Groovy-based scripts that control what information to send and receive between the 2 platforms.

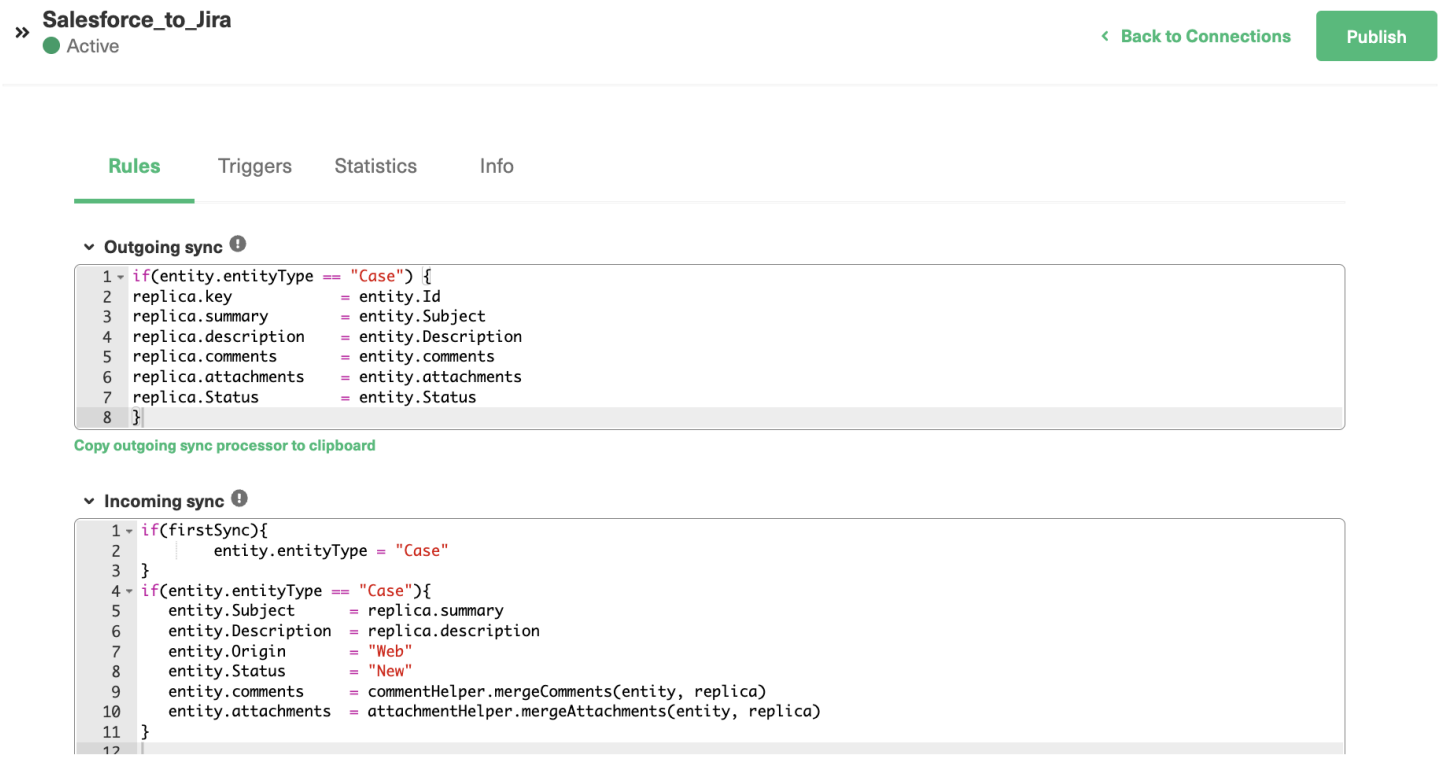
You can click on the “Configure Sync” button after the Connection has been set up to configure these rules or you can also edit the Connection in the “Connections” tab in the Exalate Console.



Rules are present at both ends of the synchronization. The “Outgoing Sync” on the Jira side decides what information must be sent from Jira to Salesforce and the “Incoming Sync” decides what and how information must be received from Salesforce.

The same exists in the Salesforce instance.

The scripts you see are generated by default when the Connection has been created. So common fields like **summary**, **description** , **comments**, etc are already there and can be synchronized out-of-the-box.



Now, we need to edit them to accommodate our sync requirements.

Jira: Outgoing Sync

```
replica.key         = issue.key
replica.type        = issue.type
replica.assignee    = issue.assignee
replica.reporter    = issue.reporter
replica.summary     = issue.summary
replica.description = issue.description
replica.labels      = issue.labels

replica.comments = issue.comments.collect {
    comment ->
    def matcher  = comment.body =~ /\[~accountid:([\w:-]+)\]/
    def newCommentBody = comment.body
    matcher.each {
        target = nodeHelper.getUser(it[1])?.email
        newCommentBody = newCommentBody.replace(it[0],target)
    }
    comment.body = newCommentBody
    comment
}

replica.resolution  = issue.resolution
replica.status      = issue.status
replica.parentId    = issue.parentId
replica.priority    = issue.priority
replica.attachments = issue.attachments
replica.project     = issue.project

//Comment these lines out if you are interested in sending the full list of //versions and components of the
```

```
replica.project.versions = []
replica.project.components = []
```

The Outgoing sync looks like this.

Outgoing sync

```
replica.key             = issue.key
replica.type            = issue.type
replica.assignee        = issue.assignee
replica.reporter        = issue.reporter
replica.summary         = issue.summary
replica.description     = issue.description
replica.labels          = issue.labels

replica.comments = issue.comments.collect {
    comment ->
    def matcher  = comment.body =~ /\[~accountid:([\w:-]+)\]/
    def newCommentBody = comment.body
    matcher.each {
        target = nodeHelper.getUser(it[1])?.email
        newCommentBody = newCommentBody.replace(it[0],target)
    }
    comment.body = newCommentBody
    comment
}

replica.resolution      = issue.resolution
replica.status          = issue.status
replica.parentId        = issue.parentId
replica.priority        = issue.priority
replica.attachments     = issue.attachments
replica.project         = issue.project

//Comment these lines out if you are interested in sending the full list of versions and components of the source project.
replica.project.versions = []
replica.project.components = []
```

Here's what happens in the script:

- The **collect** method iterates over the comments array of the Jira issue and transforms them before assigning them to the replica (to be sent to the other side).
- The **transformation** taking place is at the heart of handling user mentions. The script extracts them in the comment body and replaces them with the email address of that user instead. The replica will now contain the comment not with the Jira-specific mention, but rather with an email address corresponding to the mentioned user.

Note: *Replica works as a payload to pass information between the two applications.*

Salesforce: Outgoing Sync

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

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.autho
    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}%2
        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
    }

}
```

The Outgoing Sync script looks like this.

Here's what happens in the script:

- The inject method iterates over all the comments and performs several operations:
  - For each Salesforce comment, the script first fetches the username of the comment author and appends it to the comment body (so that it can be reflected on Jira as such). In addition, the script employs the stripHtml() method to transform the HTML formatted Salesforce comments into plain text to be properly reflected on the Jira side. We add this main comment to the result variable temporarily.
  - For each Salesforce main comment, the script then fetches associated threaded replies and populates them in the feedResponse. Each of these threaded replies is then sanitized by removing HTML and appending the author's name to the comment body. They are then added to the result variable.
  - This result, once the iterations are over, contains the main comments and threads that have already been transformed. It is then assigned to the replica to be sent over to the Jira side.

Salesforce: 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.attachments  = attachmentHelper.mergeAttachments(entity, replica)

def commentMap = [
   "mathieu.lepoutre@idalko.com" : "0058d000004df3DAAQ",
   "syed.majid.hassan@idalko.com" : "0057Q000006fOOOQA2"
   ]
```

```
def flag = 0
replica.addedComments.collect {
    comment ->
    def matcher  = comment.body =~ /([a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+\.[a-zA-Z0-9_-]+)/
    def newCommentBody = comment.body

    matcher.each {
        newCommentBody = newCommentBody.replace(it[0],"")
        def res = httpClient.post("/services/data/v54.0/chatter/feed-elements", \
        "{\"body\":{\"messageSegments\":[{\"type\":\"Text\", \"text\":\"${newCommentBody} \" },{\"type\":\"Me
        flag = 1
    }
}

if (flag == 0)
    entity.comments  = commentHelper.mergeComments(entity, replica)
}
```

Here's what happens in the script:

- Remember that when sending comments from the Jira side, we replaced the user mentions with the email addresses. So, in Salesforce we create a mapping called commentMap that maps the email addresses to the corresponding Salesforce User IDs.
- The next step is to iterate over the comments contained in the replica and for each comment extract the email address, map it using the commentMap and then replace the email address in the comment with the Salesforce mention of the mapped user.

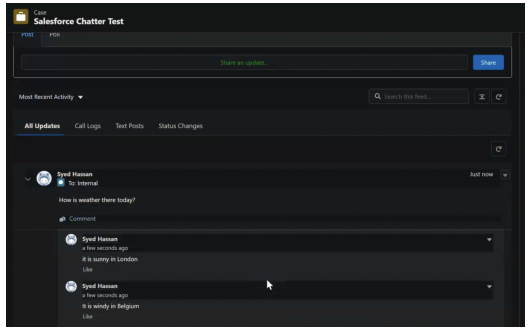
Jira: Incoming Sync

There is no need to modify this since the default behavior is sufficient for our use case.

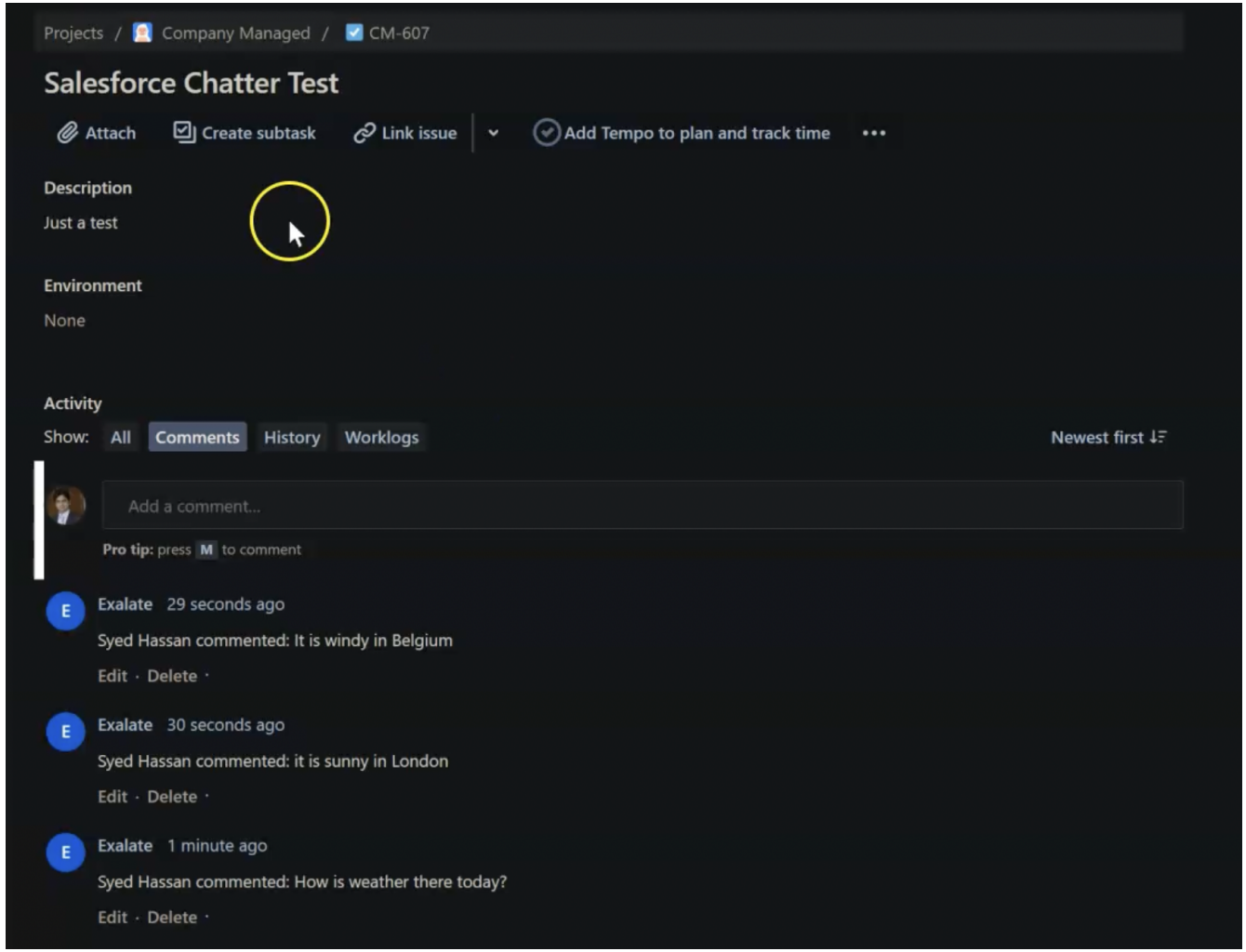
Output

Once the code is inserted into the respective outgoing and incoming syncs, comments, and threaded replies will automatically be reflected in Jira.

So when you insert a threaded comment in Salesforce.

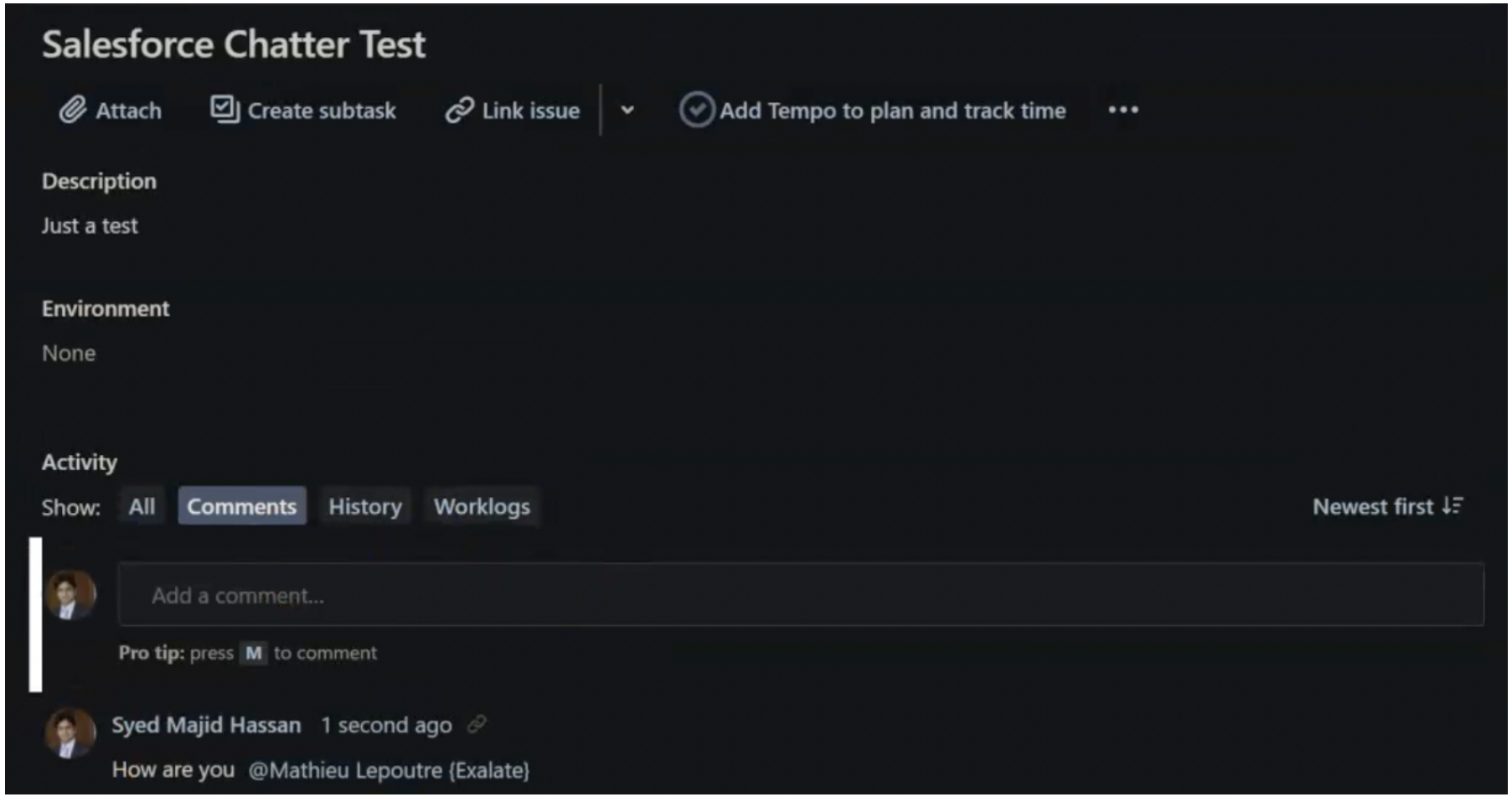


All of them get reflected in Jira.

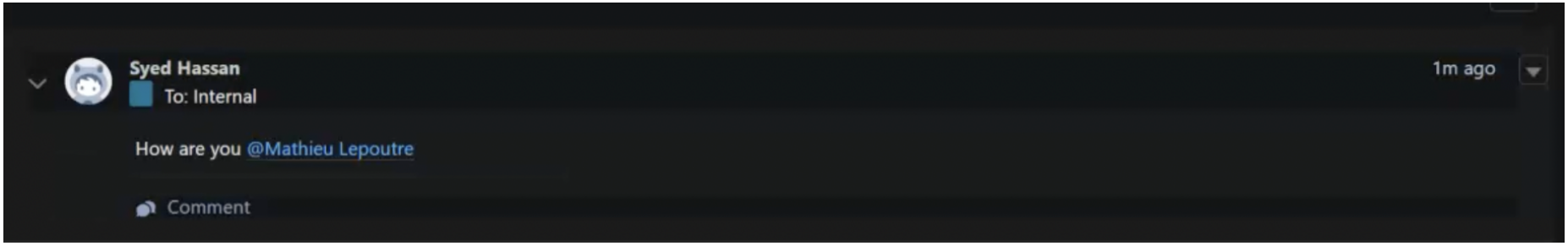


Also, user mentions in Jira will tag the corresponding user in Salesforce, if he/ she exists.

So mention a user in Jira.



And see the corresponding user being tagged in Salesforce.



Conclusion

In this article we just saw an example of an advanced comment sync between Jira and Salesforce. A lot of other advanced sync requirements can also be implemented using Exalate because of its support for Groovy-based scripts.

Book a demo with Exalate to learn more about how it can be customized for your specific scenario.