How to Sync Worklogs in Jira On-premise

Last Modified on 03/27/2024 6:55 am EDT

This page describes how to synchronize worklogs information between two Jira instances.

Introduction

Jira worklogs synchronization requires worklog author lookup. By default, Exalate tries to find the worklog author on the remote side and create a worklog on behalf of that author. The author lookup is performed by email, but you can also configure other ways to find corresponding users on the local system (by full name, username or any other arbitrary logic that you might have). Additionally, you can configure user mapping in the *Incoming sync* processors or set a default worklog author in case the original author is not found.

mergeWorkLogs helper method helps to merge worklogs:

- adds the added remote worklogs
- removes the removed
- changes the changed ones

  mergeWorkLogs does not touch any of the worklogs that exist only locally (were not synchronized yet or are filtered out from the synchronization)

Source Side

Outgoing sync

To send worklogs to the Destination side use the code below

```
replica.workLogs = issue.workLogs
```

Destination Side

Incoming sync

To add/update received worklogs from the source side use the code below. Merges the worklogs from two sides and allows adjusting details of every synced workLog.

```
issue.workLogs = workLogHelper.mergeWorkLogs(issue,replica)
```

Worklog author user mapping

Exalate allows to map worklog authors, but in case the user from mapping settings is not found you can do the following:

- set the original worklog author if the originally mapped user is not found on the local side

```
//---------------------------
// IF THE MAPPED USER IS NOT FOUND, USE THE ORIGINAL(SOURCE SIDE) AUTHOR
//---------------------------
// SETTINGS
def userMapping = [
      "fooey@mycompany.com" : "barry@mycompany.net"
]
// END: SETTINGS
issue.workLogs = workLogHelper.mergeWorkLogs(issue,replica, { w ->
   w.author = nodeHelper.getUserByEmail(userMapping[w.author?.email]) ?: w.author
   w.updateAuthor = nodeHelper.getUserByEmail(userMapping[w.updateAuthor?.email]) ?: w.updateAuthor
})
```

- set a default worklog author if the originally mapped user is not found on the local side

```
//-----------------------
// IF THE MAPPED USER IS NOT FOUND, USE THE DEFAULT WORKLOG AUTHOR
//-----------------------
// SETTINGS
def userMapping = [
      "fooey@mycompany.com" : "barry@mycompany.net"
]
def defaultUser = nodeHelper.getUserByEmail("bazzy@mycompany.net")
// END: SETTINGS
issue.workLogs = workLogHelper.mergeWorkLogs(issue,replica, { w ->
   w.author = nodeHelper.getUserByEmail(userMapping[w.author?.email]) ?: defaultUser
   w.updateAuthor = nodeHelper.getUserByEmail(userMapping[w.updateAuthor?.email]) ?: defaultUser
})
```

Syncing Worklogs between Jira Cloud and Jira on-premise

According to the Atlassian changes in user privacy policy, Jira Cloud user may hide the email address visibility in the Atlassian account setting. This affects worklog synchronization since Exalate uses API to search by email. Therefore, worklogs will be synced only in case the user has set the email to be visible.

If you sync from Jira on-premise to Jira Cloud

```
final def userMapping = [
      "admin@admin.com" : "557058:ada77a72-56a9-4a19-9c23-7d66e4813883"
]
issue.workLogs = workLogHelper.mergeWorkLogs(issue,replica, { w ->
   w.author = nodeHelper.getUser(userMapping[w.author?.email]) ?: w.author
   w.updateAuthor = nodeHelper.getUser(userMapping[w.updateAuthor?.email]) ?: w.updateAuthor
})
```

If you sync from Jira Cloud to Jira on-premise

```
final def userMapping = [
      "512358:ada77a72-56a9-4a19-9c23-7c66ee8t3883" : "admin@admin.com"
]
issue.workLogs = workLogHelper.mergeWorkLogs(issue,replica, { w ->
   w.author = nodeHelper.getUserByEmail(userMapping[w.author?.key]) ?: w.author
   w.updateAuthor = nodeHelper.getUserByEmail(userMapping[w.updateAuthor?.key]) ?: w.updateAuthor
})
```

if you sync from Jira Cloud to Jira Cloud

```
issue.workLogs = workLogHelper.mergeWorkLogs(issue,replica, { w ->
 w.author = nodeHelper.getUser(w.author?.key) ?: w.author
 w.updateAuthor = nodeHelper.getUser(w.updateAuthor?.key) ?: w.updateAuthor
})
```