How to sync Group Picker custom field

JIRA SERVER | JIRA CLOUD

This article shows how to synchronize Group Picker Jira custom field.

Jira provides an advanced custom field Group Picker (single group or multiple groups) which helps to choose user group(single or multiple) in a popup picker window.

You can synchronize the field values using Exalate.

When syncing group picker field Exalate checks whether the group with a certain name exists on the remote side.

In case the group with such name does not exist on the receiving side you can do the following:

- set default value
- create a group with the received group name and set the value to the custom field on the receiving side (applicable only for Jira Server)
- don't set any value and skip the sync of this field

Below you can find examples of the group picker filed(single or multiple groups) synchronization for the last option: when the group does not exist on the remote side - don't set any value.

Jira Server

Group Picker(single group) synchronization

Source side

Outgoing sync

```
if (issue.customFields."Group picker custom field name"?.value instanceof Iterable) {
    issue.customFields."Group picker custom field name".value = issue.customFields."Group picker custom field
name".value.find()
}
replica.customFields."Group picker custom field name" = issue.customFields."Group picker custom field name"
```

Destination side

Incoming sync

```
def remoteGroup = replica.customFields."Group picker custom field name"?.value?.name?.asString
if (remoteGroup) {
    def gm = com.atlassian.jira.component.ComponentAccessor.groupManager
    def group = gm.getGroup(remoteGroup)
    if (group) {
        issue.customFields."Group picker custom field name".value = [group]
    }
} else {
    issue.customFields."Group picker custom field name".value = null // don't sync the group picker field if a
local group is not found
}
```

Group picker (multiple groups) synchronization

Source side

Outgoing sync

```
if (issue.customFields."Group picker custom field name"?.value instanceof Iterable) {
    issue.customFields."Group picker custom field name".value = issue.customFields."Group picker custom field
name".value.find()
}
replica.customFields."Group picker custom field name" = issue.customFields."Group picker custom field name"
```

Destination side

Incoming sync

```
def remoteGroups = replica.customFields."Group picker custom field name"?.value?.collect { it.name?.asString }
if (remoteGroups?.any { it != null }) {
    def gm = com.atlassian.jira.component.ComponentAccessor.groupManager
    def groups = remoteGroups
        .collect { g -> gm.getGroup(g) }
        .findAll()

    issue.customFields."Group picker custom field name".value = groups
} else {
    issue.customFields."Group picker custom field name".value = []
}
```

Jira Cloud

Group Picker(single group) synchronization

Source side

Outgoing sync

```
replica.customField."Group picker custom field name" = issue.customField."Group picker custom field name"
```

Destination side

Incoming sync

```
def remoteGroup = replica.customFields."Group picker custom field name"?.value?.name?.asString
if (remoteGroup) {
    def getGroup = { String group ->
        def wc = new JiraClient(httpClient)
        def groupsResultStr = wc.http(
                "GET",
                "/rest/api/3/groupuserpicker",
                ["query":[group]],
                null,
                [:]
        )
        def js = new groovy.json.JsonSlurper()
        def groupsResult = js.parseText(groupsResultStr)
        def groups = groupsResult?.groups?.groups as List
        if (groups == null || groups.empty) {
            return null
        }
        groups.find {it.name == group}
    }
    def group = getGroup(remoteGroup)
    if (group?.name) {
        def restApiGroup = ["name":group.name]
        issue.customFields."Group picker custom field name".value = restApiGroup
    }
} else {
    issue.customFields."Single Group Picker".value = null // don't sync the group picker field if a local group
is not found
}
```

Group picker (multiple groups) synchronization

Source side

Outgoing sync

```
replica.customField."Group picker custom field name" = issue.customField."Group picker custom field name"
```

Destination side

Incoming sync

```
def remoteGroups = replica.customFields."Groups"?.value?.collect{ it?.name?.asString }?.findAll()
if (remoteGroups) {
    def restApiGroups = remoteGroups.collect { String remoteGroup ->
        def getGroup = { String group ->
            def groups = httpClient.get("/rest/api/3/user/properties?accountId=5ce5eec03b428d0dcd7cac19")
            groups.find {it.name == group}
        }
        def group = getGroup(remoteGroup)
        if (group?.name) {
            def restApiGroup = ["name":group.name]
            restApiGroup
        } else {
            null
        }
    }

    issue.customFields."Multiple Group Picker".value = restApiGroups
} else if (issue.customFields."Multiple Group Picker".value != null && (!remoteGroups || remoteGroups.empty)) {
    issue.customFields."Multiple Group Picker".value = null // don't sync the group picker field if a local
group is not found
```