syncBackAfterProcessing

Signature

Helper	syncHelper
Signature	`syncBackAfterProcessing()`
Description	Use this helper method in the **Incoming sync** to update the source issue once the synchronization to the destination side is finished.
Introduced in	4.6.5

Using the Feature

Assume

- you have 2 instances 'Internal' and 'External'.
- You raise an issue on 'Internal', and exalate it to 'External'
- You need to have a custom field on Internal containing the remote key of the issue created on External

What needs to happen is

- Once that the issue is created on External, a message is sent back to Internal containing the key of the created issue
- The syncHelper.syncBackAfterProcessing() triggers this message as it queues a sync event which is handled as if the issue on external has been changed
- The processing of this sync event is identical to any other sync event

What you need to configure:

On the **destination** side (external) in the incoming sync processor

```
// trigger a sync back to the internal instance
if(firstSync){
  syncHelper.syncBackAfterProcessing()
}
```

On the **source** side (internal) in the incoming sync processor

```
// update the custom field 'remote issue key'  with the  key of the newly created issue
issue.customFields."remote issue key".value = replica.key
```

Warning: Method syncHelper.syncBackAfterProcessing() only works for the bidirectional synchronization. Using it for the unidirectional sync might lead to unexpected sync errors.

Warning: Using syncHelper.syncBackAfterProcessing() outside the firstSync if block might cause infinite loops.

Avoiding conflicts

A sync conflict is possible whenever changing information on the source issue immediately after an exalate.

Source Jira	Target Jira
Exalate issue Status is 'open'	
	Create issue Status is 'Open'
Change status to 'In Progress	
	Syncback issue Status is open
Syncback is processed Status is changed back to 'Open'	
	Status is updated to 'In Progress' because of message sent in step 3
Status = Open	Status = 'In Progress'

To avoid this conflict, one can add this code

```
if (!firstSync && previous?.status.name == replica?.status.name) {
    // do not update the status as it might conflict
}
```

Example usages:

`syncBackAfterProcessing` function can be used to ensure that changes are propagated back to the source instance after the synchronization process. The placement of `syncBackAfterProcessing` depends on the flow of your synchronization and how you want to handle updates.
Here's a general guideline on where to place the `syncBackAfterProcessing` function:

In the FirstSync Script:

- **At the Beginning:** If placed at the beginning of the firstSync, it ensures that changes are sent back to the source immediately when the first sync occurs. This can be useful if you want to immediately reflect any initial changes made during the creation or initial synchronization of the issue.

- **At the End:** If placed at the end of the firstSync, it ensures that any changes made during the synchronization process are reflected back to the source. This is typically more common and useful as it allows the synchronization process to complete all necessary updates before sending back any changes.

General Recommendation:

- It is generally recommended to place `syncBackAfterProcessing` at the end of the firstSync script. This way, you ensure that all the necessary synchronization and processing steps have been completed before changes are propagated back to the source instance.

Example of Placement at the End of firstSync:

```
if(firstSync){
  issue.projectKey   = "CLOUD"
  issue.typeName     = nodeHelper.getIssueType(replica.type?.name, issue.projectKey)?.name ?: "Task"
     Product
  //Rest of required fields for issue creation
     Release History
  syncHelper.syncBackAfterProcessing()
}
```

Product

About Us

Release History

GitHelper

API Reference

Security

Pricing and Licensing

Resources

Subscribe for a weekly Exalate hack

Academy

Blog

YouTube Channel

Ebooks

Still need help?

Join our Community

Visit our Service Desk

Find a Partner