Custom Fields are Randomly Updating

Last Modified on 12/30/2025 11:26 am EST

This article documents behavior that has been found out by one of our customers. This article provides details of the cause and ways to work around it.

The Problem

> **Note**: In the context of release 5.1 of exalate, the store function was introduced.

The store function allows to apply changes in the middle of the incoming sync script.
Check for more information on How to use a store(issue) function.

Most platforms allow automation which is triggered whenever a value is changed.
Typically, whenever one value is set, another custom field needs to be adapted.

A typical example is as follows

- You have the custom field 'SLA respected' which is either 'No' or 'Yes'
- When the issue is resolved within the SLA criteria, the custom field is automatically changed from 'No' to 'Yes'

What we found out is that when the exalate store() function is used, the value is reset to 'No' - while it should remain to 'Yes'

The Cause

To explain how this can happen - it is important to understand how Exalate is applying changes that are composed in the incoming sync processor.
Each time that the incoming sync processor is run, the following steps are followed

1. Exalate extracts the entity under sync and populates a structure in memory - called 'hub-issue'
   This hub issue contains the values of all fields
2. Exalate runs the incoming sync processor, applying any of the changes to the hub-issue
3. The resulting hub issue is applied to the real entity.

With the introduction of the store function, that sequence is influenced

1. Exalate extracts the entity under sync and populates a structure in memory - called 'hub-issue'

This hub issue contains the values of all fields

2. Exalate runs the incoming sync processor, applying any of the changes to the hub-issue
3. The store function will apply all the changes to the entity
4. The incoming sync script is continued
5. The resulting hub issue is applied again to the entity.

The root cause of the problem is that when step 3 in the previous sequence is executed, the automation will update the entity independently from exalate.
In step 5 this update (of automation) will be reverted to the previous value.

Illustration on custom field 'SLA respected'

	Field on Entity	hubissue	Action of Exalate	Automation
	Field on Entity	hubissue	Action of Exalate	Automation
1	No		Exalate retrieves the data	
2	No	No	Incoming sync is started	
3	No	No	Store Function is applied,	Automation changes the value of the field on the entity to yes
4	Yes	No	Incoming sync is continued	
5	No	No	Hubissue is applied onto the entity.	

Workaround

This is considered a bug and will be resolved in the platform.

Workaround: In the incoming sync script remove the custom fields from the hubissue which should not be applied

- look up the customField name and customField id of the custom field that should not be updated
- Add following code to the incoming sync script

```
// replace issue with the name of the entity

issue.customFields.remove("SLA Respected")


// 12345 is the field id as known by the tracker
issue.customFields.remove("12345")
```

The reason why this works is that by removing the customField from the hubIssue, it will not be applied.

Product
About Us
Release History
Changelog
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