Syncing cascading select custom fields

JIRA ON-PREMISE | JIRA CLOUD

This article shows how to synchronize a custom field of type ***cascading select list.***

Cascading select field allows to set values for the list options. The value in the cascading select field is of type ***cascading*** and consists of two dependent options: **parent value** and **child value**.

Exalate uses the following helper methods to sync cascading select fields:

- getCascadingSelect
- getOption

Source side

Include the custom field with name 'Cascading select' in the replica sent to the other side

Outgoing sync

```
replica.customFields."Cascading select" = issue.customFields."Cascading select"
```

Destination side

You can sync cascading select list option values in different ways:

Cascading select to cascading select

It's possible that option values don't match on source and destination sides.

There are different ways to handle this. If the option value does not exist on the receiving side Exalate can do the following:

- don't sync
- throw an error (helps to debug, add value to fix the error)
- check if at least one value exists and sync only that value
- create value with the help of external scripts

- **option values match**

  Source Region/Country - a cascading select field on the source side; ***Destination Region/Country*** - a cascading select field on the destination side

```
//check if at least one option value exists and sync only existing values
def sourceRegion = replica.customFields."Source Region/Country"?.value?.parent?.value
def sourceCountry = replica.customFields."Source Region/Country"?.value?.child?.value

def region = nodeHelper.getOption(
  issue,
  "Destination Region/Country",
  sourceRegion
)
def country = region.childOptions.find{it.value == sourceCountry}
if ( region != null && (sourceCountry == null || country != null)) {
  issue.customFields."Destination Region/Country"?.value = nodeHelper.getCascadingSelect(
              region,
              country
  )
} else if (sourceRegion == null) {
  issue.customFields."Destination Region/Country"?.value = null
}
```

Set default cascading select list values manually

You can set default cascading select values using nodeHelper.getCascadingSelect method.

There're multiple ways to set default values:

Destination Region/Country - cascading select field name on your side;

```
def parent = nodeHelper.getOption(issue, "Destination Region/Country", "Europe")
issue.customFields."Destination Region/Country".value = nodeHelper.getCascadingSelect(
       parent,
           parent.childOptions.find{it.value == "Belgium"}
)
```

Cascading select list options to text custom field

Exalate allows extracting the received option values from the cascading select to a text field.

For example, you have a custom field called **Cascading Select** (multi-select) to the custom field **Cascading field details** (text field).

Receive Cascading Select field values and display them in the text custom field:

Incoming sync

replica.customFields."Cascading select".value is an object of type Cascading Select List. It contains **parent value** and **child value** of the cascading select list field

issue.customFields."Cascading field details".value is a text type custom field, which includes string value.

```
issue.customFields."Cascading field details".value  = replica.customFields."Cascading select"?.value?.parent?.
value +
         ", " +  replica.customFields."Cascading select"?.value?.child?.value
```

See also

How to Synchronize Custom Fields

Back to General articles