

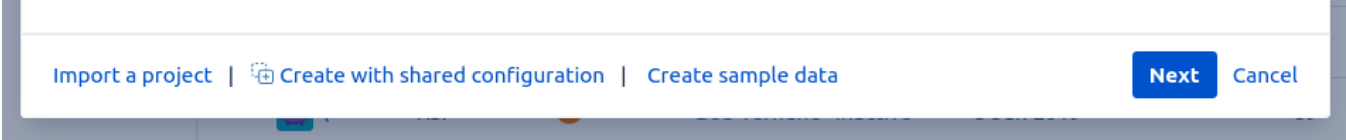
# Jira Bulk Move pitfalls

Say you have a large Jira project with components for things like UI, Docs, API, Mobile, Auth, etc. Over time you realise that the Auth code is reusable. So you turn it into a library, create a new AUTH Jira project, and now you want to Bulk Move all `component=Auth` issues into your new AUTH project.

Sounds simple, right? Just create a new project and Bulk Move? How hard can it be?

- Bulk Move bug: Don't move sub-tasks with their parent
- Solution: move just the parents
- Missing versions and components
  - Solution: the Jira Version/Component Cloner
- Bulk Move bug: CC field
- Bulk Move bug: 'Epic Name is required'
- Summary

Let's start. We assume you have created the destination project, using the 'Create with shared configuration' option to ensure all workflows and field configurations are identical:



Do a JQL search `project=SRC and component=Auth` , hit the Bulk Move and..

Wait, we've already made a mistake.

The bulk move, if it ever succeeds, processes issues sequentially. Our JQL has no order defined. It might return the last issue, SRC-10000 before the first, SRC-1, in which case SRC-10000 might be mapped to DST-1. We want our destination issue keys in creation order, just like the source.

So amend your JQL to `project=SRCPROJ and component=Auth ORDER BY id ASC`

Proceed with the bulk move:

## Bulk Operation

- Choose Issues  
Selected **4298** issues from **1** project(s)
- Choose Operation
- Operation Details
- Confirmation

### Step 2 of 4: Choose Operation

Choose the operation you wish to perform on the selected **4298** issue(s).

<input type="radio"/>	Edit Issues	Edit field values of issues
<input checked="" type="radio"/>	Move Issues	Move issues to new projects and issue types
<input type="radio"/>	Transition Issues	Transition issues through workflow

Slow, but so far so good..

## Bulk Operation

- Choose Issues

Selected **4298** issues from **1** project(s)

- Choose Operation

- Operation Details

- Confirmation

### Step 3 of 4: Select Projects and Issue Types



You have chosen to move **4298** issues from **1** project(s) with **11** issue type(s). You can either select a new project and issue type for each of the existing **18** combinations below or choose to move all standard issues to a single project and issue type.

Click the help icon to get more information on the Bulk Move process.

**⚠ Please note that 195 sub-task issues were removed from the selection and do not appear in the table below. You are not allowed to bulk move sub-task issues together with their parent issue. In this case, you will only be asked to move the sub-task if you move the parent issue to a new project.**

[Next](#) [Cancel](#)

**i** The change will affect **1497** issues with issue type(s) **Bug** in project(s) **SrcProj**.

Move		To
SrcProj	→	DestProj (DestProj) ▼
Bug	→	Bug ▼
<input type="checkbox"/> Use the above project and issue type pair for all other combinations.		

In Step 3 we get to choose the destination project for every issue type in our set of issues.

## Bulk Move bug: Don't move sub-tasks with their parent

Things are normal (as above) until we get to sub-tasks, and it all goes wrong:

**i** The change will affect **1** issues with issue type(s) **Sub-task** in project(s) **SrcProj** and parent issue(s) **SRC-276968**.

Move		To
SrcProj	→	SrcProj (SRC) ▼
Sub-task	→	Sub-task ▼
Select Parent Issue:		SRC-276968 - Persona will be e... ▼
Begin typing to find recently viewed issues		

Jira is demanding that we provide a 'Parent Issue'. But surely sub-tasks already have a parent? One that we're moving as part of the batch?

Well yes, but the Bulk Move wizard is dumb as rocks. It doesn't know that the sub-task parents are being moved too. It acts as if you are moving a sub-task all on its own, which will thus need a new parent in its new project.

## Solution: move just the parents

The solution is to do a JQL search for just the 'parent', non-subtask issues:

```
project=SRCPROJ and component=Auth and issuetype not in subtaskIssueTypes()
```

Bulk move these, and the sub-tasks automatically come along for the ride.

## Missing versions and components

Moving on, we get to 'Step 3':

## Bulk Operation

- Choose Issues  
Selected **4090** issues from **1** project(s)
- Choose Operation
- Operation Details
  - - Bug
  - - Epic
  - - Story
  - - Inquiry
  - - Milestone
  - - Task.
  - - Incident
  - - Test
  - - Improvement
- Confirmation

### Step 3 of 4: Update Fields for Target Project 'DestProj' - Issue Type 'Epic' ?

Update fields for issues with current issue type(s) **Epic** in project(s) **SrcProj**.

**Retain Original Values:** it is possible to retain original field values where the original value is valid within the target destination. This can be achieved by checking the checkbox associated with the required field

- **Checked:** All valid original field values will be retained. The field will not be updated with the new value.
- **Unchecked:** All field values will be overwritten with the new value.

[Next](#) [Cancel](#)

Field Name	Field Value	Retain
Component/s	Auth [Project: SrcProj] → <input type="text" value="Unknown"/>	<input checked="" type="checkbox"/>
Fix Version/s	SRC-2020 [Project: SrcProj] → <input type="text" value="Unknown"/>	<input checked="" type="checkbox"/>
	auth-1.0 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	auth.20.12 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	DestProj-FY23-Q3 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	DestProj-FY23-Q4 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	DestProj-FY24-Q1 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	IntSol-FY24-Q1 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	DestProj-FY24-Q2 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	DestProj-FY24-Q3 [Project: SrcProj] → <input type="text" value="Unknown"/>	
	DestProj-FY24-Q4 [Project: SrcProj] → <input type="text" value="Unknown"/>	

If you want to preserve your issue versions and components, identically named versions/components must exist in the destination project. Otherwise the Bulk Move wizard won't find anything to map them to.

So you'll have to recreate relevant versions and components in the destination project.

"But", you say, "I have 4000+ issues to move. How do I know which versions and components those issues actually use? Also, my source project has <checks database> 1013 versions and 58 components. I'll go crazy recreating all those by hand!"

Let me spoil the surprise, and say that we're going to end up doing this with a hacky Python script. But humour me for a bit while we investigate what Normal People would do.

Firstly, what *doesn't* work is simply creating a new destination Project using 'Create with shared configuration' with the source project:

[Import a project](#) | [Create with shared configuration](#) | [Create sample data](#)

[Next](#) [Cancel](#)

Your destination project won't contain versions and components, or role mappings for that matter.

One way to *properly* copy a project, if you have ScriptRunner (as everyone should) is to use the ScriptRunner 'Copy Project' built-in script:

Browse  
Console  
Built-in Scripts  
Jobs  
Listeners  
Fields  
Behaviours  
Workflows  
Fragments  
JQL Functions  
REST Endpoints  
Resources  
Mail Handler  
Script Editor  
Settings






Browse Console Built-in Scripts Jobs Listeners Fields Behaviours Workflows F

## Copy project

This tool will create a new project, from the configuration of another project. This includes: Sch Custom field configurations. Optionally: Issues, Versions, Components, Request types, Queues,

### Documentation & Tips

#### Source project

 SrcProj  

Project to copy

Select a service desk project to get more options

#### Target project key

DESTPROJ

Enter the key of the project to be created

#### Target project name

Destination Project

Enter the name of the project to be created

#### Copy versions



Do you want versions to be replicated in the target project?

#### Copy components



Do you want components to be replicated in the target project?

#### Copy issues




Do you want all the issues to be copied to the target project? You must also opt to copy Components and Versions.

#### Copy project-specific dashboard and filters



Experimental. Copy dashboards and filters beginning with PKEY-. Will update JQL, and gadgets etc.

#### Copy rapid board

Select... 

Copy the *configuration* of an associated rapid board. The board will be set up for the new project.

#### Target board name

Name to give the copied board

**Run** Preview Cancel

#### Result

Project copied to DESTPROJ

I have 3 gripes with this, from very minor to major.

- 1) *Minor* - The copy is good but not perfect. Version Start Date is lost, as is component 'archived' status
- 2) *Medium* - I didn't actually want *all* versions (1000+ in my case) copied! Not all will be relevant to moved issues
- 3) *Major* - After this I *still* have some (older) versions not being mapped by the Bulk Move:

Fix Version/s	abc-2020 [Project: ABC]	→	Unknown ▼
	abc-1.0 [Project: ABC]	→	Unknown ▼
	abc.20.12 [Project: ABC]	→	Unknown ▼
	abc.20.13 [Project: ABC]	→	Unknown ▼
	abc.20.15 [Project: ABC]	→	Unknown ▼
	abc.20.20 [Project: ABC]	→	Unknown ▼
	UX3-1.14.0 [Project: ABC]	→	Unknown ▼

This is because these particular versions are **archived**, and to the Bulk Move wizard, archived versions are invisible.

So we don't actually want a perfect version cloner/copier. We want copied-but-unarchived versions, at least until the bulk move is done.

As a workaround, you could un-archive the destination project's versions by hand, do your Bulk Move and re-archive them afterwards.

But this is all getting tedious, so let's get back to that Hacky Script I promised earlier, which solves all our problems.

## Solution: the Jira Version/Component Cloner

It lives at:

<https://github.com/redradishtech/jira-versioncomponent-cloner>

Simply, a python script that, given a source and destination project, will recreate the source project's components and versions in the destination project. Follow the README to see how it works.

It has two features relevant to our Bulk Move needs:

- versions can be created un-archived, before later (after the bulk move) being archived as needed
- we can choose a subset of versions/components to copy, e.g. only those relevant to our issues being moved

Run first with `unarchive=True` as the README suggests, to create un-archived versions. Then after finishing the bulk move, run again with `unarchive=False`.

After running the script with `unarchive=True`, Bulk Move should detect the matching versions and components:

## Bulk Operation

- Choose Issues  
Selected **4090** issues from **1** project(s)
- Choose Operation
- Operation Details
  - - Bug
  - - Epic
  - - Story
  - - Inquiry
  - - Milestone
  - - Task.
  - - Incident
  - - Test
  - - Improvement
- Confirmation

### Step 3 of 4: Update Fields for Target Project 'DestProj' - Issue Type 'Epic' ?

Update fields for issues with current issue type(s) **Epic** in project(s) **SrcProj**.

**Retain Original Values:** it is possible to retain original field values where the original value is valid within the target destination. This can be achieved by checking the checkbox associated with the required field

- **Checked:** All valid original field values will be retained. The field will not be updated with the new value.
- **Unchecked:** All field values will be overwritten with the new value.

[Next](#) [Cancel](#)

Field Name	Field Value	Retain
Component/s	Auth [Project: SrcProj] → <input type="text" value="Auth"/>	<input checked="" type="checkbox"/>
Fix Version/s	SRC-2020 [Project: SrcProj] → <input type="text" value="SRC-2020"/> auth-1.0 [Project: SrcProj] → <input type="text" value="auth-1.0"/> auth.20.12 [Project: SrcProj] → <input type="text" value="auth.20.12"/> DestProj-FY23-Q3 [Project: SrcProj] → <input type="text" value="DestProj-FY23-Q3"/> DestProj-FY23-Q4 [Project: SrcProj] → <input type="text" value="DestProj-FY23-Q4"/> DestProj-FY24-Q1 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q1"/> IntSol-FY24-Q1 [Project: SrcProj] → <input type="text" value="IntSol-FY24-Q1"/> DestProj-FY24-Q2 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q2"/> DestProj-FY24-Q3 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q3"/> DestProj-FY24-Q4 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q4"/>	<input checked="" type="checkbox"/>

## Bulk Move bug: CC field

On the same Step 3 page, you may see a **CC** field:

CC

Start typing to get a list of possible matches.  
People to be CC'd on the issue

The CC field appears if you have the **Jira Watcher Field** plugin installed:

## User-installed apps

>

JIRA Component Watcher

▼

JIRA Watcher Field

Custom field type that allows watchers to be modified on the creation and update of issue.

Uninstall

Disable

No screenshots available

Version: 2.8.5

Vendor: (unknown)

App key: com.burningcode.jira.issue.customfields.impl.jira-watcher-field

6 of 6 modules enabled


The Jira Watcher plugin lets you edit the watcher list on regular issue transition screens, as the CC field above. Normally this is what we want, but not during a bulk move.

There is no good option here. If you leave **CC** blank, your issue watchers will all be lost. If you fill something in, your watchers will be overwritten.


Solution: abort the Bulk Move and disable the watcher field for the duration of the bulk move:

## User-installed apps

>

 JIRA Component Watcher

▼

 JIRA Watcher Field

Custom field type that allows watchers to be modified on the creation and update of issue.

Uninstall


Enable

No screenshots available

Version: 2.8.5

Vendor: (unknown)

App key: com.burningcode.jira.issue.customfields.impl.jira-watcher-field

 0 of 6 modules enabled

When you try again the CC fields should be gone.

## Bulk Move bug: 'Epic Name is required'

The next problem may show up in Step 3 when you get to Epics. The bulk move wizard says **Epic Name is required** :

## Bulk Operation

- Choose Issues  
Selected **4090** issues from **1** project(s)
- Choose Operation
- Operation Details
  - - Bug
  - - Epic
  - - Story
  - - Inquiry
  - - Milestone
  - - Task
  - - Incident
  - - Test
  - - Improvement
- Confirmation

### Step 3 of 4: Update Fields for Target Project 'DestProj' - Issue Type 'Epic' ?

Update fields for issues with current issue type(s) **Epic** in project(s) **SrcProj**.

**Retain Original Values:** it is possible to retain original field values where the original value is valid within the target destination. This can be achieved by checking the checkbox associated with the required field

- **Checked:** All valid original field values will be retained. The field will not be updated with the new value.
- **Unchecked:** All field values will be overwritten with the new value.

[Next](#) [Cancel](#)

Field Name	Field Value	Retain
Component/s	Auth [Project: SrcProj] → <input type="text" value="Auth"/>	<input checked="" type="checkbox"/>
Fix Version/s	SRC-2020 [Project: SrcProj] → <input type="text" value="SRC-2020"/> auth-1.0 [Project: SrcProj] → <input type="text" value="auth-1.0"/> auth.20.12 [Project: SrcProj] → <input type="text" value="auth.20.12"/> DestProj-FY23-Q3 [Project: SrcProj] → <input type="text" value="DestProj-FY23-Q3"/> DestProj-FY23-Q4 [Project: SrcProj] → <input type="text" value="DestProj-FY23-Q4"/> DestProj-FY24-Q1 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q1"/> IntSol-FY24-Q1 [Project: SrcProj] → <input type="text" value="IntSol-FY24-Q1"/> DestProj-FY24-Q2 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q2"/> DestProj-FY24-Q3 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q3"/> DestProj-FY24-Q4 [Project: SrcProj] → <input type="text" value="DestProj-FY24-Q4"/>	<input checked="" type="checkbox"/>
Epic Name	<b>Epic Name is required.</b> <input type="text"/> Provide a short name to identify this epic.	<input checked="" type="checkbox"/>
Affects Version/s	SRC-2020 [Project: SrcProj] → <input type="text" value="SRC-2020"/> Version to which the fix should be targeted.	<input checked="" type="checkbox"/>

☒ Send mail for this update

By selecting this option, an update notification will be sent for each issue affected by this bulk operation

[Next](#) [Cancel](#)

What is going on? Well, it's true that for Epics, **Epic Name** is a required field. But all our moved Epics already have an **Epic Name**! So why are we being asked for one?

For me, it turned out that all my Epics did *not* have an **Epic Name**. Despite the relevant JQL (`project = SrcProj AND issuetype = Epic AND "Epic Name" is empty`) returning nothing, JQL shows there are Epics without a Name:



```

SELECT project.pkey || '-' || jiraissue.issuenum AS issuekey,
       cfv.stringvalue AS "Epic Name"
FROM project
JOIN jiraissue ON project.id=jiraissue.project
JOIN issuetype ON issuetype.id=jiraissue.issuetype
LEFT JOIN customfieldvalue cfv ON cfv.issue=jiraissue.id
LEFT JOIN customfield cf ON cf.id=cfv.customfield
WHERE project.pkey='SRC'
      AND issuetype.pname='Epic'
      AND cf.cfname='Epic Name'
      AND cfv.stringvalue IS NULL;

issuekey    Epic Name

SRC-178105
SRC-178099
SRC-178123
SRC-178143
SRC-178101
SRC-178140
SRC-178098
SRC-178083
SRC-178139
SRC-178142
SRC-178095
SRC-178132
SRC-178147
SRC-178133

(14 rows)

```

In fact these are trivial to create: start with an issue of any other type (a Story, for instance), click the 'Type' attribute and change it to Epic, and bam - you've got yourself a nameless Epic.

For the bulk move, it appears that *if you check the 'Retain' checkbox*, what you enter is only used for move issues with a blank Epic Name – i.e. it's not going to overwrite Epic Names everywhere. So just set a fake Epic Name. Make it unique enough that you can later search for it in JQL or the database; e.g. myepicname:

Epic Name	<div>Epic Name is required.</div> <div>myepicname</div> <div>Provide a short name to identify this epic.</div>	<input checked="" type="checkbox"/>
-----------	--	-------------------------------------

Also, *check the "Retain" checkbox* on the field. Otherwise all your other Epics will have their Epic Names overwritten.

I didn't either the first time! Here is some SQL to recover the lost Epic Name from the change history:

```

UPDATE customfieldvalue cfv
SET stringvalue=oldstring
FROM
  (SELECT cg.issueid,
         ci.oldstring
   FROM changegroup cg
   JOIN changeitem ci ON ci.groupid=cg.id
   WHERE ci.newstring='myepicname'
        AND ci.field='Epic Name') history
WHERE history.issueid=cfv.issue
      AND customfield=
  (SELECT id
   FROM customfield
   WHERE cfname='Epic Name') RETURNING cfv.issue;

```

## Summary

Things we've learned:

- Always append `order by ID ASC` to the JQL of issues you're bulk moving, so the issue keys ascend in order of issue creation.

- Do not try to bulk-move sub-tasks. Only bulk-move the parents. Append to your search JQL: `issuetype not in subtaskIssueTypes()`
- The destination project must have versions and components defined for incoming issues. The versions must be un-archived. Do this with [code](#).
- Disable the **Jira Watcher Field** plugin before you bulk move, and re-enable it after
- Check the 'Retain' box wherever possible, and especially on the **Epic Name** field if it appears.

Once you're done, don't forget to re-run `JiraVersionComponentCloner.py` with `unarchive=False`. If you used the Jira Watcher Field plugin, re-enable it after the move.