

# ScriptRunner's issueFunction not available to all users?

I recently encountered a Jira instance with ScriptRunner installed, where there **issueFunction** custom field provided by ScriptRunner was not available to all users. The error was `Field 'issueFunction' does not exist or you do not have permission to view it.`

Search

Save as

Email

Edit

issueFunction in epicsOf()



Search

Basic

Field 'issueFunction' does not exist or you do not have permission to view it.

This happened for some users, but not all.

And indeed, navigating to the Custom Fields admin screen ('gg custom fields'), searching for `issueFunction` revealed that it was scoped to just one project – a project to which not all users had access:

## Custom fields

Optimize

Add custom field

Add extra fields to your issues to define them more precisely. From simple text fields to development summaries, you can create different types of custom fields and configure how they're displayed to your users. Here you can manage your existing custom fields, or create new ones.

issueFunction



Project: All

Type: All

Screen: All

Name	Type	Available Contexts	Screens	Actions
<b>issueFunction</b> <b>LOCKED</b> Field created by script runner plugin for advanced issue fun...	JQL Functions ...	1 project	0 screens	

### Configure Custom Field: issueFunction **LOCKED**



Below are the Custom Field Configuration schemes for this custom field. Schemes are applicable for various issues types in a particular context. You can configure a custom field differently for each project context or in a global context. Moreover, project level schemes will override global ones.

- [View Custom Fields](#)

#### Default Configuration Scheme for issueFunction

Default configuration scheme generated by JIRA

Applicable contexts for scheme: Issue type(s):  
Global (all issues)  
Project(s):  
Administration

How did this field get scoped like this? I don't know.

How do we fix it? The field is 'locked', so the configuration scheme cannot be edited.

Time for a bit of database spelunking.

Here we see our custom field:

```
jira=> select * from customfield where cfname='issueFunction';
```

[ RECORD 1 ]	
id	10900
customfieldtypekey	com.onresolve.jira.groovy.groovyrunner:jqfFunctionsCustomFieldType
customfieldsearcherkey	com.onresolve.jira.groovy.groovyrunner:jqfFunctionsSearcher
cfname	issueFunction
description	Field created by script runner plugin for advanced issue functions. Do not place on any screen.
defaultvalue	NUL
fieldtype	NUL
project	NUL
issuetype	NUL
cfkey	NUL

Custom fields can be scoped to a set of projects, and also scoped to a set of issue types. There are two sets of tables achieving this:

	Table 1 (joined to customfield)	Table 2 (joined to table 1)
Per-project scoping	fieldconfiguration	configurationcontext
Per issue type scoping	fieldconfigscheme	fieldconfigschemeissuetype

For our `issueFunction` field here are the tables:

### Per-project tables

```
jira=> select * from fieldconfiguration where fieldid='customfield_10900';
```

id	configname	description	fieldid	customfield
11000	Default Configuration for issueFunction	Default configuration generated by Jira	customfield_10900	NUL

(1 row)

Time: 0.419 ms

```
jira=> select * from configurationcontext where fieldconfigscheme=11000;
```

id	projectcategory	project	customfield	fieldconfigscheme
13610	NUL	10600	customfield_10900	11000

(1 row)

Here we see, as expected, that `issueFunction` is scoped to one project with ID 10600. There would be more than one `configurationcontext` table if more projects were selected.

### Per-issuetype tables

```
jira=> select * from fieldconfigscheme where fieldid='customfield_10900';
```

id	configname	description	fieldid	customfield
11000	Default Configuration Scheme for issueFunction	Default configuration scheme generated by JIRA	customfield_10900	NUL

(1 row)

Time: 0.412 ms

```
jira=> select * from fieldconfigschemeissuetype where fieldconfigscheme=11000;
```

id	issuetype	fieldconfigscheme	fieldconfiguration
13710	NUL	11000	11000

(1 row)

As `issueFunction` is not scoped per issue type ( `issuetype` is null).

### Fixing our field scoping


The fix is quite simple: in `configurationcontext` we want to change that project ID (10600) to null.

You can do this via `update configurationcontext set project=null where id=...`; but here is SQL that is a bit safer:

```
update configurationcontext set project=null where customfield='customfield_' || (select id from customfield
where cfname='issueFunction' and customfieldtypekey='com.onresolve.jira.groovy.groovyrunner:
jqlFunctionsCustomFieldType') and project is not null;
```

Normally you would need to restart Jira to pick up this change, but thanks to ScriptRunner's ever-useful **Clear Groovy classloader or Jira internal caches** [built-in script](#) we don't have to:

[manage apps](#) [user management](#) [latest upgrade report](#) [system](#)

 **ScriptRunner** [Settings](#) [...](#)

[Browse](#) [Console](#) [Built-in Scripts](#) [Jobs](#) [Listeners](#) [Fields](#) [Behaviours](#) [Workflows](#) [More](#) [v](#)

### Clear Groovy classloader or Jira internal caches

Clear the Groovy internal caches (if this is not working automatically), or the Jira caches if you have changed something in the database. Expect a delay after executing this.

Documentation & Tips

Show

Which cache?

☐ Groovy class loader

☒ Jira internal caches

Which cache do you want to clear?

Preview

Run

Cancel

Result

Jira cache cleared.

After this, all users should be able to run JQL involving `issueFunction`.