Home

This is the home of the Kepler Custom Fields space.

To help you on your way, we've inserted some of our favourite macros on this home page. As you start creating pages, blogging and commenting you'll see the macros below fill up with all the activity in your space.

Recently Updated

KCF30
Feb 28, 2017 • attached by Alexandru Geageac

Kepler Custom Fields 3.0
Feb 24, 2017 • created by Confluence Administrator

What should I do if I installed an incompatible version?
Dec 02, 2015 • created by Alexandra Topoloaga

Install notes for JIRA 7
Nov 16, 2015 • created by Alexandra Topoloaga

Requirements
Nov 16, 2015 • created by Alexandra Topoloaga

Backup and restore
Feb 19, 2015 • created by Alexandru Geageac

Log Custom Field
Nov 25, 2014 • created by Confluence Administrator

User Guide
Sep 12, 2014 • created by Maria Cirtog

Manual Uninstall
Sep 12, 2014 • created by Maria Cirtog

Uninstall via Atlassian Universal Plugin Manager
Sep 12, 2014 • created by Maria Cirtog

uninstallKcf_success.png
Sep 12, 2014 • attached by Maria Cirtog

uninstallKcf_2.png
Sep 12, 2014 • attached by Maria Cirtog

uninstallKcf.png
Sep 12, 2014 • attached by Maria Cirtog

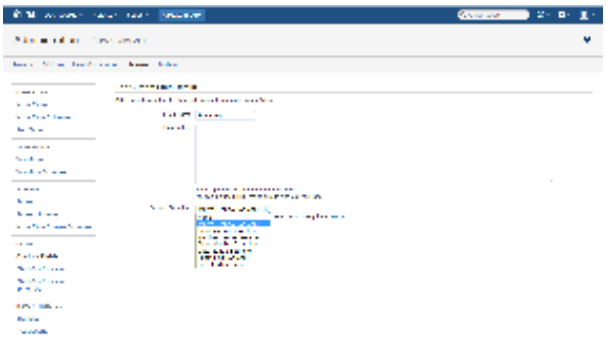
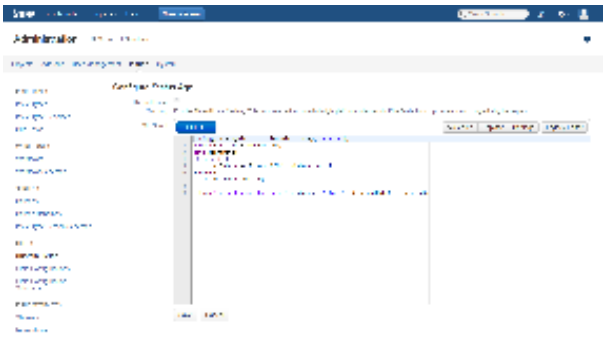
plugins admin menu.PNG
Sep 12, 2014 • attached by Maria Cirtog

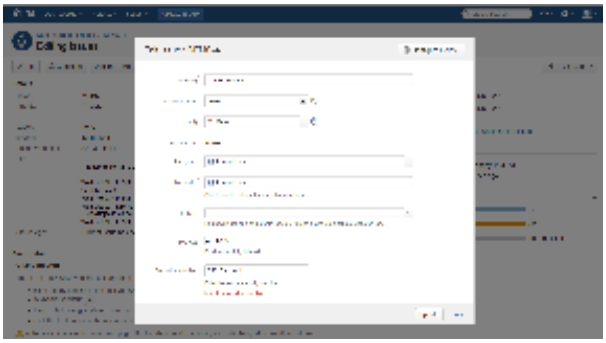
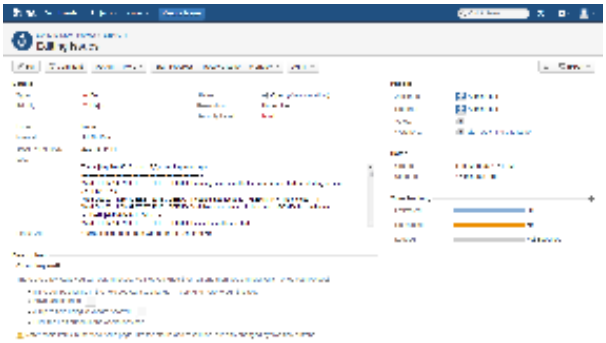
Manual Install
Sep 12, 2014 • created by Adrian Iasinschi

Navigate space

Kepler Custom Fields Documentation

Gallery


Available Custom Fields

A suite of useful custom fields developed by Kepler-Rominfo.

New Features in version 1.1.1 and 2.1.1

- Log Custom Field

New Features in version 1.1 and 2.1

- Regex Custom Field
- SIL Script Custom Field

Custom Fields in version 1.0 and 2.0

- Interval Custom Field

Installation

Installation via Atlassian Universal Plugin Manager

This page points the simple steps to follow for installing the plugin using the Universal Plugin Manager. This method requires an internet connection.

Manual Install

It may seem more complicated, but a manual install is quite easy to do. After all, all you have to do is to copy some files. Here's how.

> **Good practice**
> If you are already using a previous version of this plugin and you are installing a new one, please clear the cache of your browser after the installation to avoid errors due to the javascript changes.

Installation via Atlassian Universal Plugin Manager

Installation via Atlassian Universal Plugin Manager

If you are not familiar with Universal Plugin Manager (UPM), please read this document before we begin.

Steps are simple:

1. Enter the administration screen and go to *Plugins->Install* tab.

2. Search for **katl-commons** plugin and install it. This is a required dependency for Kepler Custom Fields.

3. Search for **keplercf** plugin and install it.

That's all.

Install notes for JIRA 7

When upgrading from an older version of JIRA to JIRA 7, you must update all our plugins as well.

As you can see on this page, the versions compatible with JIRA 7 are the 3.1.x versions.

What should I do if I installed an incompatible version?

As we have said before, **3.0.x** versions are compatible with **JIRA 6.x** and **3.1.x** versions are compatible with **JIRA 7.x**.

If you have installed KCF 3.0.x on JIRA 7.x or KCF 3.1.x on JIRA 6.x, you should do the next steps :

1. Uninstall katl-commons
2. Uninstall KCF
3. Install the right version of KCF (the one compatible with your JIRA)
4. katl-commons should now have the right version as well

> After you uninstall katl-commons, some plugins may remain disabled, so you may need to re-enable them manually.

Manual Install

Manual Install

Do not worry, it's a simple task to install it manually:

1. Download the correct kepler custom fields obr file from Atlassian Marketplace or from our site: Kepler Products.

2. Go to Administration->Add-ons->Manage add-ons. Install the previously downloaded obr file by using 'Upload add-on' link.

3. [Optional, but highly recommended]: Enable logging on our modules. Open with a text editor of your choice the JIRA log4j configuration file *JIR*

A_INSTALL_DIR/atlassian-jira/WEB-INF/classes/log4j.properties and add these 2 lines at the end of it. Restart JIRA.

```
log4j.logger.com.keplerrominfo=INFO, filelog
log4j.additivity.com.keplerrominfo=false
```

Uninstall

Uninstall via Atlassian Universal Plugin Manager

This page shows the steps to uninstall the plugin using the Universal Plugin Manager.

Manual Uninstall

At first sight, this seem a little bit complicated but actually it isn't. All it has to be done is to remove the plugin manually and delete its tables from the internal database.

Manual Uninstall

Uninstall manually

> You need to have access where the Jira server has been installed.

Goto the folder where Jira server has been installed.

Access **<JIRA_APPLICATION_DATA>/plugins/installed-plugins** and manually delete `Kepler Custom Fields` plugin.

Restart the server

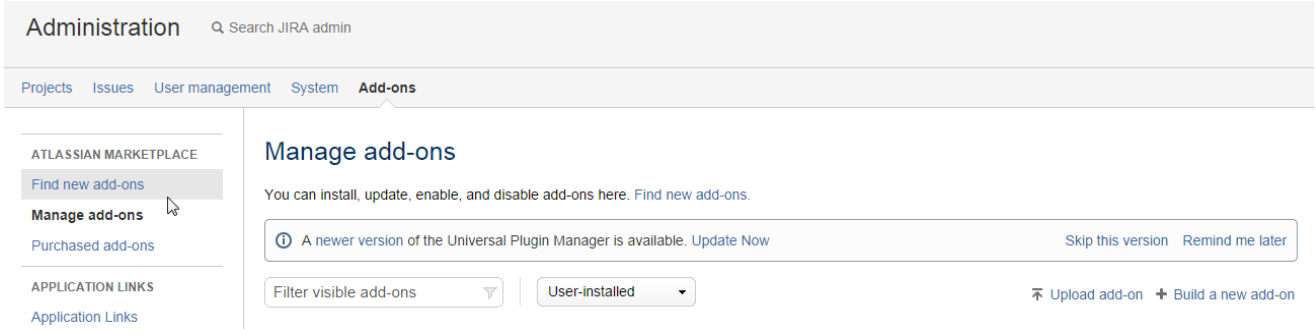
Now you can restart Jira server

Uninstall via Atlassian Universal Plugin Manager

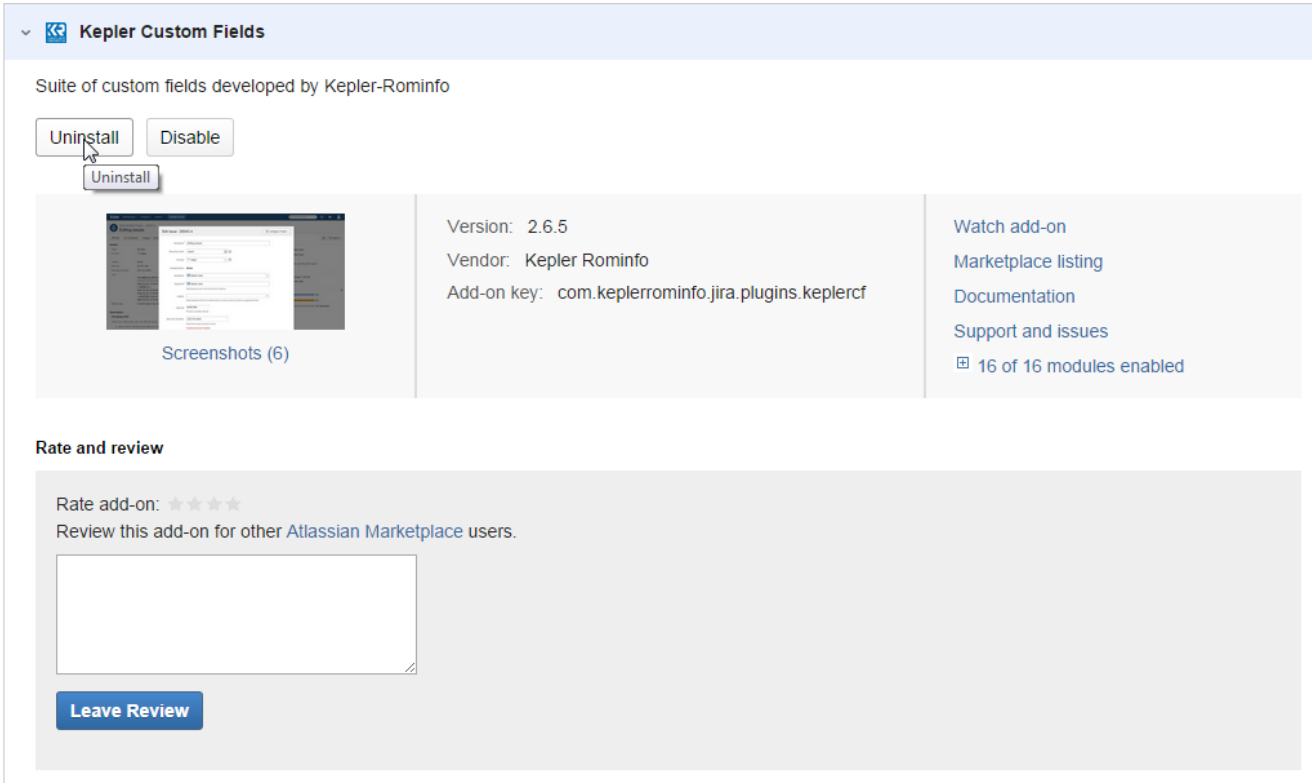
Uninstall via Atlassian Universal Plugin Manager

If you are not familiar with Universal Plugin Manager (UPM), please read this document before we begin.

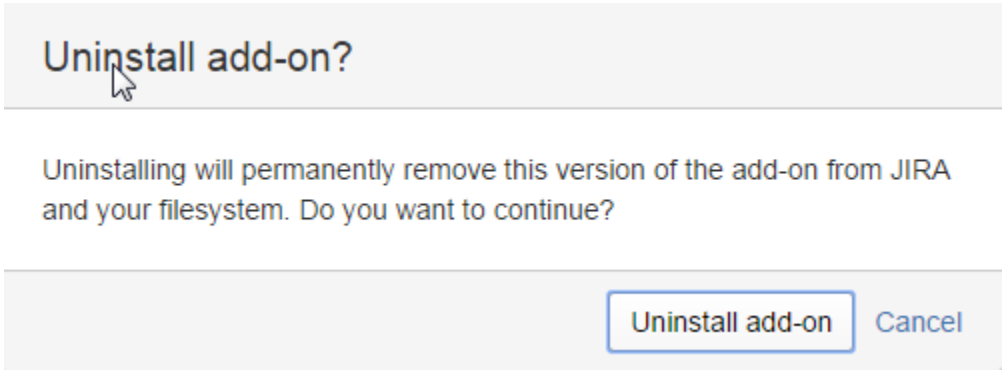
1) Log in as administrator and go to Administration->Add-ons->Manage add-ons



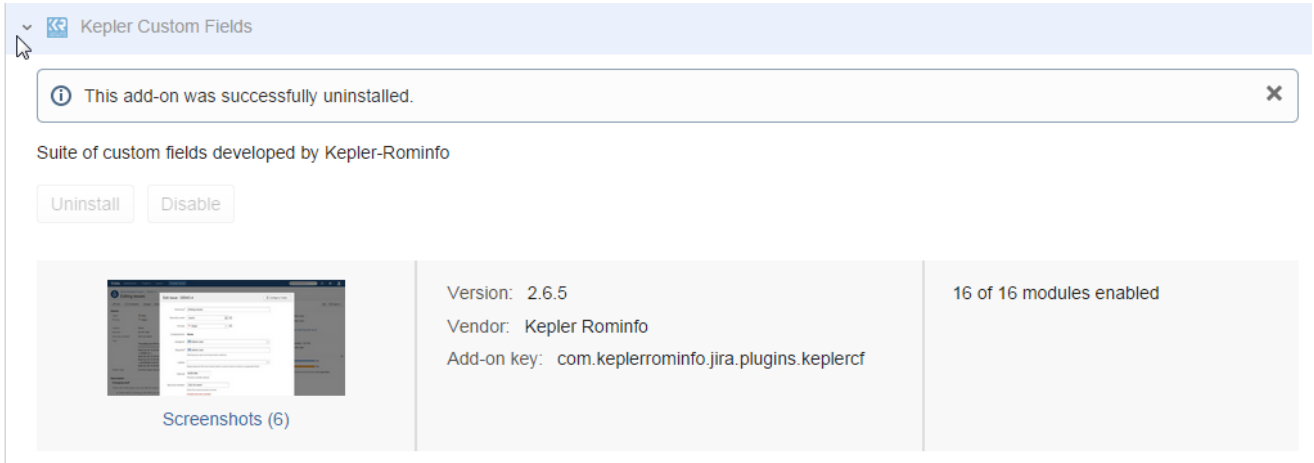
2) Search for `Kepler Custom Fields` plugin in `User-installed add-ons` section and click on `Uninstall` button



3) Press `Uninstall add-on` when the uninstall confirmation dialog box appears



4) A message "successfully uninstalled" should appear



User Guide

This documentation refers to Kepler Custom Fields, version 1.0 and above.

> Step-by-step guides, previews, demo images and screenshots were made under JIRA 6.x.

Table of Contents

Introduction

Kepler Custom Fields

The plugin offers a suite of custom fields that we think should be part of every JIRA installation.

These will improve the user experience and will allow you to customize JIRA to better suit your needs.

Included custom fields:

- Interval Custom Field
- SIL Script Custom Field
- Regex Custom Field

Requirements

A fully installed Kepler Custom Fields consists of multiple jar files. The simplest way is to use the bundle installer when installing it. Please refer to the Install Guide for explanations and details.

Here is the list of available versions:

Version	JIRA	katl-commons	klogger
1.0	4.3.x, 4.4.x	1.1.9	-
1.1	4.3.x, 4.4.x	1.1.10	-
1.1.1	4.3.x, 4.4.x	1.1.12	1.0
1.1.2	4.3.x, 4.4.x	1.1.13	1.0
1.1.3	4.3.x, 4.4.x	1.1.15	1.0
1.1.4	4.3.x, 4.4.x	1.1.17	1.0
1.1.5	4.3.x, 4.4.x	1.1.18	1.0
2.0	5.0.x	2.0.2	-
2.1	5.0.x	2.0.3	-
2.1.1	5.0.x	2.0.5	1.0
2.1.2	5.0.x, 5.1.x	2.0.6	1.0
2.1.3	5.0.x, 5.1.x	2.0.8	1.0
2.1.4	5.0.x - 5.2.x	2.0.10	1.0
2.1.5	5.0.x - 5.2.x	2.5	1.0
3.0.0	6.x	3.0.0	1.0
3.0.1	6.x	3.0.4	1.0
3.0.2	6.x	3.0.5	1.0
3.0.3	6.x	3.0.7	1.0
3.1.0	7.x	3.1.0	1.0

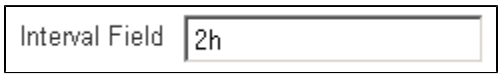
Interval Custom Field

How to configure

Go to the Administration page, choose Custom Fields link and add an interval custom field.

How to use it

Go to an issue and try to edit the field:



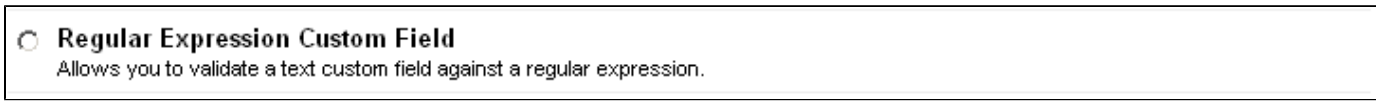
Save the changes and value should appear on the issue.

If the interval does not have the correct format (e.g. 2h 1d 3w), an error message should appear and the changes are not saved.

Regular Expression Custom Field

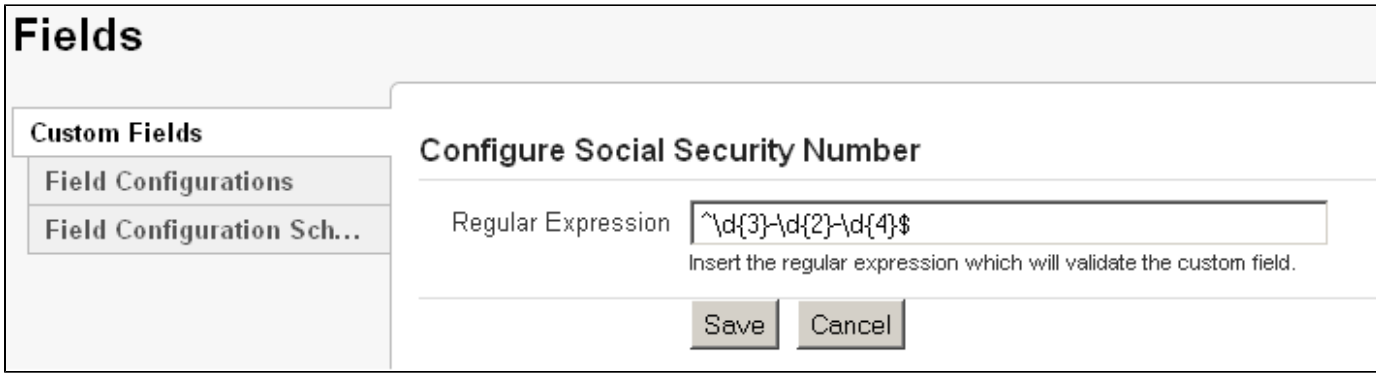
How to configure

Go to the **Administration** page, choose **Custom Fields** link and add regular expression custom fields.

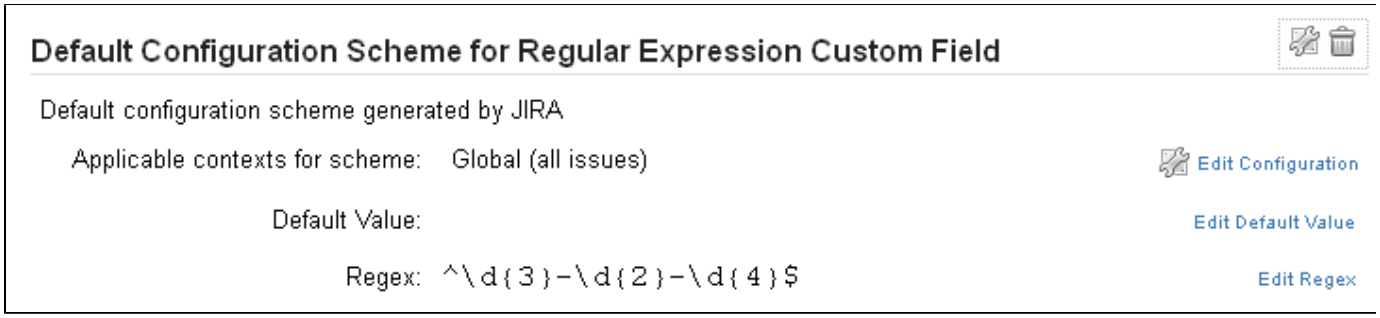


After you create the field, go to the corresponding **Configure** link. A page like in the image below will appear:

Click the **Edit Regex** link to configure the regular expression.

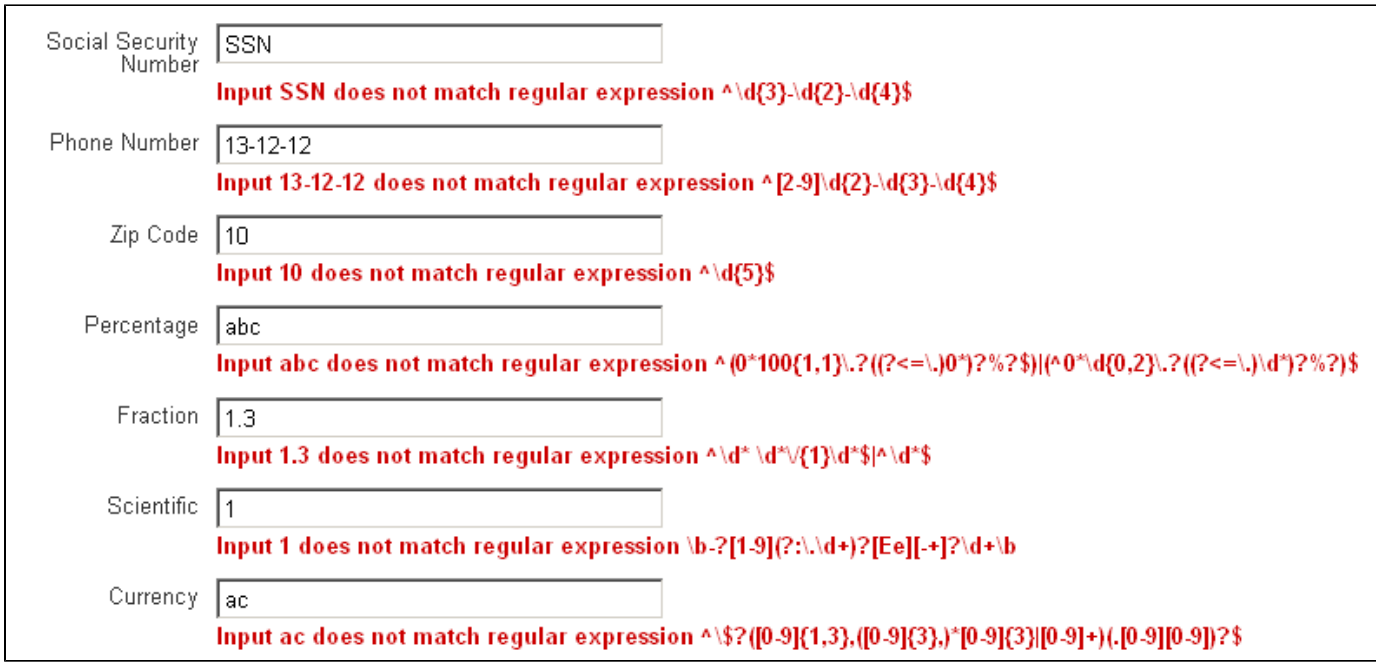


Click the **Save** button and the regular expression will be associated to the current custom field.



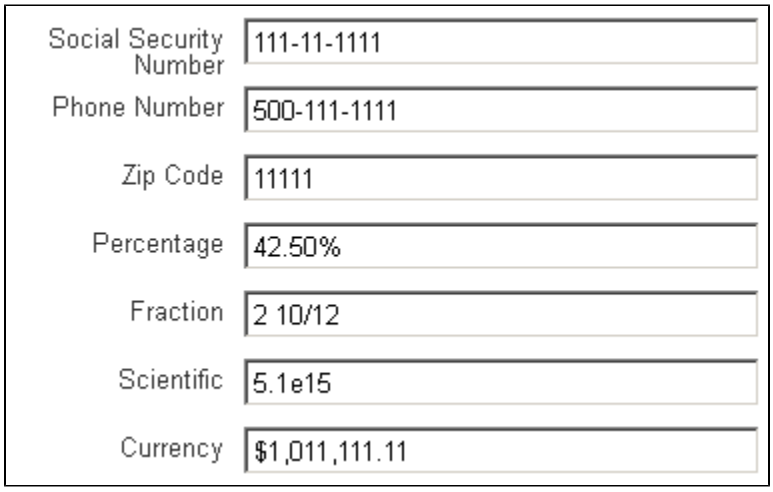
How to use it

Go to an issue and try to edit the field with a value that does not match the regex.



If you try to save the changes, an error message appears.

Otherwise, here are some usage examples for this type of custom field.



For more regex examples use the search from RegExLib.

That's it.

SIL Script Custom Field

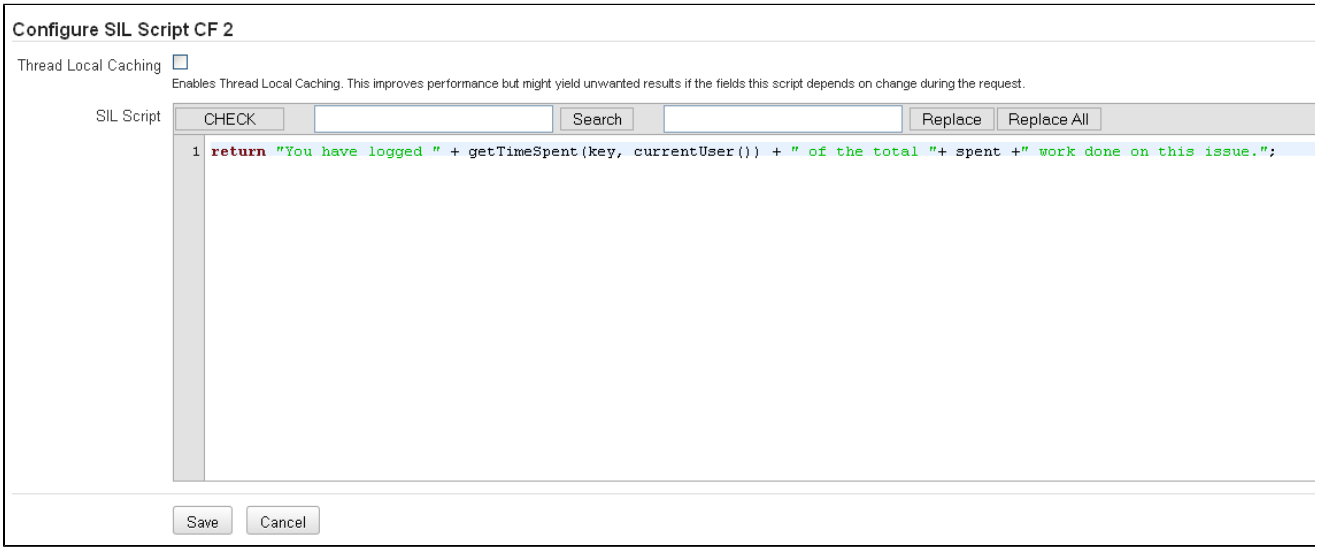
> Before using SIL Script Custom Field check out the Simple Issue Language documentation for a better grasp of SIL usage and capabilities.

How to configure

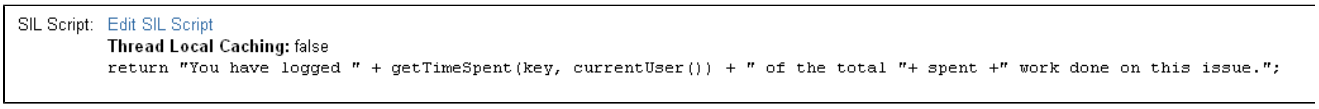
Go to the **Administration** page, choose **Custom Fields** link and add a SIL script custom field.

After you create the field, go to the corresponding **Configure** link. A page like in the image below will appear:

Click on the Edit SIL Script link to see the SIL editor.



Click on the **Save** button and the script is associated to the current custom field.



The value returned by the script will be the value of the custom field.

> The script is read-only. You must avoid changing any issue values in the script. In fact, you may change the issue variables BUT they **will not be changed** (issue will not be saved). However, routines may have side effects.

Thread Local Caching

> If you are not fully aware of what this implies, it is recommended that you leave the option off.

Because multiple calls to get the value of a SIL Script Custom Field for a certain issue are inevitable, we implemented an option to generate the value only once per HTTP request. This can improve performance, but might have some side effects.

For example, if the value is generated *before* some other values it depends on are modified, the result *might* not reflect latest updates**.**

How to use it

Go to an issue and this is how the value of the field is the value returned by the script.

SIL Script CF 2:	You have logged 12h of the total 13h 30m work done on this issue.

That's it.

Other Examples

Issue Age

```
interval age = currentDate() - created;
return "This issue is " + age + " old";
```

SIL Script CF 1:	This issue is 10w 1d 1h 5m 26s old

Number of subtasks

```
number noSubtasks = size(subtasks(key));
return "This issue has " + noSubtasks + " subtasks";
```

SIL Script CF 1:	This issue has 2 subtasks

Average Issue Age

```
date now = currentDate(); // just to make sure we use the same reference
date
string [] subtasks = subtasks(key);
interval age;
for(string task in subtasks){
 age = age + (now - %task%.created);
}
return "Average age of subtasks is " + (age / size(subtasks));
```

SIL Script CF 1:	Average age of subtasks is 8m 48s

Results from Database

```
//select the city from the specified region
return sql("TestSQL", "select city from cities c, district d, region r
where c.iddistrict=d.id and d.idregion=r.id and r.region='Bucharest'");
```

City from Region:	Bucharest

Status Age

```
string field_name = "status";
string[] field_history = fieldHistory(key, field_name);
number n = arraySize(field_history);
date startDate;
if (n > 0) {
 startDate = arrayGetElement(field_history, n - 2);
 return "Current issue has been " + status + " for " + (currentDate() -
startDate);
}
return "Current issue has been " + status + " for " + (currentDate() -
created);
```

Status:	Current issue is In Progress for 11m 56s

Issue Statuses Count

```
string[] statuses;
string[] statusHistory = fieldHistory(key, "status");
for(number i = 1; i < size(statusHistory); i = i + 2) {
 string statusStr = getElement(statusHistory, i);
 statuses = addElementIfNotExist(statuses, statusStr);
}
return size(statuses);
```

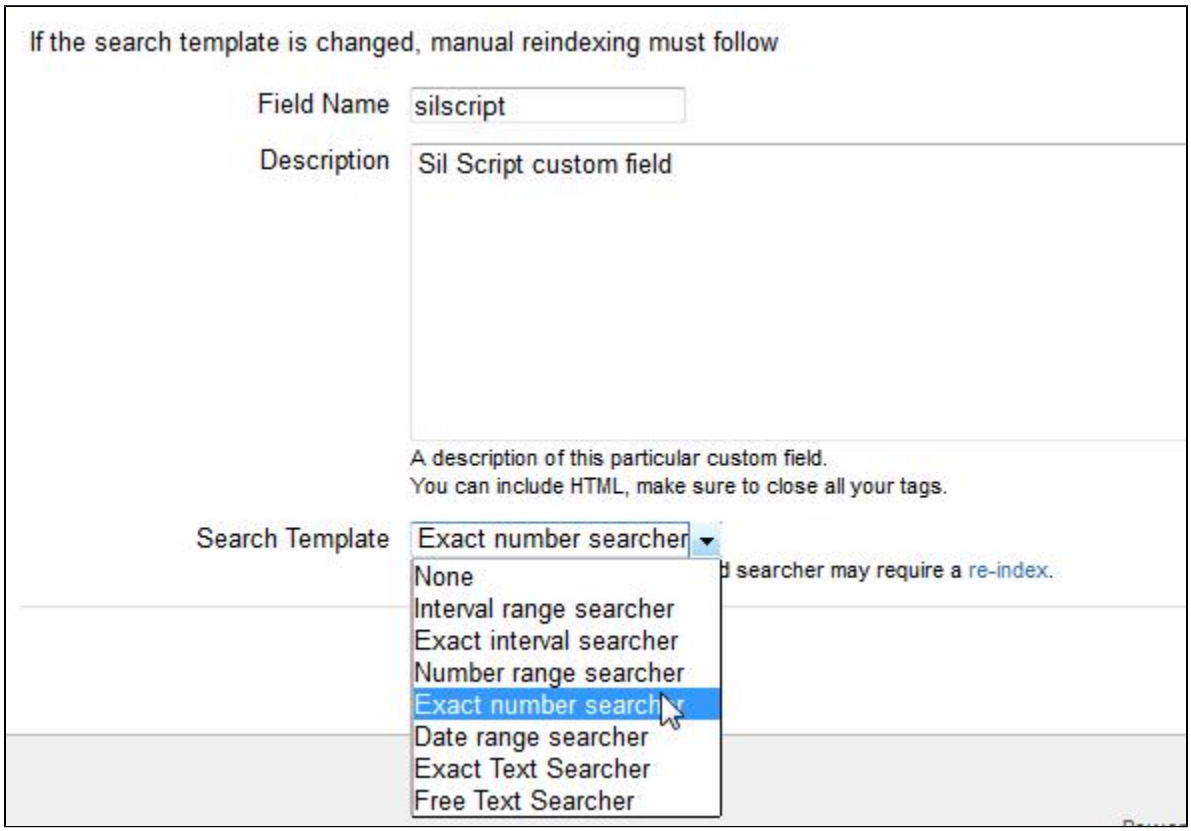
Issue Statuses Count:	4

Issue Assignees

```
string[] assignees;
assignees = addElementIfNotExist(assignees, userFullName(assignee));
string[] assigneeHistory = fieldHistory(key, "assignee");
for(number i = 1; i < size(assigneeHistory); i = i + 2) {
 string assigneeName = userFullName(getElement(assigneeHistory, i));
 assignees = addElementIfNotExist(assignees, assigneeName);
}
return assignees;
```

Search Issues

If you want a Sil Script Custom Field to be searchable you must select a Search Template for that custom field.



If the custom field already exists, in **Administration->Custom fields**, click **Edit** for the desired custom field, and choose the proper Search Template for your custom field according the value type returned by it.

If you are about to add a new Sil script custom field, you can choose the Search Template at the step **'Create Custom Field - Details (Step 2 of 2)'**.

> **Perform a re-index**
> After changing the Search Templates for all the custom field that you want, perform a re-index in JIRA for the search to work fine.

After a searcher has been set for the Sil Script custom field, you can perform a search for all issues containing the desired value for that custom field.

For a detailed explanation on searching issues in JIRA, you can check the Searching for Issues tutorial from the JIRA documentation.

Log Custom Field

How to configure

First of all, please download the klogger archive from our site: http://www.kepler-rominfo.com/static/downloads/com.keplerrominfo.jira.plugins.keplercf/resources/klogger-1.0.jar

The jar archive should be copied in the **<JIRAHome>/atlassian-jira/WEB-INF/lib** folder of your JIRA installation folder.

Log4j Configuration

Before adding a custom field in JIRA, please configure the **log4j.properties** file from: **<JIRAInstall>/atlassian-jira/WEB-INF/classes** folder and copy and paste the following code at the end of the file.

```
log4j.appender.klogger=com.keplerrominfo.jira.klogger.KeplerLogAppender
log4j.category.com.keplerrominfo = DEBUG, console, filelog, klogger
log4j.additivity.com.keplerrominfo = false
```

> **log4j.category.com.keplerrominfo and log4j.additivity.com.keplerrominfo** properties should appear only once in the log file. If you have already enabled logging, all you need to do is add the appender.

Custom Field

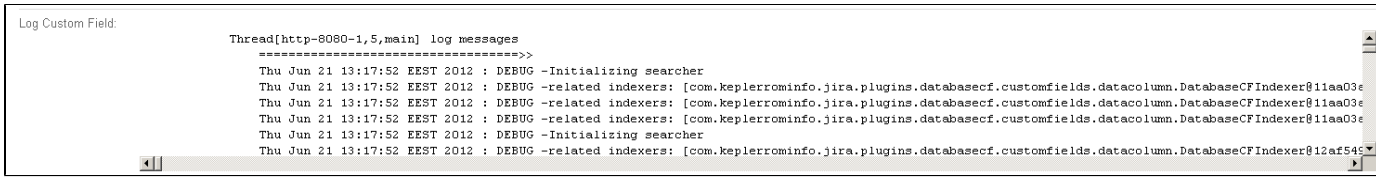
Go to the Administration page, choose Custom Fields link and add a log custom field.

Configure the field as any JIRA custom field.

How to use it

On any issue, log messages should be visible in the new added custom field. It will show messages on **thread** that is serving the current page showing the issue (last 512 messages). While our primary purpose was to offer a simple CF to show the log on the current issue, be warned that it is not advisable to enable it in production systems and that the memory consumption is quite heavy (for each thread we need to store the last 512 messages).

> The primary audience for this custom field was our SIL developers group. However, you can use it basically **for anything**, provided that your **Log4J** logger is configured to intercept those messages.



That's it.

Backup and restore

At Restore: install first the plugins

Mundane operations as backup and restore may pose some problems to the unsuspecting JIRA administrator. Since all the Kepler plugins create some tables in the JIRA schema - we created this mechanism long before Active Objects was introduced into Atlassian's framework - you need to take some precautions at restore.

Specifically, at restore you need to create the tables used by our plugins. You do not need to copy schema from the previous JIRA or fill it with data, you just need to **simply install the plugins into JIRA before restoring** (enabling the plugins would create the needed tables).

Kepler Custom Fields has one dependency:

1. katl-commons (core support)

For reference, these are the tables created by each add-on

Plugin	Tables
katl-commons	kplugins
	kpluginscfg
	kissuestate
	kstatevalues