
















| | |
|--|----|
| 1. Home | 2 |
| 1.1 KIWI Documentation | 2 |
| 1.1.1 Kiwi Features | 4 |
| 1.1.2 Requirements | 6 |
| 1.1.3 Installation | 7 |
| 1.1.3.1 Install notes for JIRA 7 | 7 |
| 1.1.3.1.1 What should I do if I installed an incompatible version? | 7 |
| 1.1.3.2 Installation via Atlassian Universal Plugin Manager | 8 |
| 1.1.3.3 Manual Install | 8 |
| 1.1.3.4 Licensing | 8 |
| 1.1.3.5 Uninstall | 9 |
| 1.1.3.5.1 Manual Uninstall | 10 |
| 1.1.3.5.2 Uninstall via Atlassian Universal Plugin Manager | 10 |
| 1.1.4 User Guide | 11 |
| 1.1.4.1 Export Workflows | 12 |
| 1.1.4.2 Import Workflows | 14 |
| 1.1.4.2.1 Custom Fields Mapping | 21 |
| 1.1.4.2.2 A Note About SIL Aliases | 23 |
| 1.1.4.2.3 Import Options | 23 |
| 1.1.4.3 KIWI Tools | 25 |
| 1.1.4.3.1 Merging .cfmap files | 25 |
| 1.1.4.3.2 Merging sil.aliases files | 26 |
| 1.1.4.4 Supported Custom Fields | 27 |
| 1.1.5 Backup and restore | 27 |

Home

This is the home of the KIWI space.

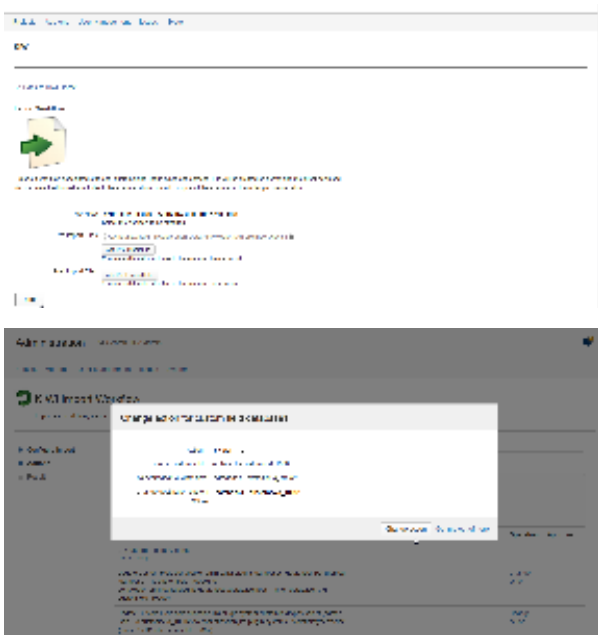
To help you on your way, we've inserted some of our favourite macros on this home page. As you start creating pages, blogging and commenting you'll see the macros below fill up with all the activity in your space.

Recently Updated

-  [KIWI30](#)
Feb 28, 2017 • attached by [Alexandru Geageac](#)
-  [KIWI 3.0](#)
Feb 27, 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 [Alexandru Geageac](#)
-  [Requirements](#)
Nov 16, 2015 • created by [Alexandru Geageac](#)
-  [Backup and restore](#)
Feb 19, 2015 • created by [Alexandru Geageac](#)
-  [Custom Fields Mapping](#)
Sep 15, 2014 • created by [Maria Cirtog](#)
-  [kiwiCfld.png](#)
Sep 15, 2014 • attached by [Maria Cirtog](#)
-  [Import Workflows](#)
Sep 15, 2014 • created by [Maria Cirtog](#)
-  [Export Workflows](#)
Sep 15, 2014 • created by [Maria Cirtog](#)
-  [Uninstall via Atlassian Universal Plugin Manager](#)
Sep 15, 2014 • created by [Maria Cirtog](#)
-  [kiwiUninstall3.png](#)
Sep 15, 2014 • attached by [Maria Cirtog](#)
-  [kiwiUninstall2.png](#)
Sep 15, 2014 • attached by [Maria Cirtog](#)
-  [kiwiUninstall.png](#)
Sep 15, 2014 • attached by [Maria Cirtog](#)
-  [plugins admin menu.PNG](#)
Sep 15, 2014 • attached by [Maria Cirtog](#)

Navigate space

KIWI Documentation



Features

Have you ever been in the situation when you built everything on your development server, everything goes smooth, then going with your solution into the production you got to re-do much of your steps, piece by piece, custom field with custom field? Do you know the pain? Remember how you wrote down each step on some big documents? Remember when you forgot to document the change from the very beginning, and you got it wrong? Remember the mistakes you have made? Remember the look on your customer's face? Do you remember their questions? "How can it be - it's just a deployment? Why is taking it so long? Will I be able to use my production JIRA on Monday?"

Well, now you're saved, because there's KIWI to help!

KIWI is a JIRA workflow implementation deployment tool and it addresses initial configuration as well as subsequent deployments, on staging or production environments.

It delivers issue statuses, custom fields, workflows, associated screens, SIL scripts and configurations on your target system, having the ability to map custom fields, change the scripts, SIL aliases and configuration so no change is needed on the deployment machine: your implementation will be ready to go with just a few clicks!

Note: some of you might know KIWI by its former name: Sunda2Sahul (http://en.wikipedia.org/wiki/Sahul_Shelf).

Kiwi Features

Kiwi is the tool you'll need to export or import your workflows in just a few seconds. Designed to satisfy many customers requests, this plugin offers you the following features:

Exports/imports an workflow

During the import, Kiwi exports and restores the selected workflow, carrying out all its steps, statuses and transitions.

The **statuses** used in the workflow are carried out at export and created or updated on the Jira Import instance at import. On the destination server, the mapping of statuses is done by name.

For each status exported, if on the Jira import instance there is a status having the same name with an exported one, this existing status will be updated to match the exported one, otherwise the status will be created.

The **transitions** used in the workflow are exported and imported together with their conditions, validators and postfunctions.

Carries SIL scripts with the workflow

If you are using our JJUPIN plugin and you have created **SIL scripts** (validators, conditions or post-functions) on the transitions of the exported workflow, these SIL scripts will be exported and restored on the Jira Import server.

At the import time, if on the destination server there is a SIL script file having the same name and path with a sil script from the kiwi file that is being imported, the existing sil file will be overwritten. You will receive a warning related to this in the Import page, at Step 2:Actions for the action 'Create or update the SIL files' (the action can be found and also disabled in the section General actions).

Exports/imports Issue type screen schemes

Kiwi exports/imports the Issue type screen schemes **related** to the selected workflow.

What does related mean?

Well, an Issue type screen scheme will be exported if it is used as a Screen Scheme in at least a project that associates to the current workflow.

The reason of exporting this was to recreate on the destination server the Create/Edit/View screens that can be associated to a project that is related to the current workflow.

As you know from JIRA's documentation, there are some associations like this:

Project -> Issue Type Screen Scheme -> [Issue Type, Screen Scheme].

And each Screen Scheme contains the desired screens: Create/View/Edit.

[Issue Type, Screen Scheme] represents a mapping between the issue type and the Screen Scheme.

For instance the issue type `Bug` can have an associated Screen Scheme, therefore its specific Create/View/Edit screens that can be different to issue type `Task` and its specific Screen Scheme.

To be more specific, an Issue type screen scheme (ITSS) contains one or more associations between an issue type and a screen scheme, so our plugin exports those screen schemes that are present in the exported ITSS.

Screen schemes associates the issue operations to screens. These screens are also exported/imported by KIWI, together with the screens used in the workflow for transitions.

Issue type screen schemes and Screen schemes are mapped by name.

If on the Jira Import instance there is found an Issue type screen scheme with the same name, it will be updated to match the exported one. So it will have the same description and [issue type, screen scheme] associations. Otherwise the Issue type screen scheme will be created.

Exports/imports workflow related screens

KIWI exports and imports all the screens used directly in the selected workflow (screens that are configured as Transition View in the workflow transitions), and also all the screens from the [Issue type screen schemes](#) the workflow is related to.

The screens are mapped by name at the import time. Every screen that is exported is analyzed, and if the screen exists on the Jira Import instance (if it is found by name), it will be updated to match the screen from the exported file (replace all the entities, i.e tabs and custom fields of the updated screen with the entities of the exported screen - we do not merge the two screens), otherwise the screen will be created.

Exports/imports workflow related issue types

Since version 1.0.1-beta3, KIWI exports all the issue types from the **contexts of related custom fields** and that are found in the related **Issue type screen schemes**.

In the import page, you can choose what issue types should be imported. Only custom fields that have at least one of the selected issue types associated in their context or that have global context, and only the relevant issue type screen schemes associations for the selected issue types will be imported.

Issue Types are mapped by name at import. Every exported issue type that is selected for import is analyzed, and if it exists on the Jira Import instance (if it is found by name), it will be updated to match the issue type from the exported file, otherwise the issue type will be created.

Exports/imports workflow related custom fields

KIWI exports/imports all the custom fields from the screens related to the workflow (e.g. **transition screens** and the screens that are found in the related **Issue type screen schemes**).

To be more specific, Kiwi also exports the custom fields that are not associated directly to the workflow. Kiwi also carries out all the custom fields that are specific to the Create/Edit/View screens only in the ITSS and not in the workflow.

These custom fields are exported along with their contexts - Issue Types and Projects.

The projects used in the contexts must exist on the destination server(the search is done by project key). Otherwise, a project mapping may be configured at Import, Step 1, Import Options section. Otherwise the contexts are partially created/updated using the existing projects, if at least one project for the context is found.

During the import time, every custom field (let's name it **exported cf**) that is present in one of the exported screens is mapped by using the following algorithm:

1. at first KIWI looks for a mapping/match (done using the Change action link from the Import page-Step 2 or from the mapping file)
2. if there is a mapping/match defined in the mapping file, it will get the custom field from the Jira Import instance.(The existing custom field is updated to match the exported custom field).
3. otherwise if it is found a custom field that has the same name and the same type, then this existing custom field will be updated to match the exported custom field
4. otherwise the custom field will be created

When a custom field is updated, the name, description, the context(s) and the searcher are updated.

If you are not satisfied about how custom fields are mapped, and you don't want to edit the mapping file, KIWI offers you the possibility to change the custom field actions directly from the [import page](#).

In a few words, you can choose to update a custom field by selecting the update operation and the desired destination for the proper custom field, or you can decide to create a new custom field. It's up to you.

Updating or creating a custom field involves carrying over its name, description, searcher and contexts(For each context the context name, description, issue types, projects, default value, and options -if any, e.g. for the Select List/Radio Buttons are carried over). Also, for the [Kepler provided custom fields](#), the special configurations are exported/imported by Kiwi.

Generates .cfmap files

A successful import will generate two mapping files(the current and the next mapping file) in the **JIRA_HOME_FOLDER/kepler/kiwi** folder.

The **next mapping file** is named using the following pattern: <imported_workflow_name>_next_<timestamp>.cfmap. It contains all the mappings resulted from the import(according to your used .cfmap file and according to your screen selection), including the created custom fields.For an update custom field action, a new entry with the match/mapping [the source custom field -> the destination custom field] is added to the file. For a create custom field action, the entry will contain the source custom field id mapped on the newly created custom field id.

The **current mapping file** is named using the following pattern: <imported_workflow_name>_current_<timestamp>.cfmap. It contains all the mappings(according to your used .cfmap file and according to your screen selection) for the custom fields existing on the Import server. For an update custom field action, a new entry with the match/mapping [the source custom field -> the destination custom field] is added to the file. Nothing is added to this file for a create custom field action.

For both mapping files names <timestamp> represents the date at which the file is generated, in the format yyyyMMddHHmm.

Further readings about **custom field mapping file** can be found [here](#).

Generates a SIL aliases file (for JJUPIN users)

If you are using our **JJUPIN** plugin (the **SIL** provider) you may have notice that there is a **sil.aliases** file in the Kepler folder of your Jira Home folder.

This file maps the custom fields onto a more easy readable names that can be used in the SIL scripts.

The export operation encapsulates the entire file form the source server, including the comment lines.

A successful import will generate this file in the Kiwi home folder with the name: <imported_workflow_name>_sil_<timestamp>.aliases, where <timestamp> represents the date at which the file is generated, in the format yyyyMMddHHmm. (For instance, the file Workflowfortest_sil_201310071330.aliases is the sil.aliases file generated when importing the workflow Workflowfortest, on 07.10.2013, 13:30). This file will contain **only those custom field aliases that were imported, re-mapped according to your .cfmap file and according to your screen selection**.

Offers tools to merge CF mapping files and sil.aliases files

Kiwi offers the [tools](#) to merge the generated .cfmap and sil.aliases files. You can keep the sil.aliases file up-to-date or create mappings that can be reused.

Executes SIL scripts at the beginning and in the end of the workflow

From the [Export Workflow](#) page, you can add to your exported workflow a list of pre-import and post-import sil files that will be executed on the Import Jira instance, before(the pre-import files) or after(the post-import files) executing the main import.

Important notices

- For custom fields, in order to fully restore the contexts, the projects (the search is done by project key) should exist on the Jira Import instance before import or a mapping must be defined for a missing project in the [Import Options](#) section. Otherwise, the contexts are partially created/updated using the existing projects, if at least one project for the context is found.
- KIWI doesn't export/import things related to a project, like Workflow Schemes, Field Configuration Schemes, Field Configurations and Issue Type Schemes. If needed, these must be manually created and associated to projects. The [Issue type screen schemes](#) are carried out by KIWI, but the association to the projects must be done manually after the import

Requirements

A fully installed KIWI plugin consists of multiple jar files. You are advised to use the bundle installer when installing KIWI. Please refer to the [Install Guide](#) for explanations and details.

At the minimal level KIWI consists from 2 dependencies (jar files): katl-commons (a library having countless utility routines, but also - most important - the SIL language parser) and a KIWI jar file, which contains specific tools to export or import your workflows.

Compatibility

| KIWI | JIRA | katl-commons |
|-------|--------------|--------------|
| 3.0.0 | 6.0 - 6.4.12 | 3.0.0 |
| 3.0.1 | 6.0 - 6.4.12 | 3.0.7 |
| 3.1.0 | 7.0.0 | 3.1.0 |

Installation

- [Install notes for JIRA 7](#)
 - [What should I do if I installed an incompatible version?](#)
- [Installation via Atlassian Universal Plugin Manager](#)
- [Manual Install](#)
- [Licensing](#)
- [Uninstall](#)
 - [Manual Uninstall](#)
 - [Uninstall via Atlassian Universal Plugin Manager](#)

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.

Please note that the kiwi plugin must be installed on both the export and the import Jira instances.

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 KIWI 3.0.x on JIRA 7.x or KIWI 3.1.x on JIRA 6.x, you should do the next steps :

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

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

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 **kiwi** plugin and install it. Installing the obr will bring also all required dependencies: **katl-commons** and **warden**.

That's all.

Manual Install

Manual Install

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

1. Download the correct kiwi 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 `JIRA_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
```

Licensing

Dual Licensing support

KIWI plugin supports both Kepler and Atlassian licenses, but you only need one valid license to run the plugin, which can either be provided as the **kiwi.lic** file, or as the key generated via the [Atlassian Marketplace](#).

The order in which the licenses are checked is:

1. Atlassian License
2. Kepler License

It is **strongly recommended** that you do not install both licenses at once, as this might yield unwanted results. For example, consider that you have an Atlassian License with the support date expired and one valid Kepler License. In this case you cannot update the plugin, because the Atlassian License is checked first and its support date is expired.

Technical info

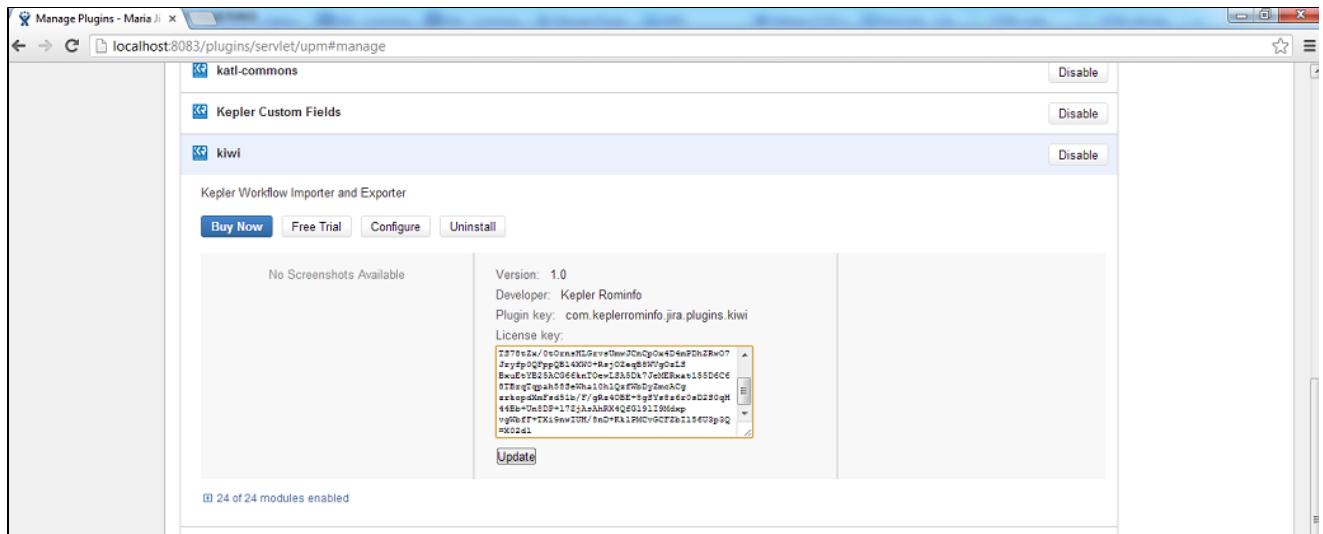
Starting with **katl-commons version 2.5.5** a new plugin, called **Warden**, will be automatically installed by any paid add-on (including KIWI). This plugin is responsible with the management of licenses (both JIRA and Kepler). Do not attempt to uninstall it without removing first all the Kepler paid add-ons.

Atlassian Licensing

The Atlassian Marketplace allows you to easily purchase or generate an evaluation license for **KIWI**.

Using Universal Plugin Manager 2.0.1+

After generating the license key, all you have to do is access the **Administration-> Add-ons-> Manage add-ons** section in your JIRA instance and paste the key into the corresponding plugin textbox.



Kepler Licensing

The Kepler license is a file (**kiwi.lic**) which must be placed in the <JIRA_HOME>/kepler folder. You can either generate and download a free evaluation license by registering on our site and accessing the **Licenses** section, or you can purchase the plugin by following [these instructions](#).

You can view details for your kepler license by accessing the **Kepler Licenses** page from **Add-ons >KEPLER PLUGINS CONFIGURATION> Kepler Licenses** menu:

Kepler Licenses

Here you can inspect all license files from your kepler home directory (C:\installs\jira\6_3\Application Data\JIRA\kepler).

License
Select a license file to see its details.

License Information

| | |
|------------------|-----------|
| Expiration Date | 01/Nov/14 |
| Maintenance Date | 30/Sep/14 |
| User Limit | Unlimited |
| Valid | YES |

The page shows the expiration and maintenance date, user limit and validity message for each selected kepler license.

If the license is expired, user limit is exceeded or license is targeted for a different JIRA server id, a red colored message shows the status.

If kepler license is close to expiration date (less than 10 days remaining), a warning message is displayed, showing the remaining time.

Reminder

Don't forget that you need only *one* valid license to run the plugin.

Removing an unused license

If you want to remove a no longer used Atlassian license, this can be done in UPM (for UPM 2.0.1+) , by removing the old license key and clicking Update. To remove a Kepler license, you have to delete the correspondent .lic file from the kepler folder. Note that any change to the Kepler license requires a server restart.

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 via Atlassian Universal Plugin Manager](#)

Manual Uninstall

Uninstall manually

At first we will uninstall the plugin manually and finally we'll remove the corresponding tables in the internal database.

Uninstall the plugin

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 KIWI plugin

Restart the server

Now you can restart Jira server

Uninstall via Atlassian Universal Plugin Manager

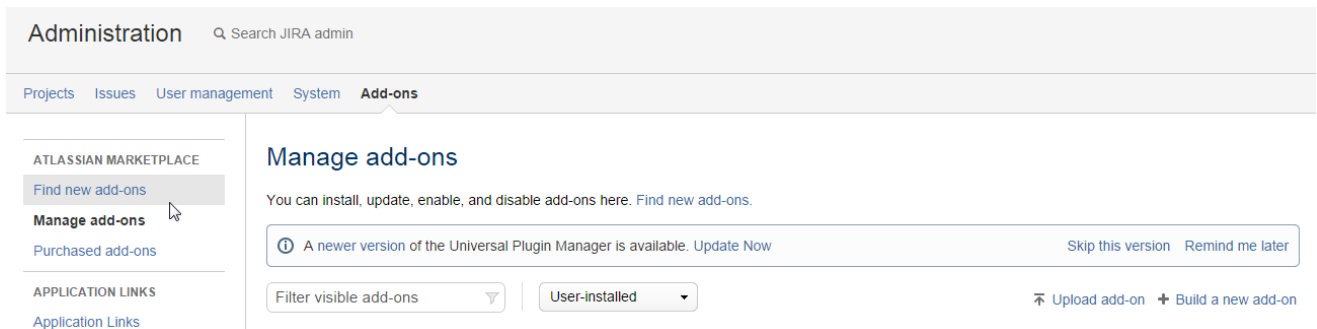
Uninstall via Atlassian Universal Plugin Manager

At first we will uninstall the plugin and finally we'll remove the corresponding tables in the internal database.

Uninstall the plugin

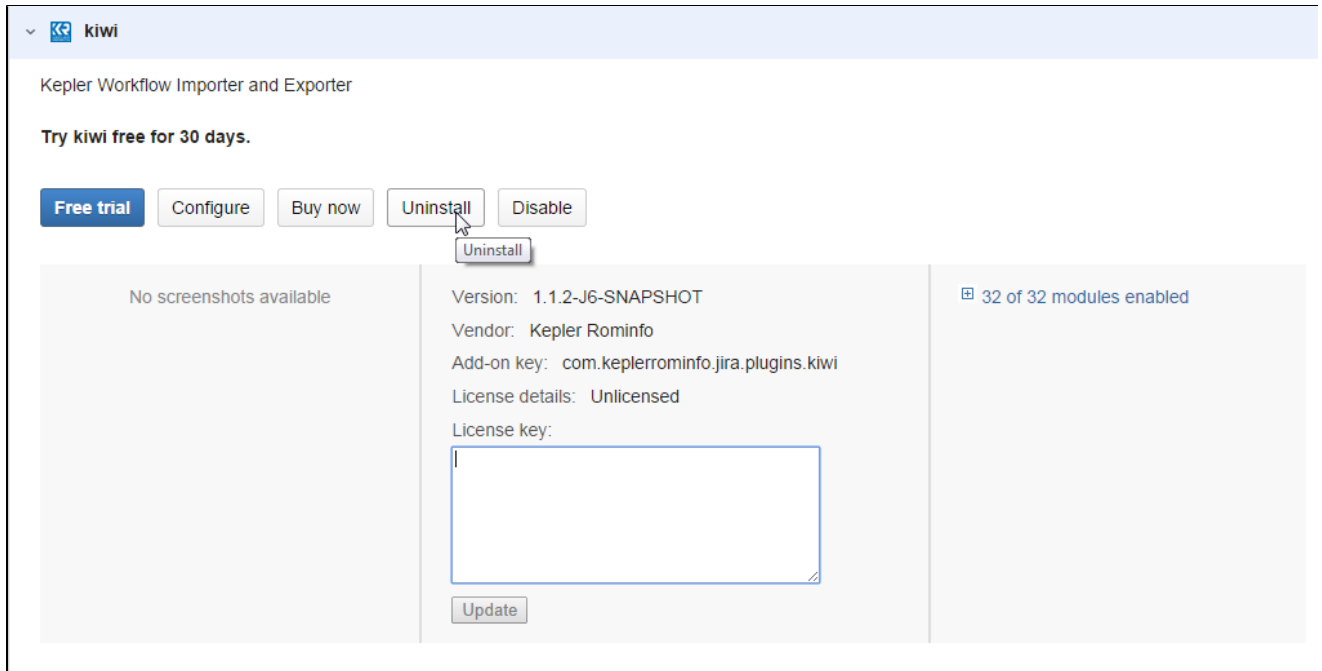
If you are not familiar with Universal Plugin Manager (UPM), please read [this document](#) before we begin.

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

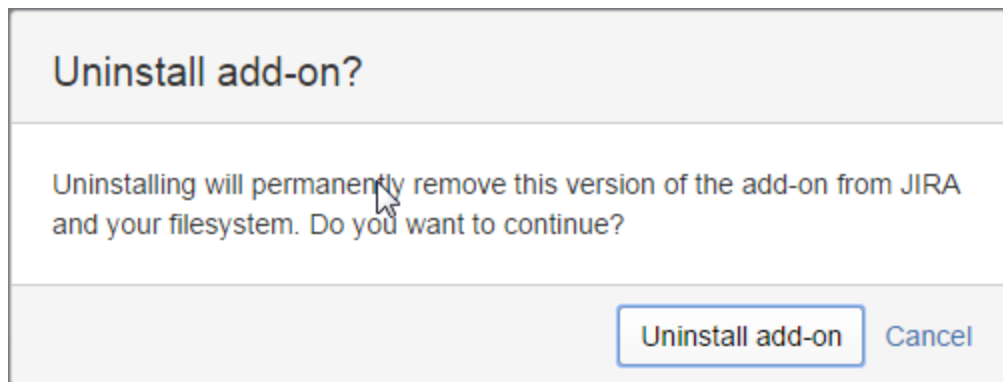


The screenshot shows the Atlassian Administration console. The top navigation bar includes 'Administration' and a search bar. Below the navigation bar, there are tabs for 'Projects', 'Issues', 'User management', 'System', and 'Add-ons'. The 'Add-ons' tab is selected. On the left sidebar, under 'ATLASSIAN MARKETPLACE', the 'Manage add-ons' option is highlighted. The main content area is titled 'Manage add-ons' and contains a notification about a newer version of the Universal Plugin Manager being available. Below the notification, there are filters for 'Filter visible add-ons' and 'User-installed'. At the bottom right, there are links for 'Upload add-on' and 'Build a new add-on'.

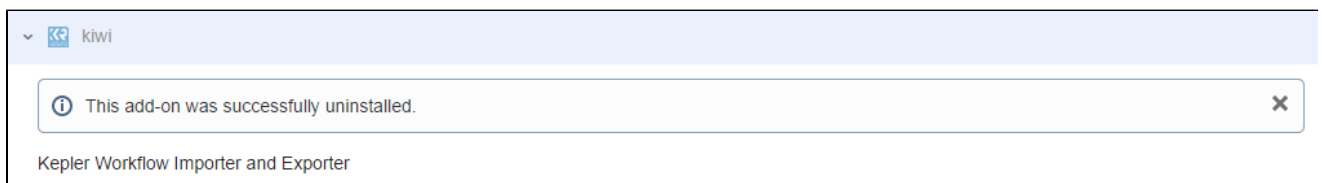
- 2) Search for 'Kiwi' plugin in 'Manage add-ons' section and click on 'Uninstall' button



3) Press `Continue` when the uninstall confirmation dialog box appears



4) A message "successfully uninstalled" should appear



User Guide

- [Export Workflows](#)
- [Import Workflows](#)
 - [Custom Fields Mapping](#)
 - [A Note About SIL Aliases](#)
 - [Import Options](#)
- [KIWI Tools](#)
 - [Merging .cfmap files](#)
 - [Merging sil.aliases files](#)
- [Supported Custom Fields](#)

In this section, you will learn about the friendly user interface that KIWI offers and its capabilities.

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

The typical scenario for KIWI is to:

- export a workflow from a Jira instance(let's name it Export Jira instance)
- copy the file obtained by exporting the workflow in <JIRA_HOME>/kepler/kiwi directory on the other Jira instance(the server where you execute the import - let's name it Import Jira instance) by normal means(ssh, copy, etc)
- import the kiwi file on the Import Jira instance

Export Workflows

Kiwi offers you the possibility to move complex workflows from a JIRA system to another just with the click of your mouse.

In order to do a complete import, your first(and mandatory) step is to export a workflow.

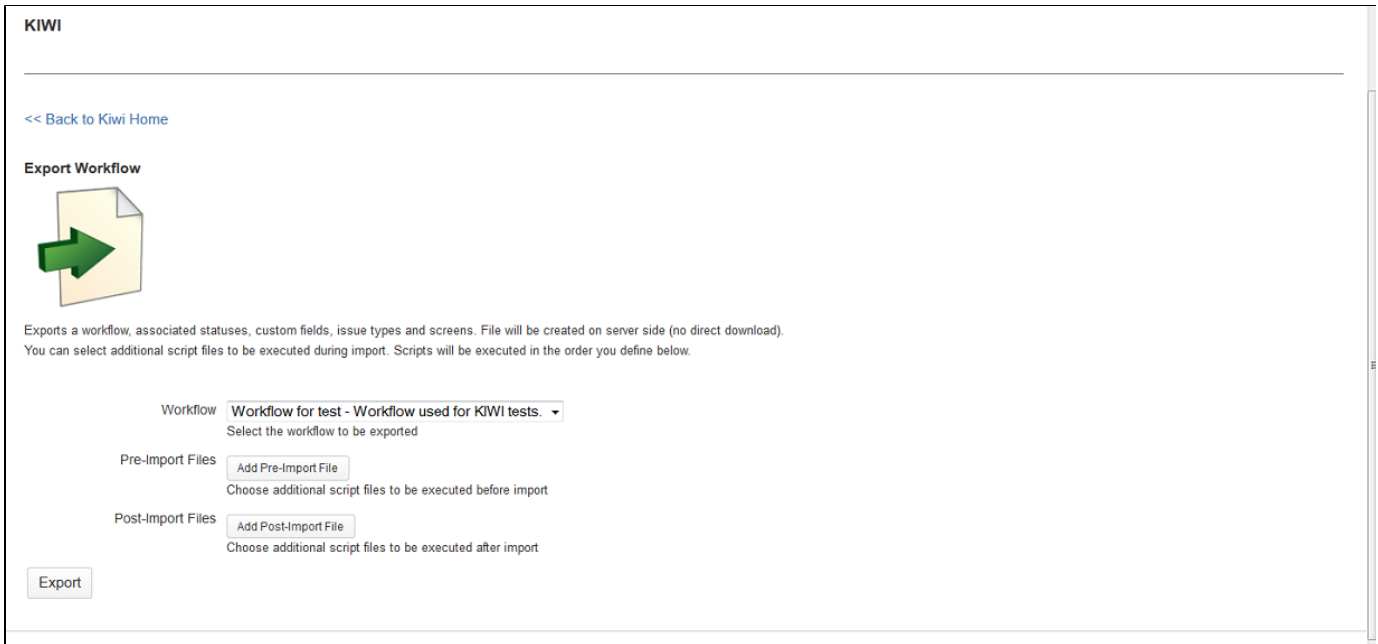
This can be done from the Export page that is accessible from the main KIWI page, which can be found in the Administration view, Add-ons->KIWI.

The screenshot shows the JIRA Administration interface. At the top, there is a search bar for 'Search JIRA admin' and a navigation menu with 'Projects', 'Add-ons', 'User Management', 'Issues', and 'System'. The 'Add-ons' section is expanded, showing a sidebar with categories like 'ATLASSIAN MARKETPLACE', 'APPLICATION LINKS', 'SOURCE CONTROL', 'ISSUE COLLECTORS', 'MONITORING', 'ADMIN HELPER', and 'KEPLER TOOLS'. The 'KIWI' section is selected, displaying two main options: 'Export Workflow' and 'Import Workflow'. The 'Export Workflow' option features a document icon with a green arrow pointing right and an 'Export' button. Below it, text explains that it exports workflows, statuses, custom fields, and screens to the server side. The 'Import Workflow' option features a circular green arrow icon and an 'Import' button. Below it, text explains that it imports workflows, statuses, custom fields, and screens, and provides instructions on where to place the exported file. A red warning box at the bottom right of the KIWI section states: 'Make sure you perform a backup first, prior to importing the workflow. You may completely mess up the current installation of JIRA !'. Underneath the main options, there is a 'Utilities' section with two buttons: 'Merge Aliases' and 'Merge CF Maps', each with a brief description of their function.

Here you will find a combo-box from which you can select the workflow you want to export.

Default Workflow

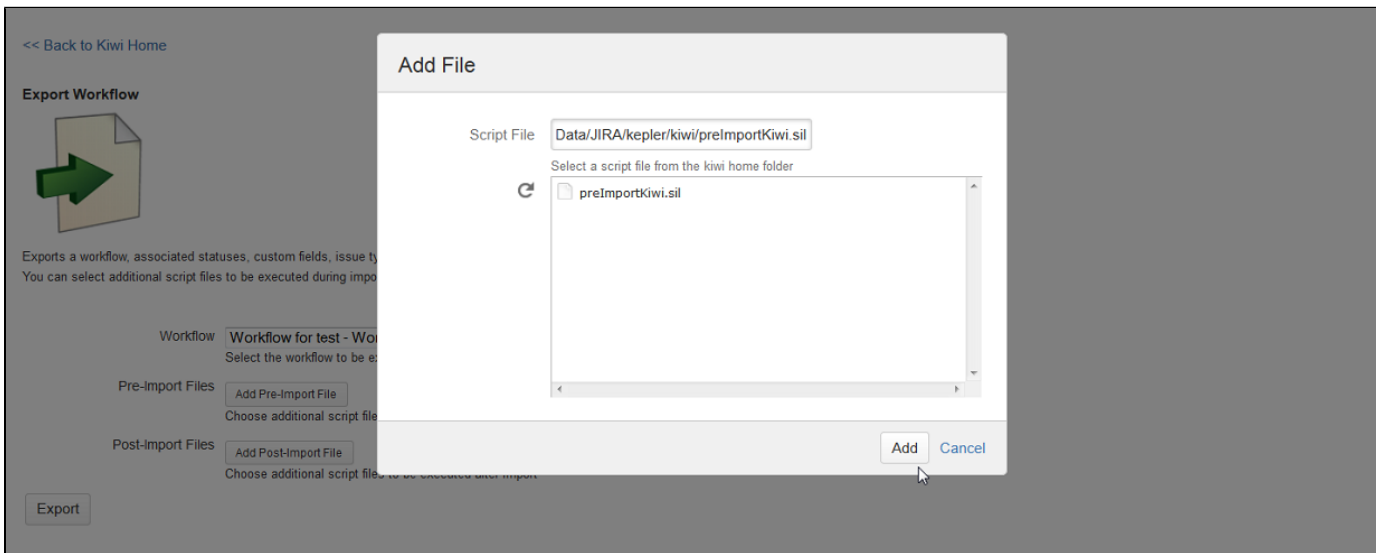
The workflow "The default JIRA workflow" can not be exported because this is a read only workflow.



As the description on the page informs, you are about to export a JIRA Workflow with all its dependencies (associated statuses, issue types, custom fields and screens).

You can add to your exported workflow a list of pre-import and post-import sil files that will be executed on the Import Jira instance, before (the pre-import files) or after (the post-import files) executing the main import. You definitely know about sil script files if you are using our **JJUPIN** plugin. **SIL**, or **Simple Issue Language** is very easy to learn yet powerful and extensible: it's a Java-like language independent of the JIRA version, offering power through simplicity and flexibility.

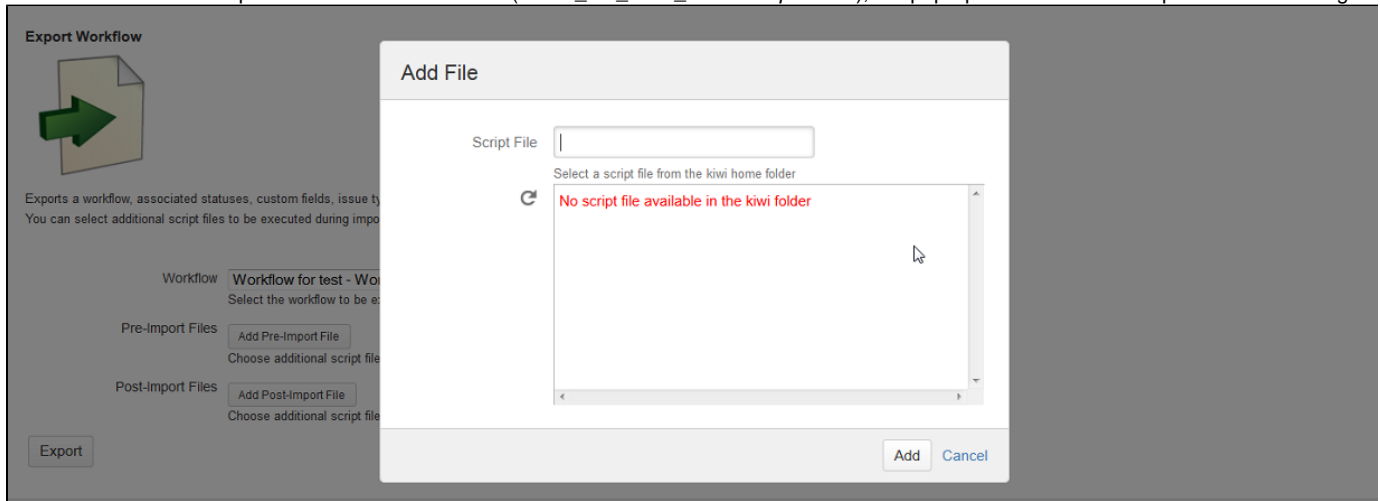
In order to add a pre-import file, press the 'Add Pre-Import File' button that will open a pop-up dialog where you can choose a sil file from the kiwi home folder ("PATH_TO_JIRA_HOME/kepler/kiwi"):



Press the Add button from the pop-up, and you will see the chosen script file above the 'Add Pre-Import File' button. The script will be added to the exported kiwi file when pressing the Export button.



If there is no sil script file in the Kiwi home folder (*PATH_TO_JIRA_HOME/kepler/kiwi*), the pop-up will show a 'No script file' error message:



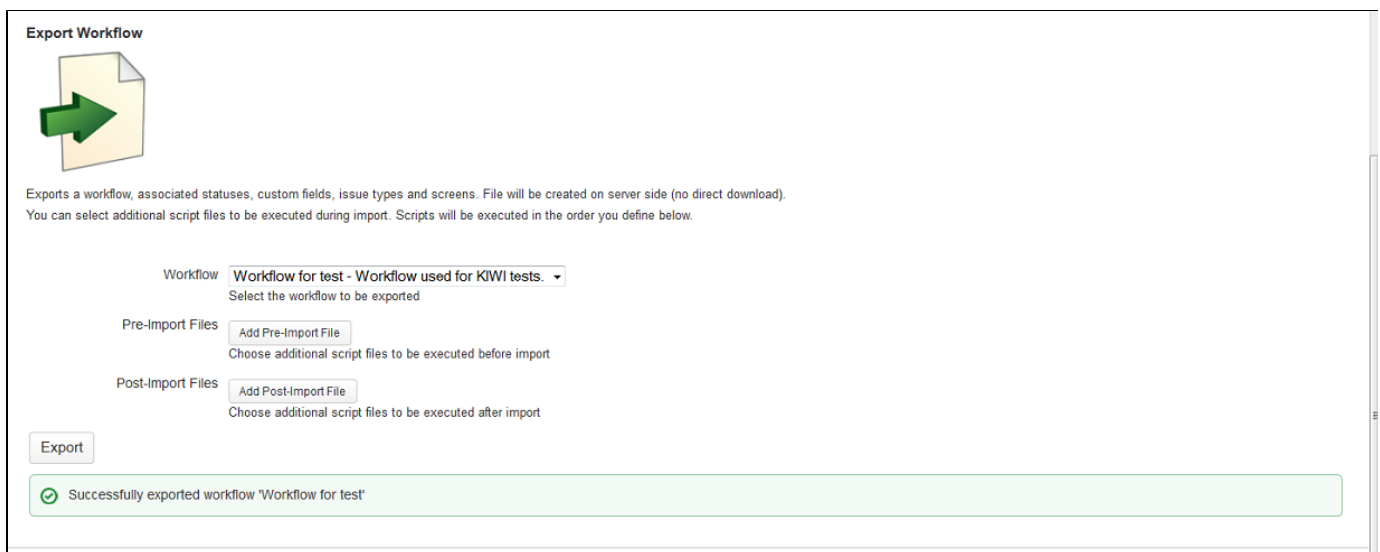
Use the 'Add Post-Import file' button and the same procedure for adding a post-import sil file.

SIL scripts

NOTE: if you are using our JJUPIN plugin and you have created SIL scripts (validators, conditions or post-functions) on the transitions of the exported workflow, these SIL scripts will be exported too.

Press the Export button.

If everything works as planned, you'll see a SUCCESS message on the screen. The exported file can be found in the **KIWI home folder** (this can be found at "*PATH_TO_JIRA_HOME/kepler/kiwi*"). Here you will find a new zip file named after your exported workflow (the name is a concatenation of the exported workflow name, the current date in the format *yyyyMMddHHmm* and *.kwi*) for instance, *Workflowfortest_201310071619.kwi.zip* is the file obtained by exporting the workflow *Workflowfortest* at local time 07.10.2013, 16:19.



After this operation you have a brand new **KIWI** file that you should copy to your JIRA system where you'll do the import. Now you've just left **Sunda**.

Import Workflows

KIWI allows you to import a workflow together with its associated statuses, screens, issue type screen schemes, custom fields and SIL scripts.

Make sure you perform a backup first, prior to importing the workflow. You may completely mess up the current installation of JIRA!

The import page is accessible from the main KIWI page, that can be found in the Administration view, Add-ons-> KIWI:

In order to import a workflow, the file obtained by exporting the workflow must be placed in <JIRA_HOME>/kepler/kiwi directory (on the server where you execute the import, of course) by normal means (ssh, copy, etc).

In the import page you must select the kiwi file to be imported (from the list of kiwi files present in your <JIRA_HOME>/kepler/kiwi directory).

Optionally you can select a mapping file, if you want to define your own [custom fields mapping](#). (Please be sure you know what you are doing in this case!)

If the Next button is pressed, but no workflow is selected, you will receive an error message:

KIWI Import Workflow
Imports a workflow, associated statuses, custom fields, issue types and screens.

- Configure Import**
- Actions
- Result

Step 1: Configure Import

Set up the configuration parameters for the workflow import.

Import File: -- Select a workflow file --
Select a kiwi workflow file to import from

Mapping File: -- Select a mapping file --
Select a mapping file for custom fields

[Next >>](#) [Back to home](#)

Please select a workflow before pressing the Next button!

After choosing an workflow, you can configure the settings for the import in the section **Import Options**.

Import File: Workflowfortest_201309271505.kwi.zip
Select a kiwi workflow file to import from

Mapping File: Workflowfortest_cfmap
Select a mapping file for custom fields

Import Options
click to collapse

Import Strategy

CF Contexts: Merge Re-create

Screens: Merge Re-create

Issue Type Screen Schemes: Merge Re-create

Issue Types

Any issue type
New Feature
Bug
Sub-task
Improvement

Only custom fields and issue type screen schemes associated to the selected issue types will be imported

Projects Mapping

| Source Project Key | Mapped Project Key | Operation |
|--------------------|--------------------|------------------------|
| MARPRJ | MARPRJ | Change |
| TSTPRJ | TSTPRJ | Change |
| PROJTWO | | Change |

Each project context for CFs from source will be applied to the mapped project context you choose above. If no project key mapping is selected, the CF context won't be applied during import.

For active workflows a draft is created. Do you want to publish it?
 Yes No

[Next >>](#) [Back to home](#)

After pressing the Next button, you will get to Step 2: the Actions that are going to be executed.

- Configure Import
- **Actions**
- Result

Step 2: Actions

Choose the actions you want to perform for the selected workflow.

Import File: Workflowfortest_201310021401.kwi.zip
 Issue Types: Any
 Projects Mapping: MARPRJ=Find By Key TSTPRJ=TSTPRJ
 Publish draft: No

| Action Description | Messages | Operation | Approve |
|---|---|-----------|---------|
| Status Actions <small>click to expand</small> | | | |
| Issue Types Actions <small>click to expand</small> | | | |
| Custom Fields Actions <small>click to expand</small> | 0 error(s), 3 warning(s). Click to expand | | |
| Screens Actions <small>click to expand</small> | | | |
| General Actions <small>click to expand</small> | | | |

[Next >>](#) [Back to home](#) [Go to Configure Import](#)

The actions are grouped into sections by their type: Status, Issue Types, Custom Fields, Screens, Issue Type Screen Schemes and General actions.

Each section can be expanded or collapsed with a simple click on the section title. When a section is collapsed, in the column Messages there are shown the total number of errors and warnings for the section's actions and the first 2 errors (if there are any). If you want to see more errors or the warnings click to expand the message.

Initially all sections are collapsed.

When a section is expanded, for each action in the column Messages there are shown the errors and warnings related to it (if there are any) and for the actions related to custom fields, there is a 'Change action' link that helps you re-define your custom fields actions as you need.

| | | | |
|--|--|-------------------------------|--|
| Issue Types Actions <small>click to expand</small> | | | |
| Custom Fields Actions <small>click to collapse</small> | | | |
| Create Custom Field com.keplerrominfo.jira.plugins.usergrouppicker-pro:siluserpicker (named 'best user picker', former id 'customfield_10403') | The project key 'MARPRJ' found in the context of custom field 'best user picker' does not exists on this server! | Change action | |
| Update Custom Field com.keplerrominfo.jira.plugins.databasecf.databaseinput (named 'database1', id customfield_10000) to com.keplerrominfo.jira.plugins.databasecf.databaseinput (name: 'database1', id customfield_10502) | | Change action | |
| Create Custom Field com.atlassian.jira.plugin.system.customfieldtypes:multiselect (named 'multisel', former id 'customfield_10600') | The project key 'MARPRJ' found in the context of custom field 'multisel' does not exists on this server! | Change action | |
| Update Custom Field com.keplerrominfo.jira.plugins.blitz-actions:blitzactions-cf (named 'blitz1', id customfield_10004) to com.keplerrominfo.jira.plugins.blitz-actions:blitzactions-cf (name: 'blitz1', id customfield_10700) | | Change action | |
| Create Custom Field com.keplerrominfo.jira.plugins.databasecf.databaseinput (named 'database1', former id 'customfield_10702') | | Change action | |
| Create Custom Field com.keplerrominfo.jira.plugins.keplercf:silscriptcf (named 'sil script', former id 'customfield_10703') | | Change action | |
| Update Custom Field com.atlassian.jira.plugin.system.customfieldtypes:select (named 'sel1', id customfield_10002) to com.atlassian.jira.plugin.system.customfieldtypes:select (name: 'sel1', id customfield_10801) | | Change action | |
| Update Custom Field com.keplerrominfo.jira.plugins.keplercf.intervalcf (named 'interval', id customfield_10003) to com.keplerrominfo.jira.plugins.keplercf.intervalcf (name: 'interval', id customfield_10802) | | Change action | |
| Update Custom Field com.atlassian.jira.plugin.system.customfieldtypes:textfield (named 'text1', id customfield_10005) to com.atlassian.jira.plugin.system.customfieldtypes:textfield (name: 'text1', id customfield_10803) | | Change action | |
| Screens Actions <small>click to expand</small> | | | |

For the actions that are not mandatory, in the column Approve there is a check-box that lets you uncheck the actions that you don't want to be executed. All actions are approved (checked) by default.

| General Actions click to collapse | |
|--|-------------------------------------|
| Change project keys in CF contexts according to mapping and remove unused contexts from the source workflow | |
| Update paths: Jira Home: 'C:\installs\jira\5_2_10\Application Data\JIRA' to 'C:\installs\jira\6_1\Application Data\JIRA', SIL Home: 'C:\installs\jira\5_2_10\Application Data\JIRA\silprograms' to 'C:\installs\jira\6_1\Application Data\JIRA\kepler\silprograms', Kepler Home: 'C:\installs\jira\5_2_10\Application Data\JIRA\kepler' to 'C:\installs\jira\6_1\Application Data\JIRA\kepler' | <input checked="" type="checkbox"/> |
| Create or update the SIL files | <input checked="" type="checkbox"/> |
| Create a partial sil aliases file | <input type="checkbox"/> |
| Creates the new mapping file (use it at the next import) | <input checked="" type="checkbox"/> |
| Creates the current mapping file (based on your selections) | <input checked="" type="checkbox"/> |
| Updating Custom Fields ids from special CF configurations | <input checked="" type="checkbox"/> |
| Update Workflow "Workflow for test" | <input checked="" type="checkbox"/> |

If some validation errors happens for some action, error messages are printed for that action.

| | | |
|--|---|-------------------------------|
| Create Custom Field com.keplerrominfo.jira.plugins.usergrouppicker-pro:siluserpicker (named 'best user picker', former id 'customfield_10403') | <div style="border: 1px solid red; background-color: #f08080; padding: 5px;"><p>❌ The custom field 'best user picker' cannot be imported because it has unknown type com.keplerrominfo.jira.plugins.usergrouppicker-pro:siluserpicker! Please check your plugins!</p></div> <div style="border: 1px solid yellow; background-color: #fff9c4; padding: 5px;"><p>⚠️ The project key 'MARPRJ' found in the context of custom field 'best user picker' does not exist on this server!</p></div> | Change action |
|--|---|-------------------------------|

If some of the actions have errors and you try to press the Next button, you will receive an error message: "Some of the actions have errors. Can not continue with import until all errors are gone!". In this case you should fix the errors and then continue with the import.

| |
|---|
| <p>General Actions click to expand</p> <p><input type="button" value="Next >>"/> Back to home Go to Configure Import</p> <div style="background-color: #d32f2f; color: white; padding: 10px; border: 1px solid #d32f2f;"><p>❌ Some of the actions have errors. Can not continue with import until all errors are gone!</p></div> |
|---|

The actions are determined by using the mapping file (if you have defined and selected one) and the automatic mapping process.

Since version 1.0.1-beta2, if you are not happy with an action related to custom fields, you can change it by using the [Change action](#) link.

Also, the old and not so pretty way is to change the [custom fields mapping](#), by creating or changing the mapping file. In this case, press 'Go to Configure Import' and configure the import with the new mapping file selected.

If no validation errors are present and you agree the actions and want to continue with the import, press the Next button.

If everything goes well, you will see the success message as below:

KIWI Import Workflow
Imports a workflow, associated statuses, custom fields, issue types and screens.

KIWI Import Step 3: Result

Import File: Workflowfortest_201310021401.kwi.zip
 Issue Types: Any
 Projects Mapping: MARPRJ=Find By Key TSTPRJ=TSTPRJ
 Publish draft: No

Successfully imported workflow "Workflowfortest_201310021401.kwi.zip."

Custom field with id: customfield_10102: The context corresponding to Issue Types Default Configuration Scheme for best user picker, Projects: [MARPRJ] could not be created. Cause: No corresponding associated project found on destination!

Generated empty sil aliases file!

1 warning(s) . 1 messages. Click to collapse

[Back to home](#) [Import another?](#)

Even if the workflow was imported successfully, it might be the case that one or more actions finished with an error or warning. All these messages(errors, warnings or info) are listed below the "Successfully imported workflow" message. For instance, when creating a context for a custom field, if no project corresponding to the context can be found on the Export server, the context is not created and a warning message is displayed. If the generated sil aliases file is empty, an info message is displayed.

Import actions

Briefly, the following rules are followed in the import process:

- Workflows are mapped by name. If on the Jira import instance an workflow with the same name as the exported one is found, this existing workflow will be updated to match the exported one, otherwise the workflow will be created
- Statuses are mapped by name
- Issue types are mapped by name
- Issue type screen schemes and Screen schemes are mapped by name. If on the Jira Import instance an Issue type screen scheme with the same name is found, it is updated to match the exported one (to have the same description and issue type -> screen scheme associations), otherwise the Issue type screen scheme will be created
- Screens are mapped by name. Every screen that is exported is analyzed, and if the screen exists on the Jira Import instance (is found by name), it will be updated to match the screen from the exported file, otherwise the screen will be created
- Custom fields are mapped using the mappings(according to the screen selection or from the mapping file). If no mapping is found, custom fields are automatically mapped by name and type. Every custom field that is present in one of the screens exported is analyzed(let's name it **exported cf**):
 1. if a mapping is found for it, get the custom field from the Jira Import instance on which the exported cf is mapped and update it to match the exported cf(The name, description, searcher and the context(s) are updated)
 2. otherwise if a custom field with the same name and type is found in the Jira import instance, this existing custom field will be updated to match the exported custom field(The description, searcher and the context(s) are updated)
 3. otherwise the custom field will be created

Change action

Here you can change an action related to custom fields.

When you first click on the Change action link for an action, a pop-up showing the current action appears:

Change action for custom field multisel

Action:

Source custom field: multisel - customfield_10600

[Change action](#) [Get me out of here](#)

This is the pop-up for the action 'Create Custom Field com.atlassian.jira.plugin.system.customfieldtypes:multiselect (named 'multisel', former id 'customfield_14900)'

If you want to change it to an update action, choose UPDATE instead of CREATE from the select list near Action. After that a select list with compatible custom fields(having the same type with the source custom field) appears:

Change action for custom field multisel

Action:

Source custom field: multisel - customfield_10600

Old destination custom field: -

Change destination custom field to:

[Change action](#) [Get me out of here](#)

You can give up any moment by pressing the Get me out of here link.

Choose the custom field you want from the select list(near 'Change destination custom field to') and press the Change action button.

You should see your changed action in the list of actions(The warning shows that the project with key 'MARPRJ', that is used in the context of the custom field 'multisel' does not exist):

| | |
|---|-------------------------------|
| Update Custom Field com.keplerrominfo.jira.plugins.databasecf.databaseinput (named 'database1', id customfield_10000) to com.keplerrominfo.jira.plugins.databasecf.databaseinput (name: 'database1', id customfield_10502) | Change action |
| Update Custom Field com.atlassian.jira.plugin.system.customfieldtypes:multiselect (named 'multisel2', id customfield_10103) to com.atlassian.jira.plugin.system.customfieldtypes:multiselect (name: 'multisel', id customfield_10600) | Change action |
| <div style="border: 1px solid yellow; padding: 5px; display: inline-block;"> <p>⚠ The project key 'MARPRJ' found in the context of custom field 'multisel' does not exists on this server!</p> </div> | |
| Update Custom Field com.keplerrominfo.jira.plugins.blitz-actions:blitzactions-cf (named 'blitz1', id customfield_10004) to com.keplerrominfo.jira.plugins.blitz-actions:blitzactions-cf (name: 'blitz1', id customfield_10700) | Change action |

Please note that a validation error might appear after changing an action if the new destination custom field selected is already mapped in another action, so you should carefully change your actions.

Validation errors

- The type for the exported custom field doesn't exist on the server where you execute the import(The plugin is not installed or not enabled):

| | | |
|---|---|--------------------------------------|
| <p>Create Custom Field com.keplerrominfo.jira.plugins.usergrouppicker-pro:siluserpicker (named 'best user picker', former id 'customfield_10403')</p> | <p>! The custom field 'best user picker' cannot be imported because it has unknown type com.keplerrominfo.jira.plugins.usergrouppicker-pro:siluserpicker! Please check your plugins!</p> | <p>Change action</p> |
|---|---|--------------------------------------|

Solution: Install your missing plugin, or make sure it is enabled.

- **The exported custom field doesn't have the same type with the custom field that will be mapped to:**

| | | |
|---|---|--------------------------------------|
| <p>Update Custom Field com.atlassian.jira.plugin.system.customfieldtypes:textfield (named 'text1', id customfield_10005) to com.keplerrominfo.jira.plugins.blitz-actions:blitzactions-cf (name: 'blitz1', id customfield_10700)</p> | <p>! The custom field 'blitz1' (former id customfield_10700) cannot be imported because it is mapped to a custom field with different type! (com.atlassian.jira.plugin.system.customfieldtypes:textfield not the same with 'com.keplerrominfo.jira.plugins.blitz-actions:blitzactions-cf')</p> | <p>Change action</p> |
|---|---|--------------------------------------|

Solution: Change the mapping for the custom field with problems(in the example above blitz1), by using the 'Change action' link or directly into the mapping file, so as to be mapped on a custom field with the same type.

- **Two or more custom fields are mapped onto the same custom field:**

| | | |
|---|--|--------------------------------------|
| <p>Update Custom Field com.keplerrominfo.jira.plugins.databasecf.databaseinput (named 'database1', id customfield_10104) to com.keplerrominfo.jira.plugins.databasecf.databaseinput (name: 'database1', id customfield_10502)</p> | <p>! The custom field 'database1' (customfield_10104) is mapped more than once!</p> | <p>Change action</p> |
|---|--|--------------------------------------|

Solution: Change your mappings (by using the 'Change action' link or directly into the mapping file) so that only one custom field is mapped on the custom field from the message (if you change directly into the mapping file, for the example above, customfield_10104 must be present only once in the right column) .

Custom Fields Mapping

As you may have noticed, in the Kiwi Import page you can choose a mapping file that allows you to define your own mapping of exported custom fields onto existent ones.

What is this file for?

It provides the plugin with rules that must be followed when mapping the custom fields you've exported onto existent custom fields on the import instance. The mapping file is a plain text file that can be edited with any text editor.

Manually creating a custom field mapping file

The first step is to create a new text file (the name is up to you) with the extension **.cfmap** in the following path: **"JIRA_HOME_FOLDER/kepler/kiwi"**.

After that fill in the file with the mappings by your needs: Choose a custom field from your exported workflow that you want to be mapped, and get **the custom field ID**. Get the custom field ID from the Jira import instance onto which the exported custom field will be mapped. Add a new line corresponding to the mapping in your mapping file as

```
customfield_eeeeee=customfield_iiiiii
```

, where customfield_eeeeee is the ID of the custom field from the exported file and customfield_iiiiii is the ID of the custom field from the Jira import instance. Repeat the operation for all the custom fields you want to map.

Here's an example:

```

customfield_10303 = customfield_11100
customfield_10305 = customfield_11101
customfield_10312 = customfield_11102
customfield_10313 = customfield_11103
customfield_10321 = customfield_11104

```

The first row from the example above can be read like this: "Map the custom field 10303 from my exported workflow onto the custom field 11100"

from this JIRA system". The first column of the mapping file has the custom field ids that you want to map from the exported workflow, and the second column contains the ids of the custom fields from the Jira server where the import is executed.

Import generated mapping files

A successful import will generate two mapping files(the current and the next mapping file) in the **JIRA_HOME_FOLDER/kepler/kiwi** folder.

The **next mapping file** is named using the following pattern: <imported_workflow_name>_next_<timestamp>.cfmap. It contains all the mappings resulted from the import(according to your used .cfmap file and according to your screen selection), including the created custom fields.For an update custom field action, a new entry with the match/mapping [the source custom field -> the destination custom field] is added to the file. For a create custom field action, the entry will contain the source custom field id mapped on the newly created custom field id. This file can be used for a future import on the same server.

The **current mapping file** is named using the following pattern: <imported_workflow_name>_current_<timestamp>.cfmap. It contains all the mappings(according to your used .cfmap file and according to your screen selection) for the custom fields existing on the Import server. For an update custom field action, a new entry with the match/mapping [the source custom field -> the destination custom field] is added to the file. Nothing is added to this file for a create custom field action.This file is useful for executing again the same import on another server that is identical as a structure to the Jira Import server where the file was generated.

For both mapping files names <timestamp> represents the date at which the file is generated, in the format yyyyMMddHHmm.

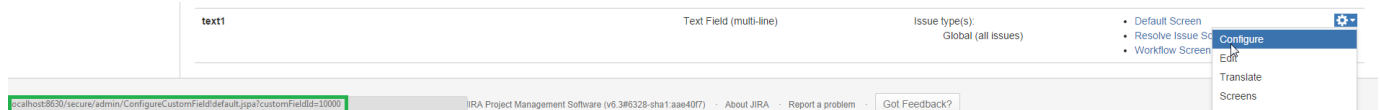
To merge two generated CF Maps files, you need to use the [corresponding tool](#).

Getting a custom field ID

Go to **Administration -> Custom Fields**.

Choose your desired custom field and click on the **cog icon**

. Set your mouse over the Configure link. In the link displayed on the bottom of the page you'll see the URL containing the custom field ID you're looking for.



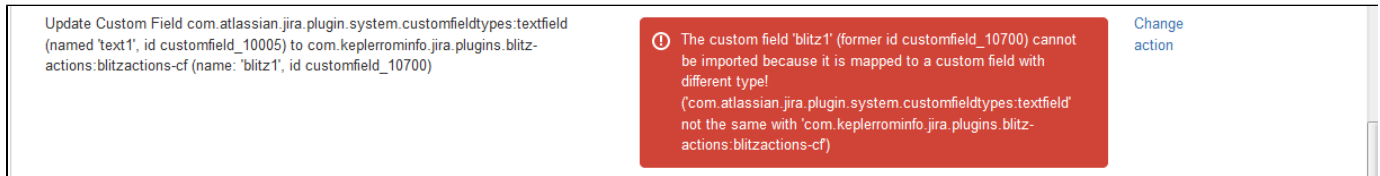
Another way to get a custom field ID is by querying your data base if you have configured an external one.

Mapping error messages

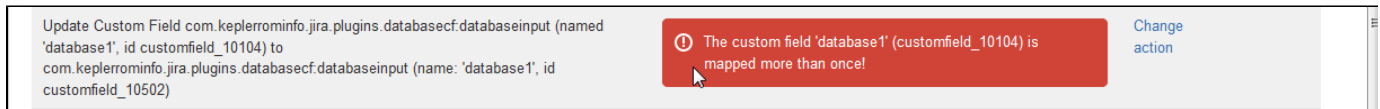
If the mapping file has wrong mappings, error messages are displayed for the corresponding actions at **Step 2: Actions**.

The problems that might occur are:

- The exported custom field doesn't have the same type with the custom field that will be mapped to. In this case a message error will appear:



- Two or more custom field are mapped onto the same custom field.



Both cases will generate errors that won't let you continue with the import until you fix them.

If your exported custom field has been mapped several times in the mapping file, it will be used the last mapping(in other words the last row in which is written on the first column) without generating any error message.

Tips and Tricks

The mapping file can force **KIWI** to avoid the automatic matcher for custom fields and create a brand new Custom Field with all its dependencies from the export system. To do this just map the custom field to the empty string (leave a blank after the equals sign), like this:

Force create a custom field

```
customfield_10304 = customfield_10222
customfield_10309 =
customfield_10316 =
customfield_10327 = customfield_10406
```

This mapping file will force the creation on the Jira import instance of the custom fields 10309 and 10316 from the export file.

A Note About SIL Aliases

If you are using our **JJUPIN** plugin (the **SIL** provider) you may have notice that there is a **sil.aliases** file in the Kepler folder of your Jira Home folder. This file maps the custom fields onto a more easy readable names that can be used in the SIL scripts. KIWI transfers this file from an instance of JIRA to another. More information about sil.aliases file can be found here: <http://confluence.kepler-rominfo.com/display/SIL/JIRA+instance-independent+programming>

The export operation encapsulates the entire file form the source server, including the comment lines.

A successful import will generate this file in the Kiwi home folder with the name: **<exported_workflow_name>_sil_<timestamp>.aliases** , where **<timestamp>** represents the date at which the file is generated, in the format **yyyyMMddHHmm**. (For instance, the file **Workflowfortest_sil_201310071330.aliases** is the sil.aliases file generated when importing the workflow **Workflowfortest**, on 07.10.2013, 13:30). This file will contain only those custom field aliases that were imported, **re-mapped according to your .cfmap file and according to your screen selection**. However, we want to make it clear here, the generated file **<exported_workflow_name_sil>.aliases** will contain only **the delta**, i.e. what has been touch by the current import process.

To merge your generated file in the **sil.aliases** you need to launch the **corresponding tool**.

Import Options

The section Import Options is available at Step 1, Configure Import, after a workflow is selected.

Here you can define your own settings for the import execution:

Step 1: Configure Import

Set up the configuration parameters for the workflow import.

Import File:
Select a kiwi workflow file to import from

Mapping File:
Select a mapping file for custom fields

IMPORT OPTIONS
click to collapse

Import Strategy

CF Contexts: Merge Re-create

Screens: Merge Re-create

Issue Type Screen Schemes: Merge Re-create

Issue Types

Bug
 Sub-task
 Improvement

Only custom fields and issue type screen schemes associated to the selected issue types will be imported

Projects Mapping

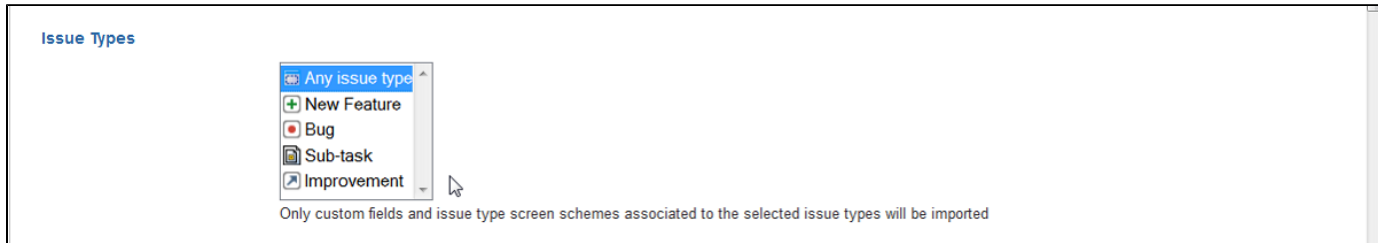
| Source Project Key | Mapped Project Key | Operation |
|--------------------|--------------------|-----------|
| TSTPRJ | TSTPRJ | Change |

Import Strategy

For the custom field contexts, screens and issue type screen schemes you can choose if in case an update is needed, the entities to be imported will be merged with the existing ones or re-created. The default is Merge for all the three entities.

Issue Types

From the select list with issue types you can choose what issue types to be taken into account when importing the workflow.



The contexts of the custom fields that are not associated to 'Any issue type', are partially imported according to the selected issue types. Also, only the custom fields that have at least one of the selected issue types associated in their context or that have global context, and only the relevant issue type screen schemes associations for the selected issue types will be imported.

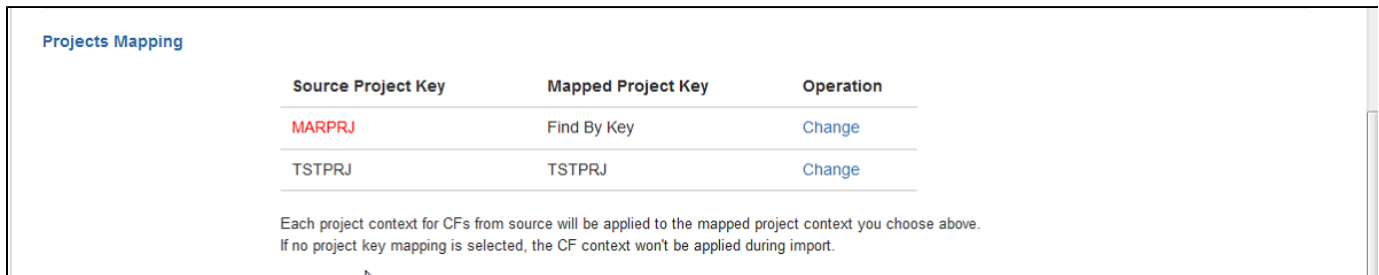
The default value is Any Issue Type.

Projects Mapping

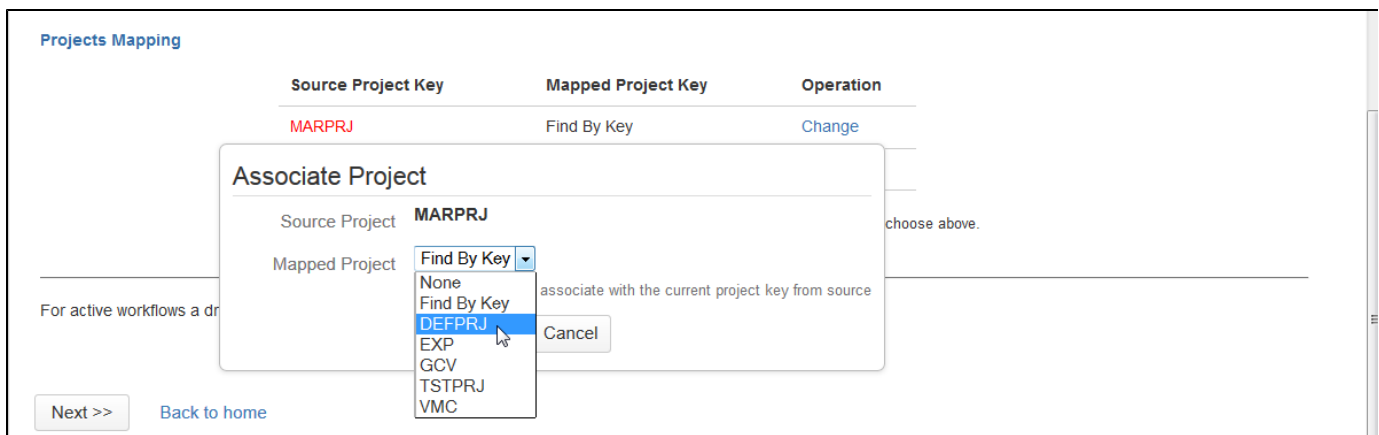
Here you can define a mapping between a project from the imported kiwi file and a project from the Jira Import server. This is especially useful when the same project has different keys on the export and import server.

The projects that can be mapped here are the projects that are used in the contexts of the exported custom fields.

By default the projects are mapped by their key. If the corresponding key can not be found on the Import server, the Source Project Key is marked with the red color (as MARPRJ is in the picture below) and the corresponding value in the column Mapped Project Key is by default 'Find By Key'. This means that the project key will be searched again in the moment the import is really executed (Maybe in the meantime the project is created, for instance by a pre-import sil script).



You can change the mapping for a project by pressing the Change link.



Here you can choose from one of the existing values from the Import server.

An existing project from the Import server can be used only once as a Mapped Project Key.

If you choose the value **None**, the corresponding source project will be ignored from the custom field contexts where it appears.

If the mapped project key for a source project key is None or can not be found at the import execution time, the project will be ignored from the custom field contexts where it is used. When trying to create a custom field context, if no associated project key can be found, then the context will not be created any more.

Publish active workflows

If the workflow that you are trying to import exists and is active on the Jira Import server, a draft is created for the imported workflow. If you want to automatically publish this draft, change the value of the radio 'For active workflows a draft is created. Do you want to publish it?' to Yes.

If the default value(No) is selected, only the draft is created(or updated if the draft exists) and it needs to be manually published.

KIWI Tools

Besides its main purpose of carrying over an workflow and its dependencies, Kiwi also produces different deltas (CF mapping files and sil.aliases files) and offers the tools to merge these files.

CF Mapping files

The cf mapping files(the current and the next mapping file) offer you the [custom field mappings](#) that can be reused. They are generated by a successful import in the **JIRA_HOME_FOLDER/kepler/kiwi** folder.

The *next mapping file* contains all the custom field mappings resulted from the import(according to your used *.cfmap* file and according to your screen selection), including the created custom fields. This can be used for a future import on the same server.

The *current mapping file* contains all the mappings(according to your used *.cfmap* file and according to your screen selection) for the **custom fields existing** on the Import server. This is different from the next mapping file by not including the newly created custom fields. This file is useful for executing again the same import on another server that is identical as a structure to the Jira Import server where the file was generated.

To merge two generated CF Maps files, you need to use the [.cfmap merging tool](#).

SIL aliases files

The *sil.aliases* file is related to **SIL**, or **Simple Issue Language** and our **JJUPIN** plugin. This file(located in the Kepler home folder) maps the custom fields onto a more easy readable names that can be used in the SIL scripts.

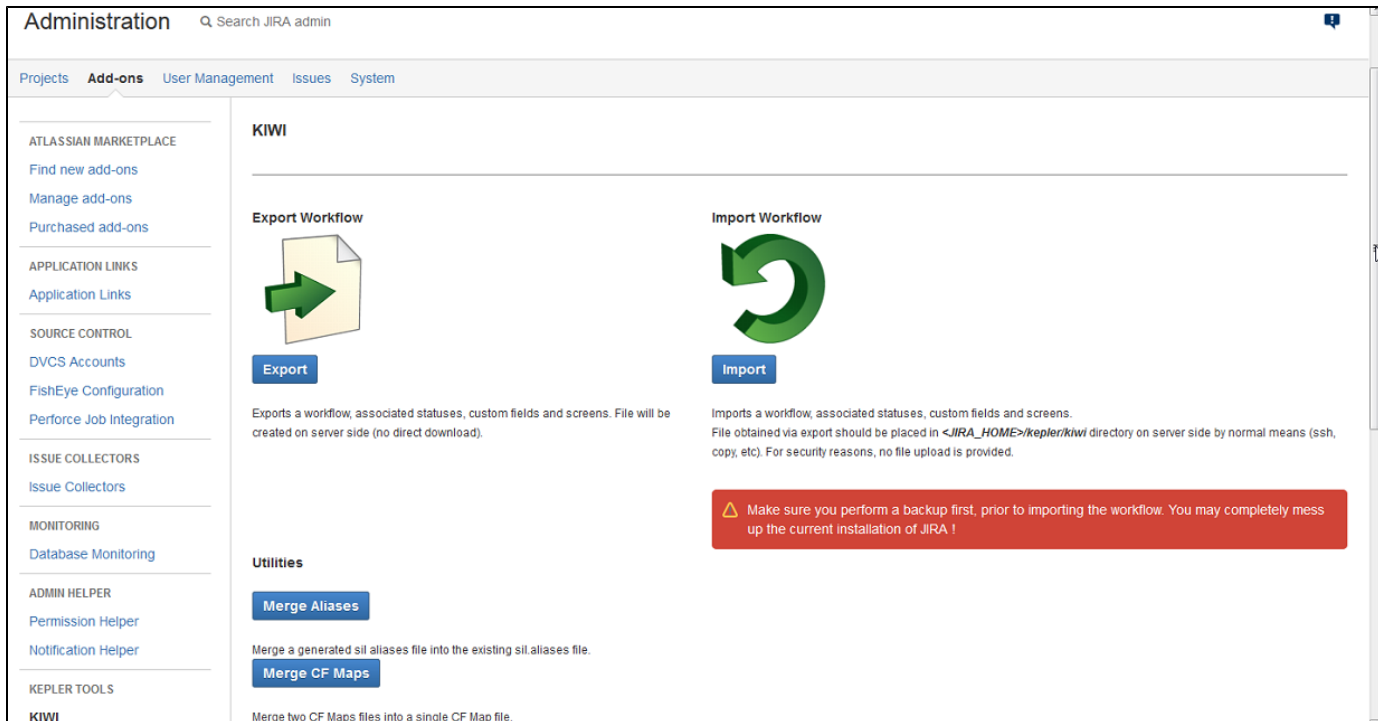
A successful import will generate a **kiwi local sil.aliases** file(different from the sil.aliases file described above) in the Kiwi home folder. This file will contain **only those custom field aliases that were imported**, re-mapped according to your *.cfmap* file and according to your screen selection.

To merge your generated file in the *sil.aliases* you need to launch the [sil.aliases merging tool](#).

Merging .cfmap files

After the import, Kiwi offers you the possibility to merge two generated CF Maps files.

This merging utility is accessible from the main KIWI page by pressing the 'Merge CF Maps' button:



The screenshot shows the Jira Administration interface. The top navigation bar includes 'Administration' and a search box. Below it, there are tabs for 'Projects', 'Add-ons', 'User Management', 'Issues', and 'System'. The left sidebar lists various categories like 'ATLASSIAN MARKETPLACE', 'APPLICATION LINKS', 'SOURCE CONTROL', 'ISSUE COLLECTORS', 'MONITORING', 'ADMIN HELPER', and 'KEPLER TOOLS'. The main content area is titled 'KIWI' and contains two primary actions: 'Export Workflow' (with a document icon and a green arrow) and 'Import Workflow' (with a circular arrow icon). Below 'Export Workflow' is an 'Export' button and a description: 'Exports a workflow, associated statuses, custom fields and screens. File will be created on server side (no direct download)'. Below 'Import Workflow' is an 'Import' button and a description: 'Imports a workflow, associated statuses, custom fields and screens. File obtained via export should be placed in <JIRA_HOME>/kepler/kiwi directory on server side by normal means (ssh, copy, etc). For security reasons, no file upload is provided.' A red warning box with a triangle icon states: 'Make sure you perform a backup first, prior to importing the workflow. You may completely mess up the current installation of JIRA !'. At the bottom of the KIWI section, there are two utility buttons: 'Merge Aliases' (with a description: 'Merge a generated sil aliases file into the existing sil.aliases file.') and 'Merge CF Maps' (with a description: 'Merge two CF Maps files into a single CF Map file.').

You must choose the two cf map files to be merged using the combo boxes CF Map 1 and CF Map 2. The numeric part from the generated cf map file names represents the date at which the file was generated, in the format yyyyMMddHHmm. For instance, the

file Workflowfortest_current_201310041808.cfmap is the current cf map file generated when importing the workflow Workflowfortest at local time 04.10.2013, 18:08.

Merge CF Maps

Merge two CF Maps files into a single file.

SIL CF Map 1: Workflowfortest.cfmap
Select first CF Map file to be merged

SIL CF Map 2: Workflowfortest_current_201310041808.cfmap
Select second CF Map file to be merged

You must select two CF Maps

If everything works as planned, you'll see a SUCCESS message on the screen. The 2 files were merged into a new file, **cfmapMerge_yyyyMMddHHmm.cfmap**, where yyyyMMddHHmm represents the current date in the format yyyyMMddHHmm.

Merge CF Maps

Merge two CF Maps files into a single file.

SIL CF Map 1: Workflowfortest.cfmap
Select first CF Map file to be merged

SIL CF Map 2: Workflowfortest_current_201310041808.cfmap
Select second CF Map file to be merged

Apply

Successfully merged the files.

Merging sil.aliases files

After the import, Kiwi offers you the possibility to merge an import generated sil.aliases file into the existing sil.aliases file.

This merging utility is accessible from the main KIWI page by pressing the 'Merge Aliases' button:

Administration Search JIRA admin

Projects **Add-ons** User Management Issues System

ATLASSIAN MARKETPLACE
Find new add-ons
Manage add-ons
Purchased add-ons

APPLICATION LINKS
Application Links

SOURCE CONTROL
DVCS Accounts
FishEye Configuration
Perforce Job Integration

ISSUE COLLECTORS
Issue Collectors

MONITORING
Database Monitoring

ADMIN HELPER
Permission Helper
Notification Helper

KEPLER TOOLS

KIWI

Export Workflow

Exports a workflow, associated statuses, custom fields and screens. File will be created on server side (no direct download).

Import Workflow

Imports a workflow, associated statuses, custom fields and screens. File obtained via export should be placed in <JIRA_HOME>/kepler/kiwi directory on server side by normal means (ssh, copy, etc). For security reasons, no file upload is provided.

Utilities

Merge Aliases

Merge a generated sil aliases file into the existing sil.aliases file.

Merge CF Maps

Merge two CF Maps files into a single CF Map file.

Make sure you perform a backup first, prior to importing the workflow. You may completely mess up the current installation of JIRA !

You must choose from the list of kiwi generated sil.aliases files. The numeric part from the generated sil.aliases files names represents the date at which the file was generated, in the format yyyyMMddHHmm. For instance, the file Workflowfortest_sil_201310071330.aliases is the sil.aliases file generated when importing the workflow Workflowfortest, on 07.10.2013, 13:30.

[<< Back to Kiwi Home](#)

Merge aliases

Merge a generated aliases file into the existing sil.aliases file.

SIL Aliases File: existing sil.aliases file

If everything works as planned, you'll see a SUCCESS message on the screen. The file Workflowfortest_sil_201310071330.aliases was merged into the sil.aliases file.

[<< Back to Kiwi Home](#)

Merge aliases

Merge a generated aliases file into the existing sil.aliases file.

SIL Aliases File: Select a generated aliases file to be merged into the existing sil.aliases file

Successfully merged SIL aliases file 'Workflowfortest_sil_201310071330.aliases' to destination.

Supported Custom Fields

KIWI exports/imports the custom fields from the screens related to the workflow (e.g. transition screens and the screens that are found in the related Issue type screen schemes).

All Jira standard custom fields are supported. These custom fields are exported along with their contexts: The issue types, projects, default value and options(if the custom field may have options) associated to each context are exported and imported on the deployment machine.

Beside these, Kiwi knows how to handle the following Kepler provided Custom Fields:

- Blitz Actions Custom Field
- SIL User Picker
- Database Information
- Data Table Information
- Database Child Information
- SIL Script Custom Field
- User Picker - per Group
- Regular Expression Custom Field
- Interval Field

For these custom fields, the special configurations(if there are any) are carried over, so no change is needed on the deployment machine.

For instance, for Blitz Actions Custom Field, all the Actions(buttons) together with their scripts are exported and restored on the import server.

For DBCF fields, the configurations from the Kepler parameters page are carried over(the JNDI names, Sql queries, SIL scripts, all that is needed to fully restore the configurations). For dynamic queries, the custom field ids are updated with their corresponding ids from the target Jira instance.

For DBCF fields the used data sources must be manually configured on the Import server. Please see [Data Source Configuration](#) for details on how to manually configure a Data Source.

For SIL User Picker and SIL Script Custom Field, the SIL Script for each context is carried over.

For REGEX custom field the Regex configuration is exported/imported.

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).

KIWI has two dependencies:

1. katl-commons (core support)
2. warden (used for licensing)

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

| Plugin | Tables |
|--------------|--|
| katl-commons | kplugins kpluginscfg kissuestate kstatevalues |
| warden | - |