JIRA Administrator's Guide

1. JIRA Administrator's Guide

1.1. JIRA Administrator's Guide

This manual contains information on administering your JIRA system:

1.1.1. Getting started

Please see the *JIRA User's Guide* for an introduction to the concepts of issues and projects.

1.1.2. Getting help

- Do you have a question that's not covered in this manual? Try the Administrator's Knowledge Base.
- Having problems or errors with JIRA? Please see debugging problems.

1.2. Getting Help

If you encounter any problems using or setting up JIRA, please let us know — we're here to help!

You may want to first search the following:

- the JIRA mailing list forums, where Atlassian staff and JIRA users can answer your questions.
- the JIRA Knowledge Base.

If you need further assistance, please raise a support request (see below).

Alternatively, if you feel you have encountered a bug in JIRA, or wish to request a feature, please file an issue. It is a good idea to first scan JIRA's Popular Issues — this helps to prevent duplicates.

Note:

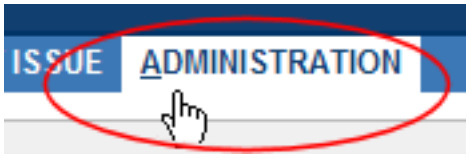
Looking for more helpful information? You can receive news, product information and code tips via our newsletter, blogs and forums.
Stay in touch with us here.

1.2.1. Raising a Support Request

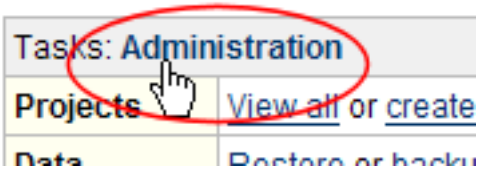
To raise a support request via your JIRA system (recommended, provided your SMTP email

is enabled),

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

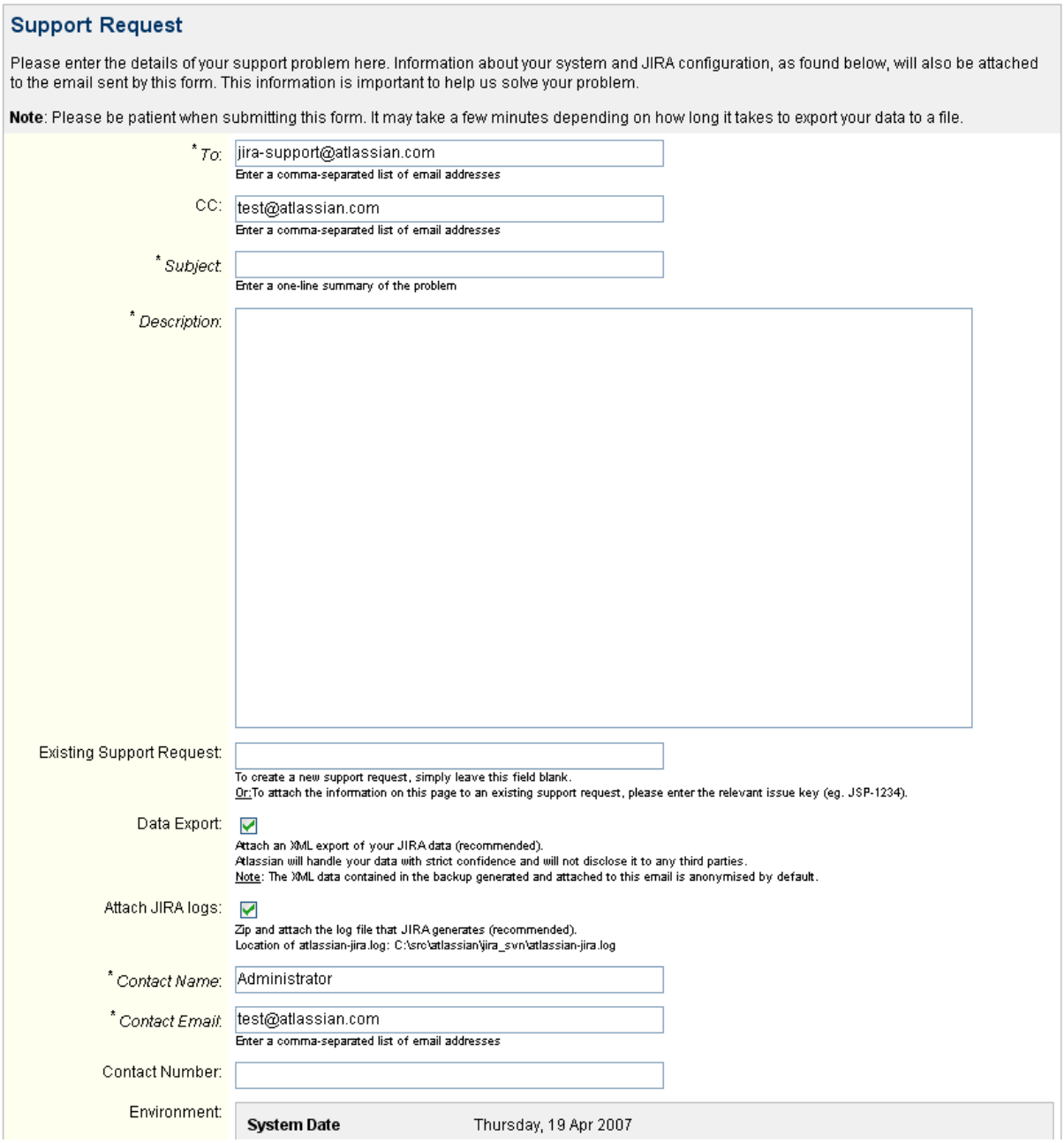


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title 'System', click the 'Support Request' link.
4. The 'Support Request' form will be displayed:
   - Please provide as much information as possible, including any error messages that are appearing on the console or via log4j.
   - Please select 'Data Export' and 'Attach JIRA logs'.
   - If you have previously raised a support request for the problem, please type the issue key (e.g. **JSP-1234** into the 'Existing Support Request' field.



Create a Support Request

5. Once you have submitted your support request, you will receive email updates about its progress. You can also view the status of your support request by visiting the Atlassian Support System

OR:

To raise a support request via the internet,

1. Please visit the Atlassian Support System and create a support request.
2. Please provide as much information as possible, including any error messages that are appearing on the console or via log4j. Please also mention the operating system, database and version of JIRA you are using.

Note:

Sometimes it is necessary to adjust JIRA's logging levels to get a more detailed error message or a stack trace. Please see the logging section of the documentation for information on how to do this.

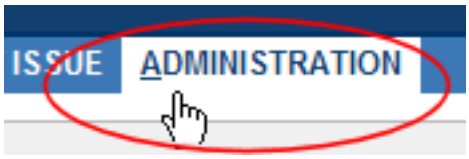
1.3. Look and Feel

1.3.1. Configuring Look & Feel in JIRA

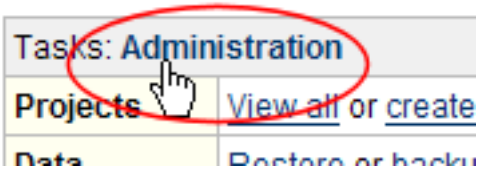
You can easily customise JIRA's look and feel to suit your needs.

To view the current look and feel configuration,

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

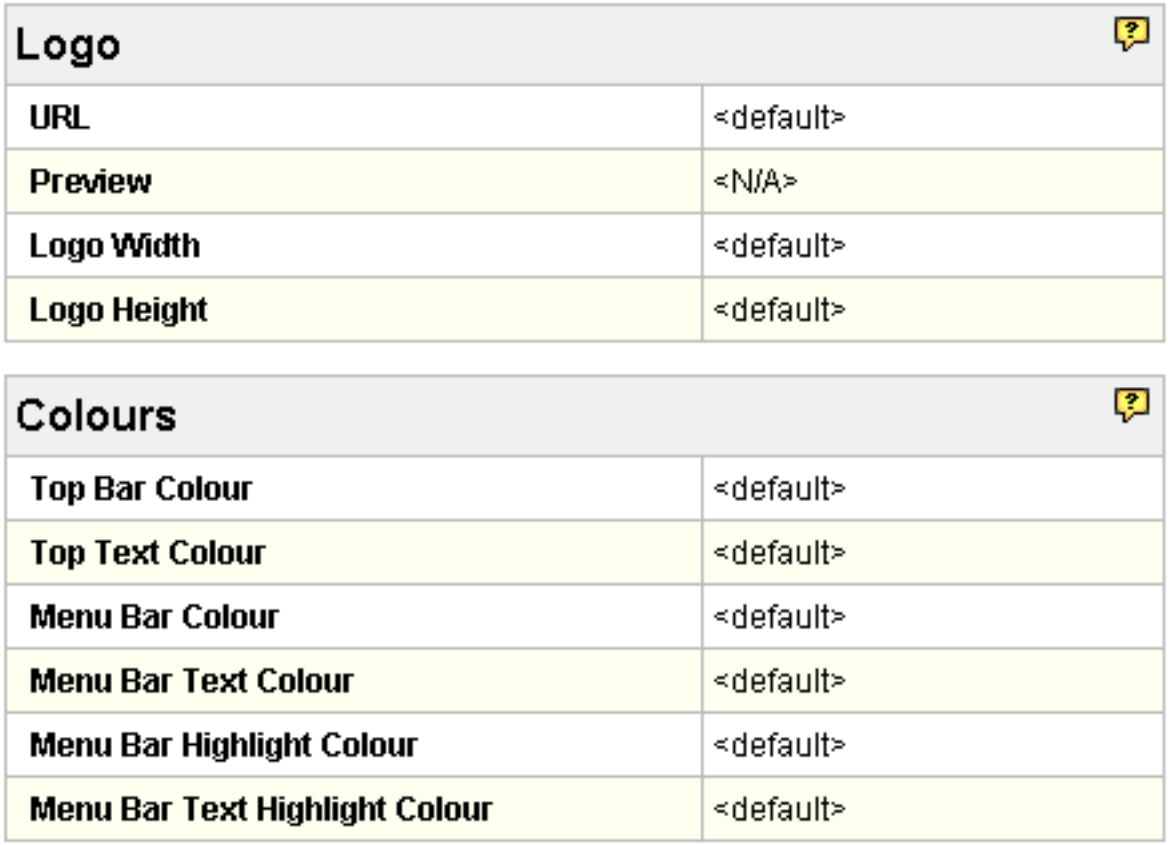


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title '**Global Settings**', click '**Look and Feel**.'
4. The '**Look and Feel Configuration**' page will be displayed. See screenshot below.



Logo	
URL	<default>
Preview	<N/A>
Logo Width	<default>
Logo Height	<default>

Colours	
Top Bar Colour	<default>
Top Text Colour	<default>
Menu Bar Colour	<default>
Menu Bar Text Colour	<default>
Menu Bar Highlight Colour	<default>
Menu Bar Text Highlight Colour	<default>

Screen shot of JIRA look and feel settings

Clicking *Edit Configuration* will allow you to edit the current configuration. To reset to the default look and feel of JIRA, simply click on the 'Reset Default' button.

Here is a list of the different configuration options available, and what they do.

1.3.1.1. Logo

Option	Explanation
URL	This URL points to the absolute or relative path of the image that you wish to display at the top of the page. If the URL begins with '**http://**' or '**https://**' then the URL is treated as an absolute URL. Otherwise it will be treated as a relative URL, and will have to be packaged in the war file when you build JIRA.
Preview	When an image is selected, a preview will be shown here.
Logo Width	The width of the image, usually in pixels. You can use any width that is valid in an image tag. eg. '*100px*', '*80%*'
Logo Height	The height of the image, usually in pixels. You can use any height that is valid in an image tag. eg. '*25px*', '*30%*'

1.3.1.2. Colours

The colours you choose for each of the sections can be anything that is valid for both a font tag, and a stylesheet's 'color:' attribute.

When choosing a colour - you can use the pop-up colour chooser, or specify your own. eg. '*#FFFFFF*', '*red*'.

To return to the original colour scheme, just clear any values that you have set.

Option	Explanation
Top Bar Colour	The background colour of the top bar (the one that includes the image).
Top Text Colour	The colour of the text that sits inside the top bar (such as your user name when you are logged in).
Menu Bar Colour	The background colour of the bar that contains the links to 'HOME' and 'BROWSE PROJECT'.
Menu Bar Text Colour	The text color of the links in the menu bar (such as 'HOME').
Menu Bar Higlight Colour	The colour of the menu bar when it is selected or when the mouse hovers over it.
Menu Bar Text Highlight Colour	The colour of the menu bar text when it is highlighted or when the mouse hovers over it.
Link Colour	The colour of the text links.
Link Active Colour	The colour of the text links when it is selected.
Heading Colour	The colour of the text headings (such as 'Logo').

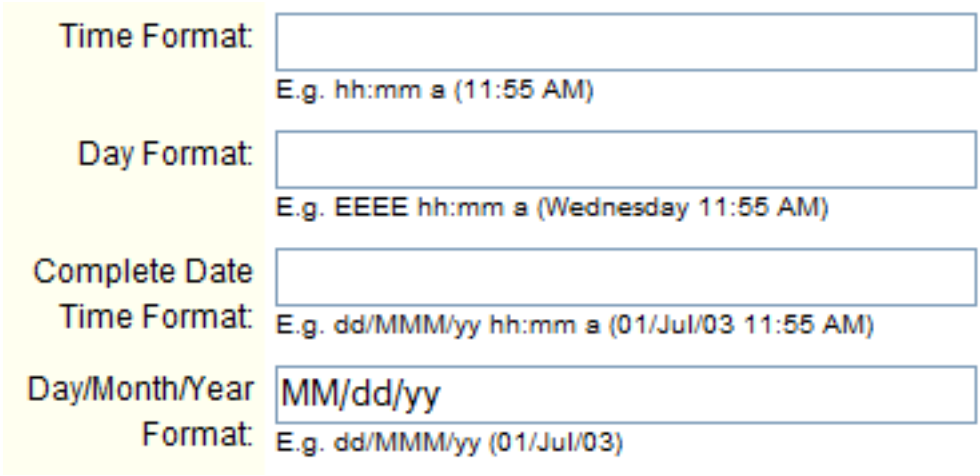
1.3.1.3. Time Formats

All time formats in JIRA are configurable. On the first look and feel page, you can preview how the current date and time are being formatted using the default formats.

Date/Time Formats	Format	Example
Time Format	hh:mm a	10:34 AM
Day Format	EEEE hh:mm a	Thursday 10:34 AM
Complete Date/Time Format	EEE MM/yy	Thu 07/03
Day/Month/Year Format	dd/MMM/yy	03/Jul/03

View Time Formats

You can edit these defaults by clicking on the "Edit Configuration" link down at the bottom. For example, you may wish to change the day/month/year format to show the month first. Simply enter MM/dd/yy beside the day/month/year format field, and preview by clicking the "update" button.



Edit Time Format

The preview page should show the current time in the new format.

Date/Time Formats	Format	Example
Time Format	hh:mm a	12:11 PM
Day Format	EEEE hh:mm a	Thursday 12:11 PM
Complete Date/Time Format	dd/MMM/yy hh:mm a	03/Jul/03 12:11 PM
Day/Month/Year Format	MM/dd/yy	07/03/03

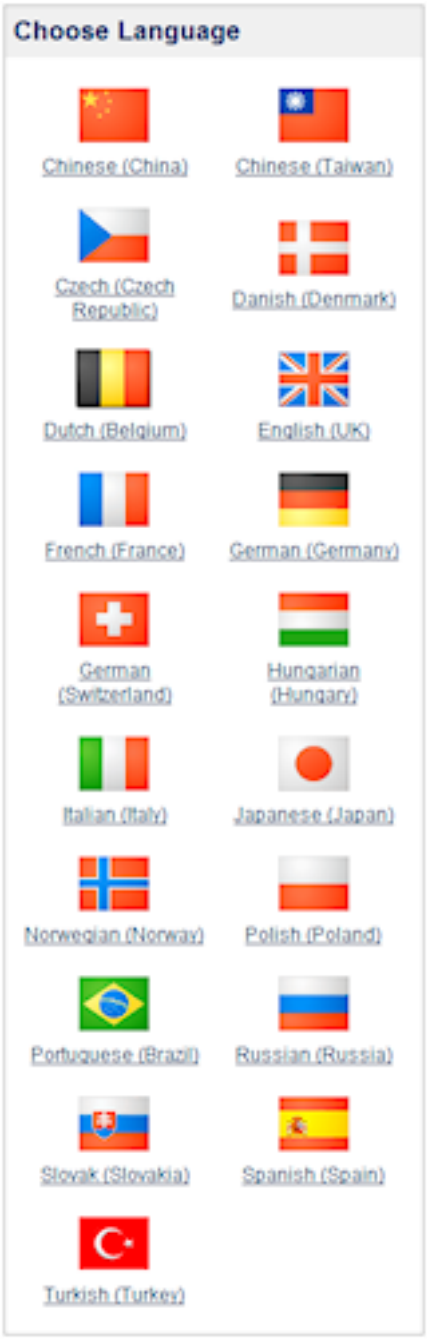
Updated date format showing month first

1.3.2. Choosing a default language

1.3.2.1. Overview

Most user-visible pages in JIRA are now internationalised. Chinese, Czech, Danish, English, French, German, Italian, Norweigian, Polish, Portuguese (Brazilian), Russian, Slovak and Spanish translations are available (at time of writing), with more in development.

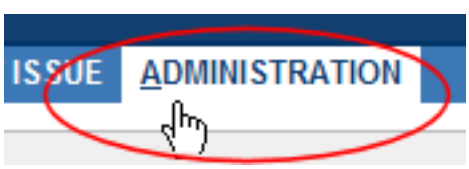
When JIRA is first installed, the default language may be chosen by clicking on a flag:



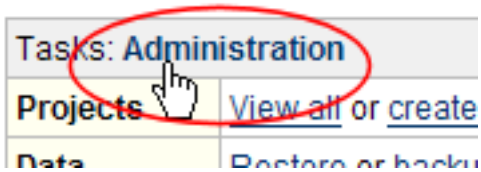
Choose Language

1.3.2.2. Changing the default language

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

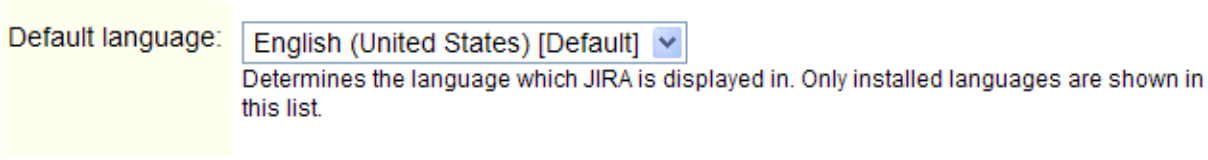


Link to Administration section



Link to Administration section

3. Click "General Configuration" (under the "Global Settings" subheading on the left).
4. Click "Edit Configuration", then select the appropriate language in the drop-down box next to "Default language".



Editing default language

Any additional languages you have installed will appear in the list.

1.3.2.3. Per-user language selection

Individual users can choose their own language, which will override the default language (see above).

1.3.2.4. Overriding the default translations of Issue Types, Resolutions, Statuses and Priorities

Should you wish, you can easily specify your own translations for the values of the following JIRA issue fields:

- Issue Type
- Priority
- Status
- Resolution

Your specified translations will override the values specified in the JIRA translation.

1.3.3. Configuring the Issue Navigator

The Issue Navigator is used within JIRA to find and filter issues, and to display the search results in various formats ('views'). It is possible to select which issue fields will be displayed as columns in the issue navigator.

JIRA **administrators** can configure which columns appear in the issue navigator by default, for all users that do not have their personal navigator columns configured. Each authenticated JIRA **user** can override these defaults by configuring their own navigator columns to fit their needs. Note that only users who can see at least one issue in the JIRA installation are able to configure issue navigator columns.

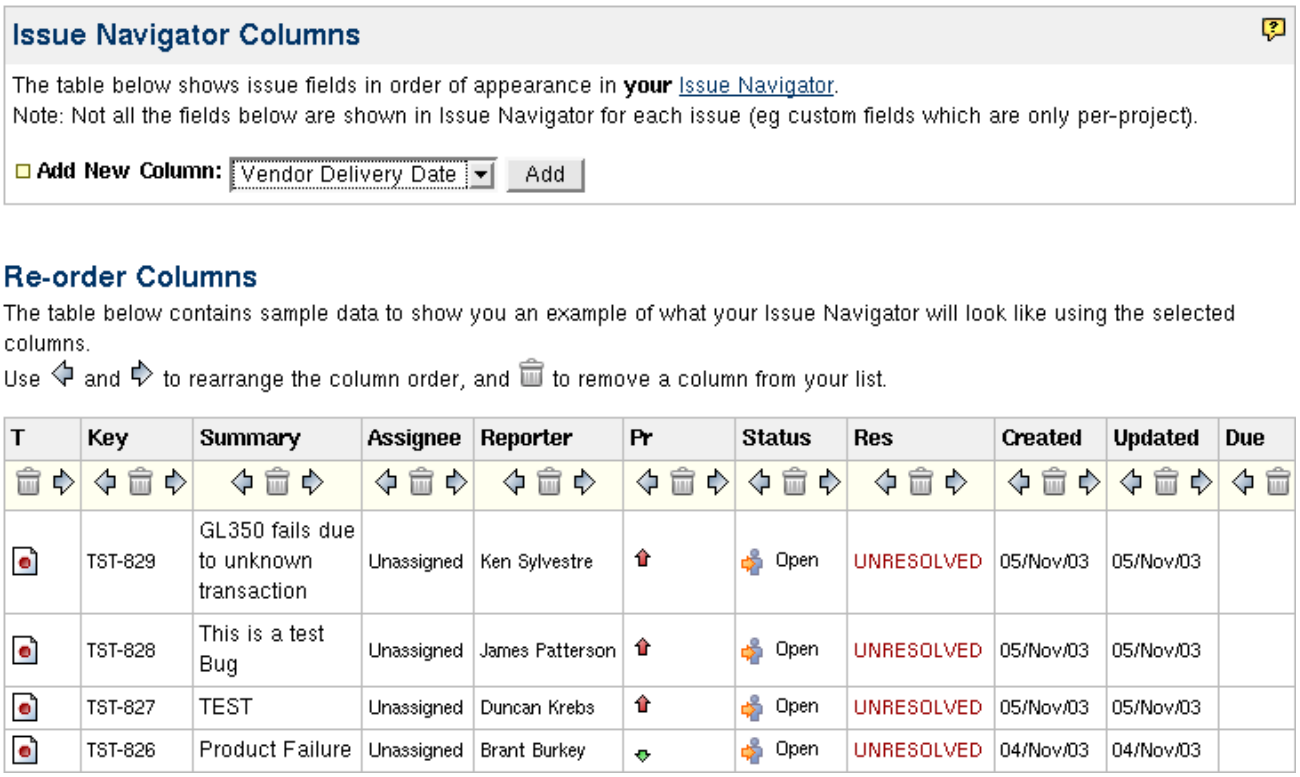
Note:

JIRA administrators can also select which views are available in the JIRA system, as views are configurable via plugins.

1.3.3.1. Configuring the default Issue Navigator Columns

JIRA **administrators** can configure the default navigator columns by navigating to the "Administration section" and then choosing "Navigator Columns" from the "Issue Fields" menu

on the left.



Configure Navigator Columns

- To move a column left or right, click on the left-arrow or right-arrow icon that appears under the column's heading.
- To remove the column from the list, click the bin icon which appears under the column's heading.
- To add a column to the list, select the issue field name from the drop-down box titled "Add New Column" and click the "Add" button. The column will appear as the right-most column in the list. It is then possible to position the column where desired using the arrow icons.
- If the column order has been modified from the defaults, users can restore the global defaults by selecting the "Restore Defaults" link (which will appear only if they have modified their issue navigator from the global defaults). When configuring the global defaults (only available to administrators), the link is called "Restore System Defaults", and when clicked restores the configuration that JIRA ships with by default.

Note:

- When configuring navigator columns, a user can only see columns for issue fields that have not been hidden.
- It is possible to add any of the existing custom fields to the navigator column list. When configuring the navigator columns a user can choose any custom field that they have permissions to see. That is, any custom field except those that are project-specific and apply only to a project that the user does not have permissions to browse. Some custom fields, even if selected to be part of the navigator columns, will not appear in the issue navigator for all issues. For example, project-specific custom fields will be shown only if the filter has been restricted to that project only. Issue type custom fields will only appear if the filter has been restricted to that issue type.
- When administrators are configuring default navigator columns, their permissions are ignored, so that they can add a project-specific custom field from a project that they do not have permissions to browse. The field would never be actually shown to users that do not have permissions to see it.

1.3.4. Configuring the Default Dashboard

The default dashboard is the screen that all JIRA users see the first time they login. Any users who have not added any dashboard pages as favourites also see the default dashboard.

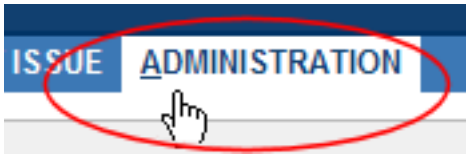
JIRA allows Administrators to configure the default dashboard. The portlets on the default dashboard can be re-ordered, switched between the left and right columns, additional portlets can be added, and some portlets can be configured.

All changes made to the default dashboard will also change the dashboards of all users currently using the default. However, portlets that users do not have permissions to see will not be displayed to them. For example, the 'Administration' portlet, although it may exist in the default dashboard configuration, will not be visible to non-admin users.

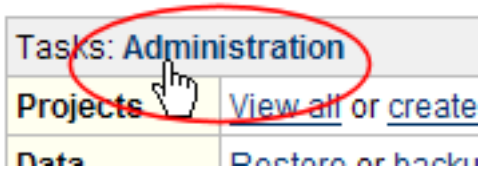
> **Note:**
> Portlets are the information boxes on the Dashboard. JIRA comes pre-configured with a set of standard dashboard portlets. It is also possible to develop custom portlets and plug them into JIRA using its flexible plugin system.

1.3.4.1. Adding and Configuring Portlets

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Click the 'Default Dashboard' link under 'Global Settings' on the left pane.
4. This will display the 'Configure Default Dashboard' screen, which consists of two selectable areas listing the current portlets.
5. If you wish to move and/or configure the existing portlets on the default dashboard, please see the instructions on using the configuration interface of a dashboard.
6. If you wish to add portlets to the default dashboard, please see the instructions on adding a portlet to a dashboard.

> **Note:**
> JIRA's default dashboard is limited to only one dashboard page. However, users can add multiple pages to their own dashboards if they wish.

1.3.5. JIRA Portlets

JIRA provides the ability to display summary information on project/issue data through the use of portlets. Each portlet can be configured to display project and issue details relevant to particular users. Portlets can be added to the dashboard - providing a central location for quick access to this information.

1.3.5.1. Preinstalled portlets

JIRA provides a set of standard portlets out-of-the-box:

Portlet	Description
Administration Portlet	The Administration portlet displays quick links to administrative functions conveniently on the dashboard.
Assigned To Me Portlet	The Assigned To Me portlet displays all open issues in all projects assigned to the current user viewing the dashboard.
Bugzilla ID Search Portlet	The Bugzilla ID Search portlet allows the user to search all JIRA issues for references to Bugzilla IDs.
Favourite Filters Portlet	The Favourite Filters portlet displays a list of all the issue filters that have currently been added by you as a favourite filter.
Filter Statistics Portlet	The Filter Statistics portlet displays the collection of issues returned from a specified filter broken down by a specified field.
2D Filter Statistics Portlet	The Two Dimensional Filter Statistics portlet displays statistical data based on a specified filter in a configurable table format.
In Progress Portlet	The In Progress portlet displays all issues that are currently in progress and assigned to the current user viewing the dashboard.
Introduction Portlet	The Introduction portlet displays a configurable introduction message on the dashboard.
Project Portlet	The Project portlet provides information and various filters related to a specified project on the dashboard.
Projects Portlet	The Projects portlet provides information and various filters related to all projects within JIRA.
Project Statistics Portlet	The Project Statistics portlet allows various per-project statistical data to be displayed on the dashboard.
Project Table Portlet	The Project Table portlet displays all the project names in a table in the dashboard.
Quick Links Portlet	The Quick Links portlet displays a number of useful links to issues associated with the current user. Each link directs the user to the Issue Navigator, displaying the relevant issues such as Reported Issues, Voted Issues and Watched Issues.
Road Map Portlet	The Road Map portlet shows versions which are due for release within a specified period of time, and a summary of progress made towards completing the issues in those versions.
Saved Filter Portlet	The Saved Filter portlet displays the results of a specified issue filter on the dashboard.

Text Portlet	The Text portlet displays a configurable HTML text on the dashboard.
Voted Portlet	The Voted Issues portlet shows issues for which you have voted.
Watched Portlet	The Watched Issues portlet shows issues which you are watching.

1.3.5.2. Extension portlets

Other portlets are available as plugins on the JIRA Extensions site. These plugins include:

- Bamboo plugin
- Charting plugin
- Calendar plugin
- Timesheet plugin

Should you wish to you use these plugins, you need to first install them (using the instructions provided with each plugin) then enable them.

1.3.5.3. Creating New Portlets

New portlets can be created by writing a Java class, a template and an XML descriptor file, packaged as a JIRA plugin. See How To Create a JIRA Portlet for more information. Many sample portlets, including source, are available in the JIRA Extensions space.

1.3.6. Managing JIRA's Plugins

Plugins allow you to customise and extend the functionality of JIRA in a variety of ways, including:

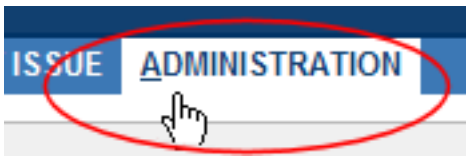
- modifying the availability of 'issue operations' links ('Create Issue', etc)
- creating new dashboard portlets
- creating new reports
- creating new types of custom fields
- configuring renderers for rich-text fields
- customising workflow

JIRA ships with a number of preinstalled plugins, and many more are available for download from the JIRA Extensions site. You can also create your own plugins (please visit the JIRA Development Hub for details).

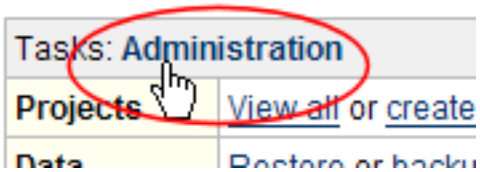
Installed JIRA plugins can be enabled or disabled as follows:

1.3.6.1. Enabling a JIRA Plugin

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

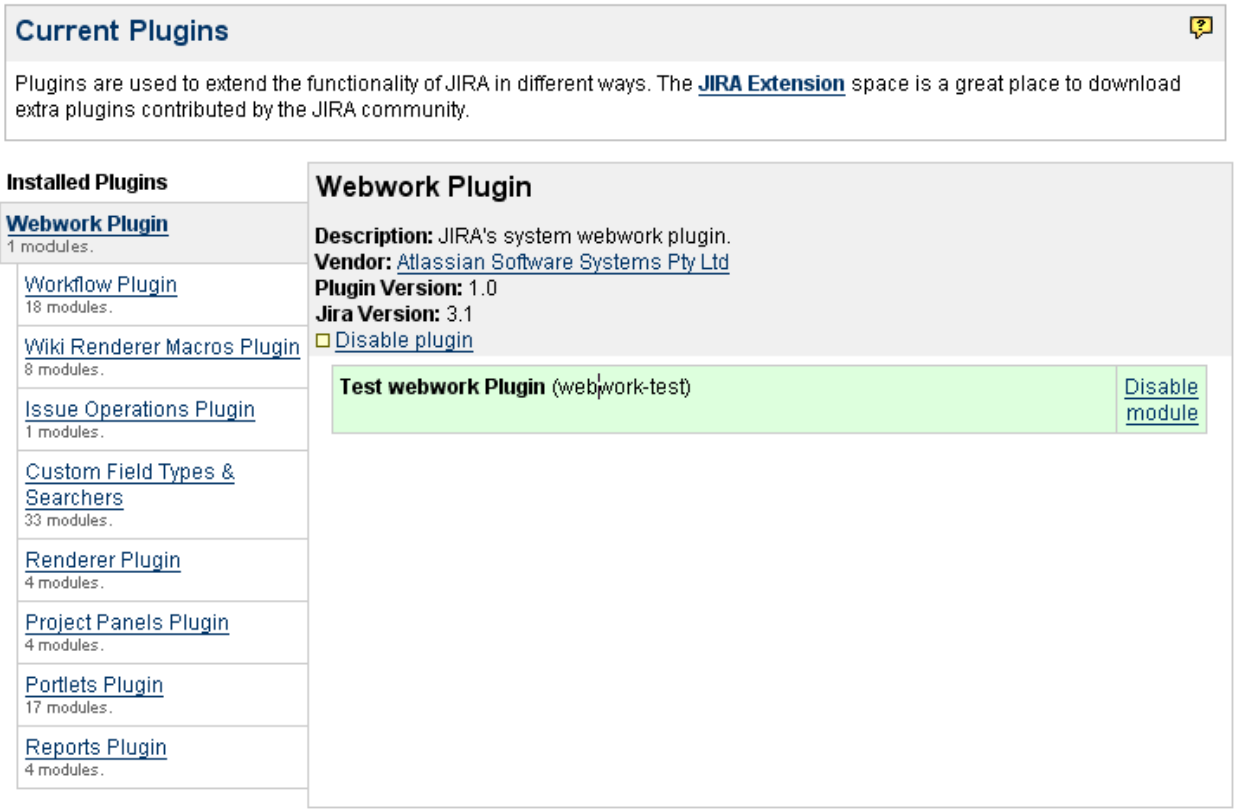


Link to Administration section



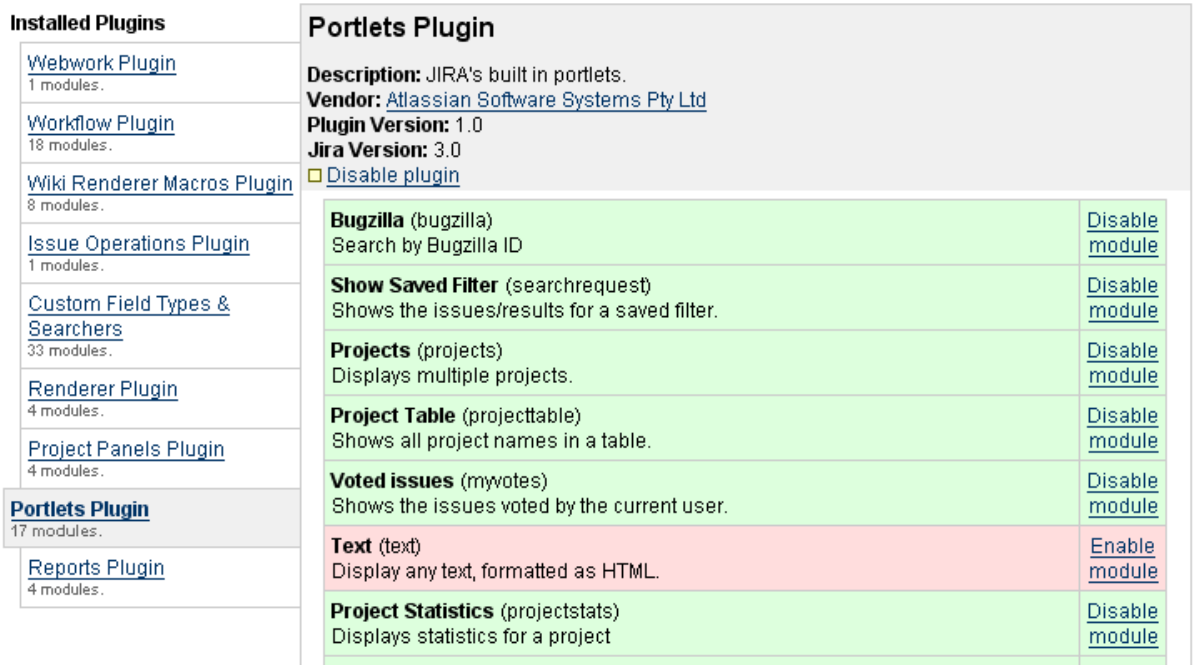
Link to Administration section

3. On the panel on the left, under the title "System", click on the link labelled "Plugins".
4. This will bring up the Current Plugins page. The page lists all the installed plugins



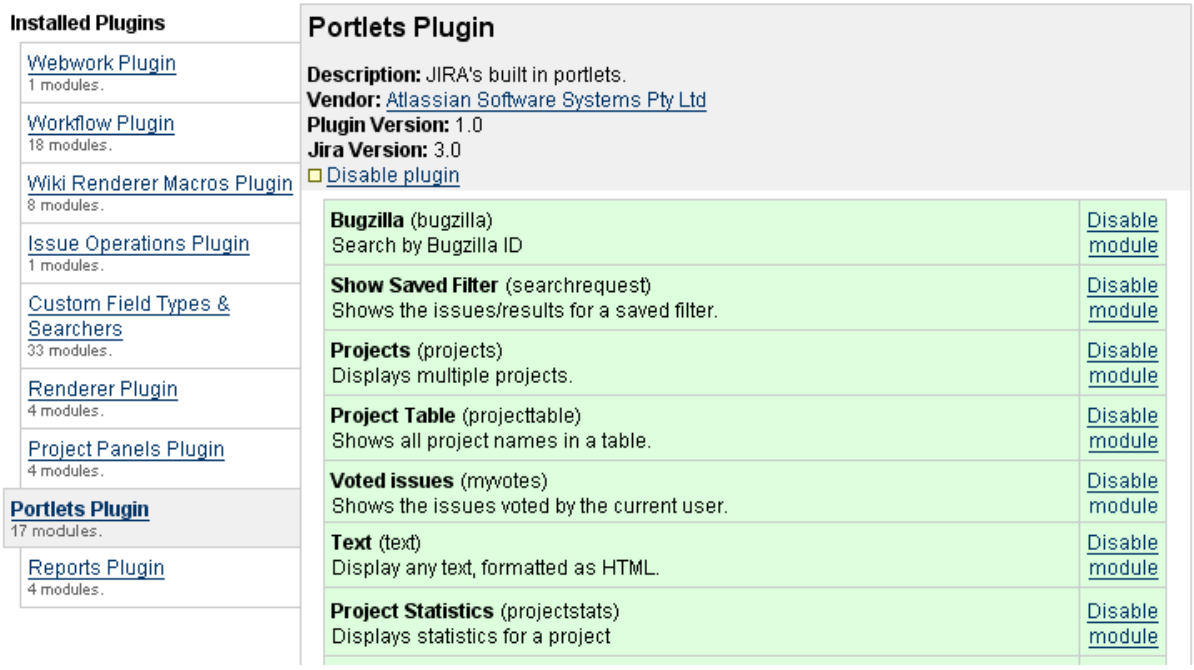
'Current Plugins' Page

5. Select the Plugin that contains the module you want to enable. (Eg. to enable the Text Portlet, select 'Portlets Plugin')



Text Portlet Module Disabled

6. Click on 'Enable Module' next to the 'Text' portlet. The text portlet should now be enabled as shown:



Text Portlet Module Enabled

1.3.6.2. Disabling a JIRA Plugin

1. Follow the above steps for enabling a plugin, but click 'Disable Module' instead.

1.3.7. Dynamic Announcement Banner

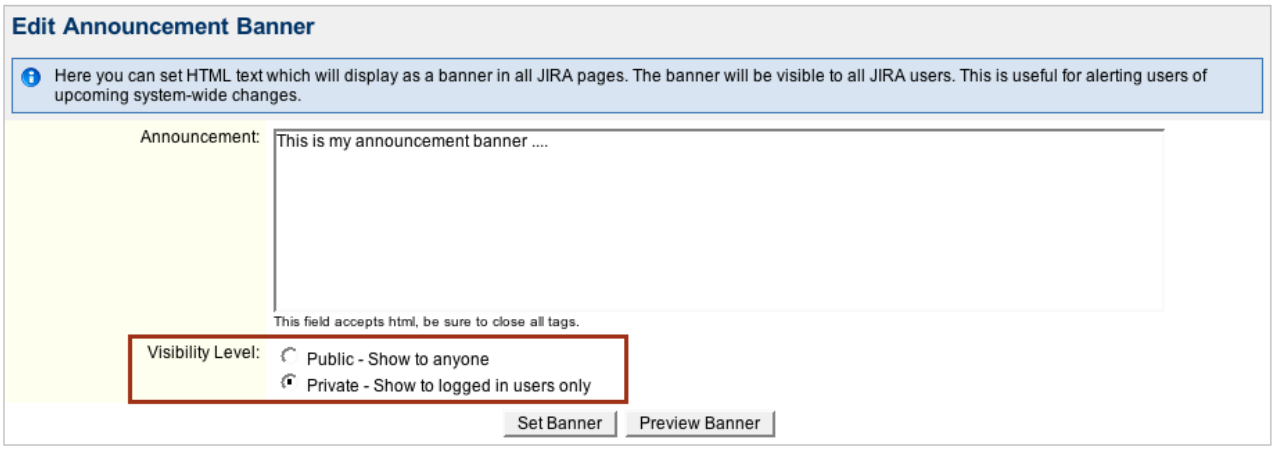
Administrators can configure an **announcement banner** to display pertinent information on all JIRA pages. The banner can be used to relate important information (e.g. scheduled server maintenance, approaching project deadlines, etc.) to all users. Further, the banner visibility level can be configured to display to all users or just logged-in users.

The banner can be configured to contain HTML text.

1.3.7.1. Configure Announcement Banner

1. Navigate to the JIRA **Administration** section.
2. Select the **Announcement Banner** under the **Options & Settings** sub-menu.
3. Enter the required text in the **Announcement** field.
4. Select the required visibility level for the banner.
5. Click **Set Banner**.

Depending on the visibilty level selected, the banner will become visible throughout JIRA.



Configure Announcement Banner

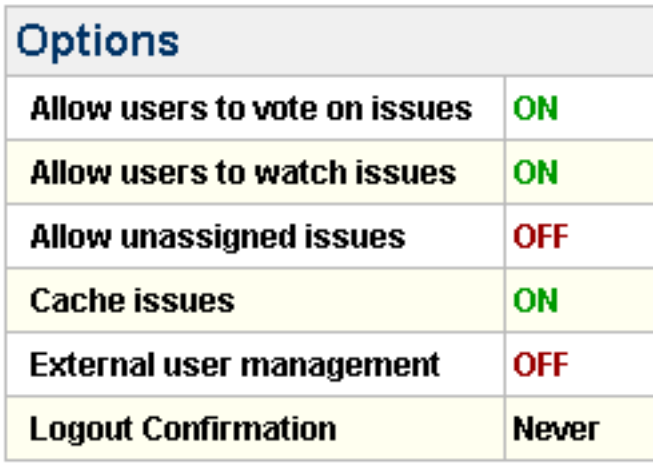
1.3.7.2. Banner Visibility Mode

The announcement banner visibility level can be configured to specify to whom the banner will be displayed. There are two modes:

- **Public** - the banner is visible to everyone
- **Private** - the banner is visible to logged-in users **only**

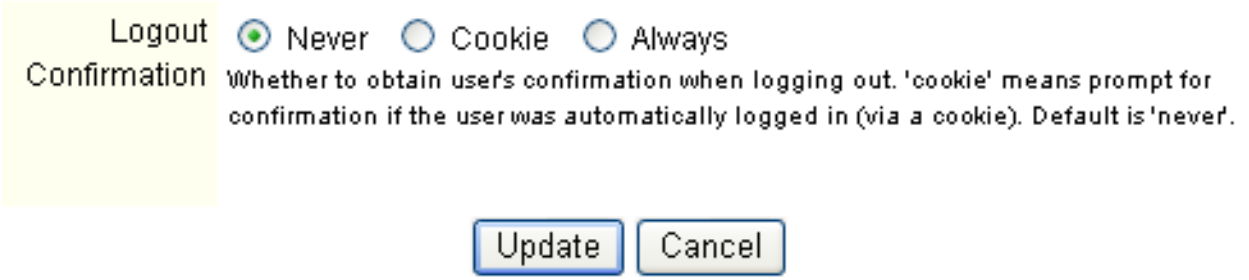
1.3.8. Logout Confirmation

Administrators can now configure JIRA to prompt users with a confirmation before logging them out. You can adjust this setting by going to the Administration page and then clicking "General Configuration" link found under "Options and Settings" on the left pane.



Logout Confirmation Setting is switched off by default

As shown above, JIRA will never prompt users for logout confirmation by default. To change this, click on "Edit Configuration".



Logout Confirmation Setting

The "Never" and "Always" settings are self-explanatory. When set to "Cookie", users will only be prompted if they have logged in using a cookie (ie. checked the box reading "Remember my login on this computer" before they logged on).

1.4. Email

1.4.1. Email Overview

JIRA can send email notifications to users when significant events occur.

1.4.1.1. Enabling Email Notifications

To enable email notifications in JIRA,

1. Configure an SMTP Mail Server.
2. Configure a notification scheme and associate it with the appropriate projects.

It is possible to customise your email content. The email address from which notifications are sent can also be configured for each project.

1.4.1.2. Disabling Email Notifications

To disable email notifications for a project, you can remove the notification scheme from the project by editing the project and selecting 'None' as the project's notification scheme.

Alternatively, you can edit the notification scheme so that no emails are sent.

1.4.1.3. Configuring a Project's Email Address

It is possible to configure the project email address that notifications are sent from.

By setting the 'Sender' email address for a project, all notifications will be sent from this address. This setting is specific to the project selected and will not affect the configuration of the other projects. The default address specified in the SMTP Mail Server configuration is used as the default "sender" address for all projects.

The "sender" email address can be configured as follows:

1. From the Administration view, select "Projects" to view all projects. Select the project to be configured.
2. Select "Edit Configuration" from the "Mail Configuration" entry in the project detail list.
3. Enter a valid email address in the "sender" field and click "Confirm" to complete the process. This email address will now be used as the "sender" address in all notifications for this project.

4. The default email address as specified in the SMTP Mail Server can be reinstated by clicking the "Reset" button.

> **Note:**
> This option is not accessible unless a SMTP Mail Server has been previously configured.

1.4.1.4. Email Recipients

For each event notification, JIRA will only send the first encountered email intended for a recipient. Hence, in the case where a user is included in two or more recipient lists (e.g. Project Lead and Current Reporter) for one event notification, the user will only recieve the first encountered email notification. JIRA will log the fact that this user was on multiple recipient lists.

1.4.1.5. Email HTML Formatting

Each JIRA user can specify in their Profile Preferences whether to send outgoing emails in text or HTML format; JIRA administrators can specify a defaut email format under 'User Defaults' in the Administration menu.

Since Jira 3.6.1, the HTML email format was improved to accomodate internationalised words in the 'Issue Details' section. However, due to Internet Security Settings, which prevent automatic download of images, the HTML e-mail may not be correctly formatted. For example, the summary column on the left may appear too wide. It is possible to correct the formatting by accepting to download these images. On some e-mail clients it is possible to do this in two different ways:

1. per each email
   - **Mozilla Thunderbird** - by clicking on the *"Show Images"* button above the e-mail
   - **Microsoft Outlook 2003** - by clicking on the *"Click here to download pictures. To help protect your privacy, Outlook prevented automatic download of some pictures in this message."* message above the e-mail
   - **Microsoft Outlook 2000** - does not have this option, it always downloads images
   - **Microsoft Outlook Express 6** - by clicking on the *"Some pictures have been blocked to help prevent the sender from identifying your computer. Click here to download pictures."* message above the e-mail
2. configuring the e-mail client
   - **Mozilla Thunderbird 1.5** - Navigate to **Tools -> Options -> Privacy -> General** tab and ensure that *"Allow remote images if the sender is in my:"* option is checked and note which address book is selected. Then return to the e-mail sent from JIRA, right-click on the sender's e-mail address and choose *"Add to address book..."* option, adding this contact to the same address book as was selected in the Privacy options
   - **Microsoft Outlook 2003 and Outlook Express 6** - Navigate to **Control Panel -> Internet Options**. On the Security tab, add JIRA's base URL to the trusted sites

1.4.2. Notification Schemes

JIRA can generate email notifications for various events that happen during the issue lifecycle. Notifications are defined within a *notification scheme* (see below), which associates particular events with particular email recipients. The notification scheme is then assigned to a particular project; note that you can use the same notification scheme for more than one project.

The events which can generate email notifications are:

Issue Created:	An issue has been entered into the system.
Issue Updated:	An issue has had its details changed.
Issue Assigned:	An issue has been assigned to a new user.

Issue Resolved:	An issue has been resolved (usually after being worked on and fixed).
Issue Closed:	An issue has been closed. (Note that an issue may be closed without being resolved; see Statuses).
Issue Commented:	An issue has had a comment added to it.
Issue Comment Edited:	An issue's comment has been modified.
Issue Reopened:	An issue has been re-opened.
Issue Deleted:	An issue has been deleted.
Issue Moved:	An issue has been moved into this project.
Work Logged On Issue:	An issue has had hours logged against it (i.e. a worklog has been added).
Work Started On Issue:	The Assignee has started working on an issue.
Work Stopped On Issue:	The Assignee has stopped working on an issue.
Issue Worklog Updated:	An entry in an issue's worklog has been modified.
Issue Worklog Deleted:	An entry in an issue's worklog has been deleted.
Generic Event:	The exact nature of this event depends on the workflow transition(s) from it was fired.
Custom Event(s):	The exact nature of these events depends on the workflow transition(s) from which they were fired.

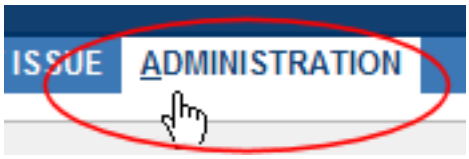
Note that email notifications will only be sent to people who have permission to view the relevant issue — that is, people who:

- have the 'Browse Project' permission for the project to which the issue belongs; and
- are members of any Issue security levels that have been applied to the issue.

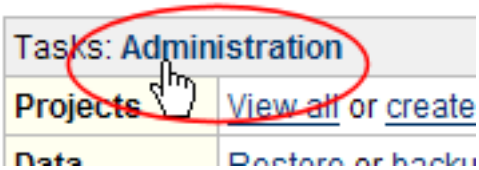
Also note that JIRA can only send email notifications if SMTP email has been enabled (see Email Overview).

1.4.2.1. Creating a Notification Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

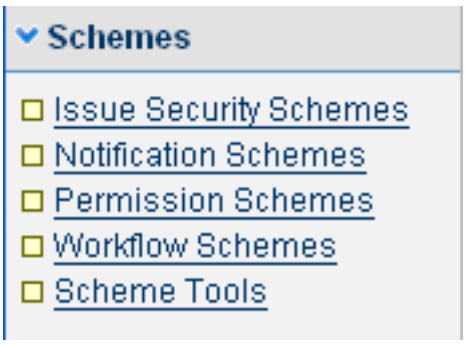


Link to Administration section



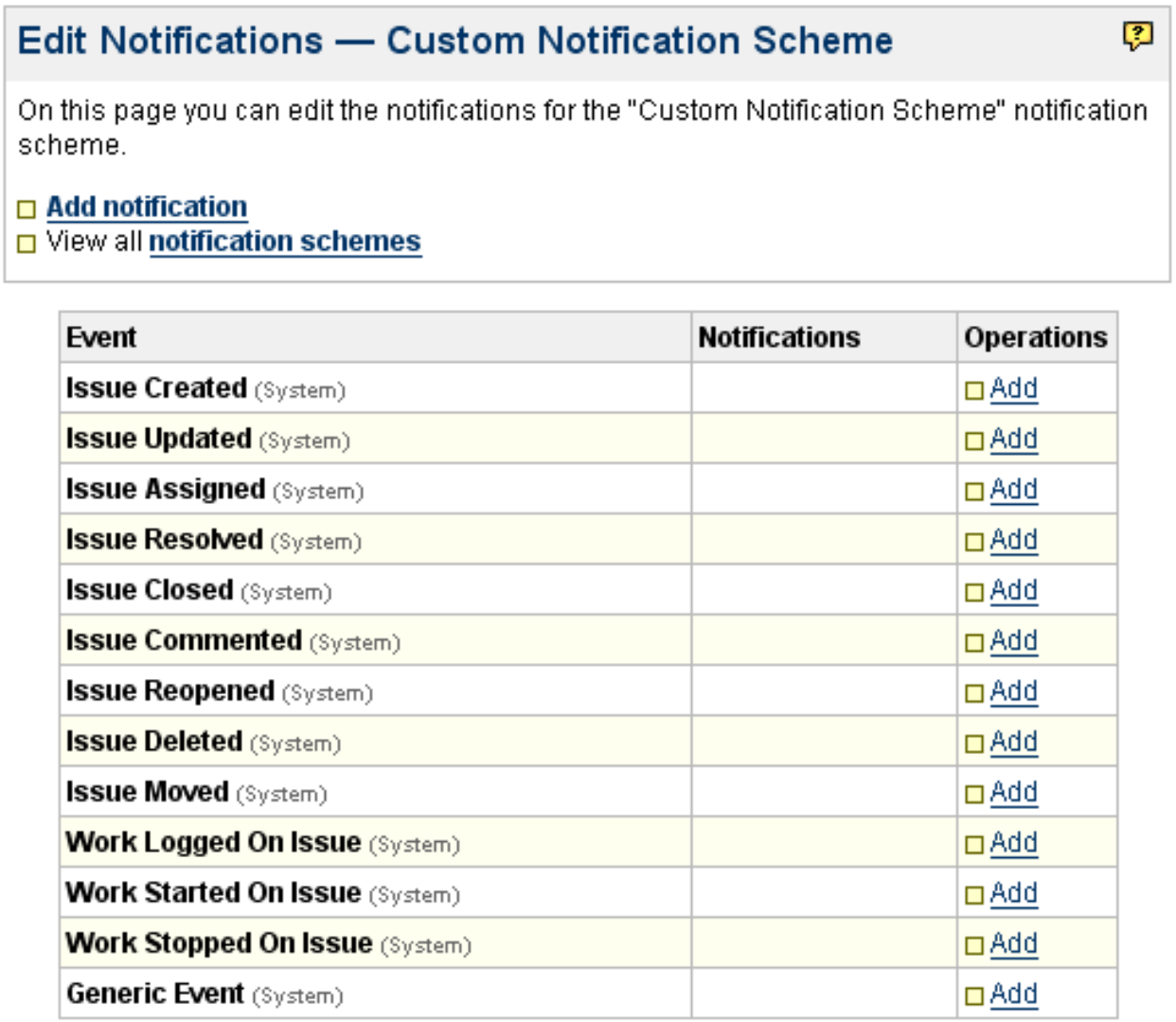
Link to Administration section

3. On the panel on the left, under the title "Schemes", click on the link labelled "Notification Schemes".



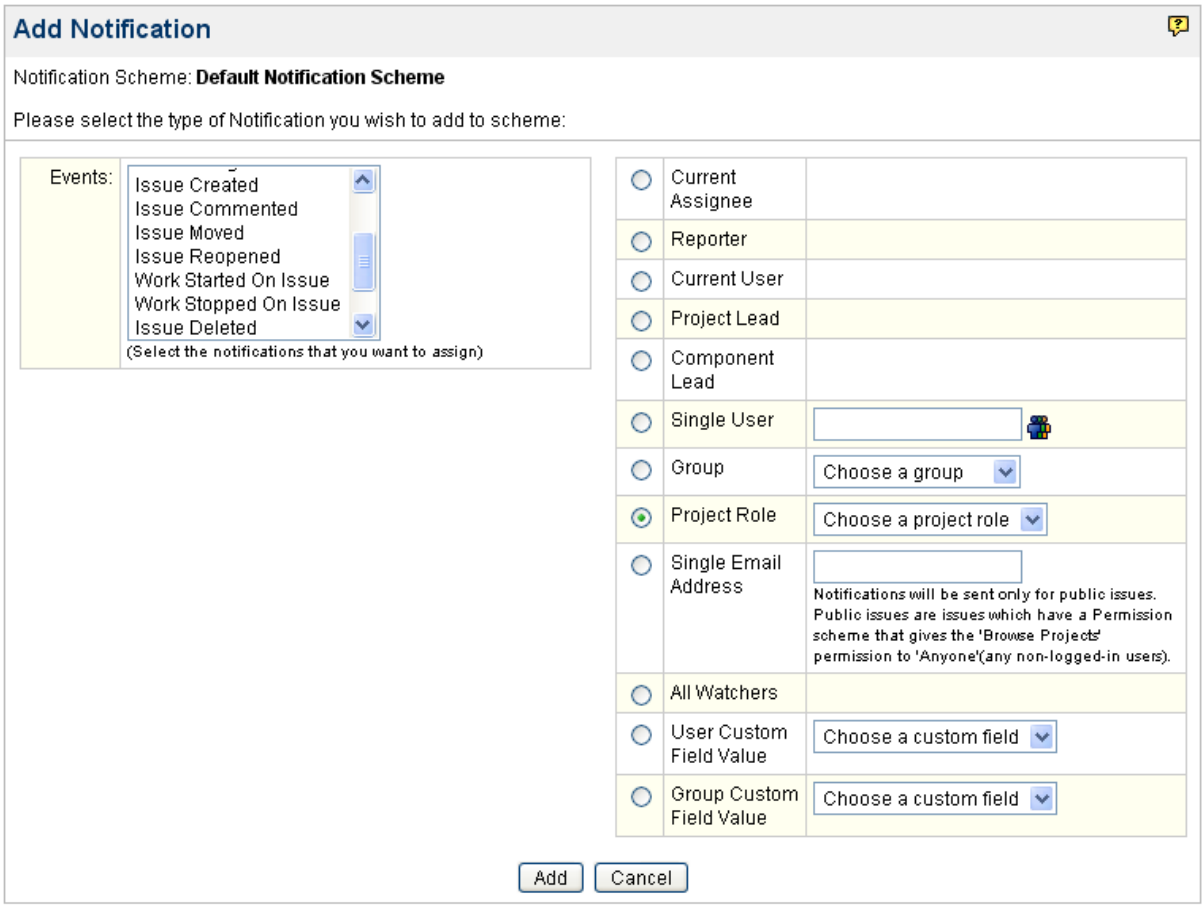
View Notification Schemes

4. This will display the "Notification Schemes" page. This page lists all of the notification schemes that JIRA currently has. Click on the "Add Notification Scheme" link.
5. In the "Add Notification Scheme" form, enter a name for the notification scheme, and a short description of the scheme. Click on the "Add" button.
6. You are then shown the "Edit Notifications" page. This page lists all of the above mentioned issue life cycle events, along with whom should be notified. It is currently empty.
7. Click on the "Add" link in the appropriate life cycle event row.



Editing a new notification scheme

8. This will display the "Add New Notification" page. Here you can choose who to notify, from the list of alternatives.



Adding a notification to a scheme

Where:

- **Current Assignee** is the user assigned to the issue.
- **Current Reporter** is the user who originally created the issue
- **Current User** is the user who performed the action triggering the event in question.
- **Single Email Address** is any email address that you wish to alert.

Note:

A Single Email Address notification will only be sent if the issue is publically viewable (as the email address of a non-JIRA user could be specified, in which case a security check is not possible). Publically viewable issues are issues which have a Permission scheme that gives the 'Browse Projects' permission to 'Anyone' (any non-logged-in users).

- **User Custom Field Value** is any custom field value of type *User Picker* or *Multi User Picker* that may have been associated with issues. An example of where this can be useful, you have a custom User field called Tester, you have the tester notified when an issue is resolved.
- (the rest are hopefully self-evident)

Note that project roles are useful for defining specific team members for each project. Referencing project roles (rather than groups) in your notifications can help you minimise the number of notification schemes in your system.

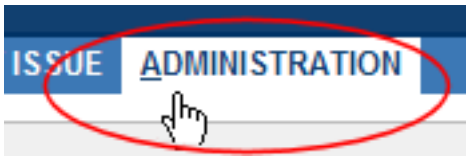
9. After selecting the appropriate option, and filling in any required information for that option, click the "Add" button.
10. You will be taken back to the "Edit Notifications" page, with the notification you just

specified now listed against the appropriate issue life cycle event.

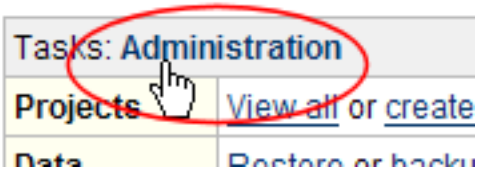
11. Repeat steps 7 through 11 until you have specified all the notifications you want to happen.
12. If you make a mistake, or you would like to remove who is being notified, simply do so by clicking on the "delete" link beside the person/group/role.

1.4.2.2. Assigning a Notification Scheme to a Project

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

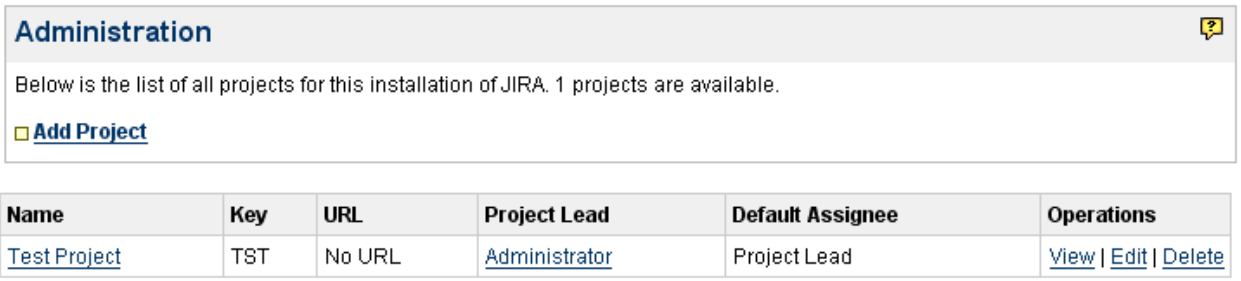


Link to Administration section



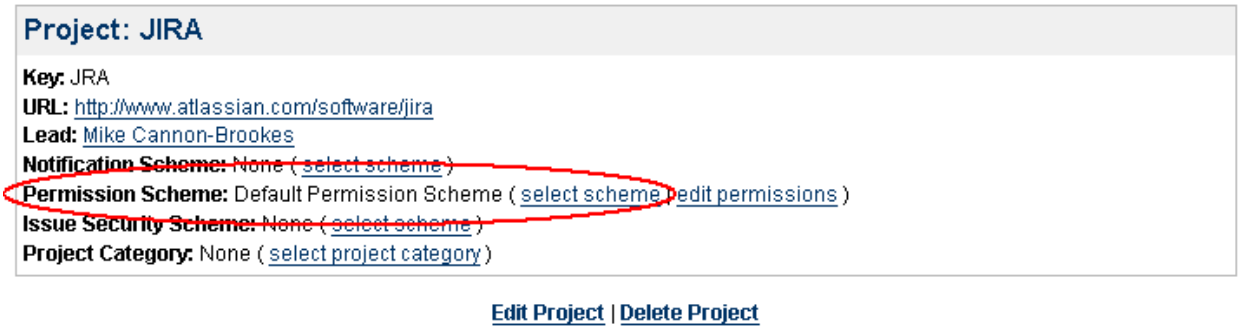
Link to Administration section

3. A list of projects is displayed



List of Projects

4. Select the project you want by clicking on the project name. This will display the project details
5. Click on the "select scheme" link beside the Notification Scheme caption.



Project Details

6. This will bring up a list of notification schemes. Select the notification scheme that you want to associate with this project.
7. Click the "Associate" button to associate the project with the notification scheme.

Note:

See also Minimising the number of Permission Schemes and Notification Schemes.

1.4.3. Customising email content

JIRA generates emails in reaction to events using a templating engine. The templating engine is Apache Jakarta's Velocity. This is a relatively easy to use templating language that can pull apart java objects in useful ways. The mails are generated inside JIRA by invoking Velocity with a set of objects of relevance to the event.

To customise email content, please follow this procedure.

1. Open up your JIRA distribution, and navigate to the following paths:
   - **Standalone:** atlassian-jira-2.0\atlassian-jira\WEB-INF\classes\templates\email\
   - **Source:** jira\src\etc\java\templates\email\
   - **WAR:** webapp\WEB-INF\classes\templates\email\
2. Under this directory there are two directories, html and text. The html subdirectory contains the templates used to create emails in html, while the text directory the plain text mail outs. The templates are named after the event that will trigger the email.
3. Bring the template up in your favourite text editor. Referring to the JIRA template documentation and Velocity Users Guide, make the customisations you want.
4. Restart JIRA.

Note:

See also Adding Custom Fields to Email.

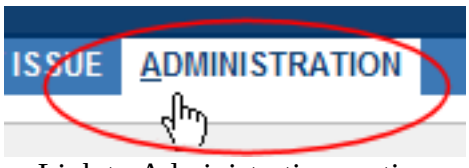
1.4.4. Creating Issues and Comments from Email

JIRA can be configured to automatically create issues or comments based on incoming emails. This is especially useful in a helpdesk or support scenario, where users send support queries via email, which you wish to track with JIRA. Subsequent emails about the issue, for example responses to Email Notifications, can be automatically recorded as comments. Additionally, any attachments in the emails can be automatically attached to the issue (with appropriate configuration).

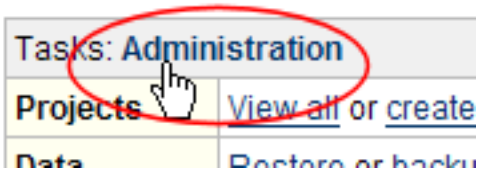
To set up issue and comment creation via email, you will need to create a mail account on your server (usually one mail account for each project). For example, for the 'ABC' project, you might establish an account `abc-issues@yourcompany.com`. This mail box should be accessible via POP, IMAP, or on the local filesystem. JIRA will periodically scan this mail box, and appropriately create issues or comments for any emails it finds, and — optionally — create new user accounts for senders not previously seen (note that this is not possible if you are using External User Management).

Once you have established a mail account, here is how to configure JIRA to periodically scan it (POP access assumed):

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

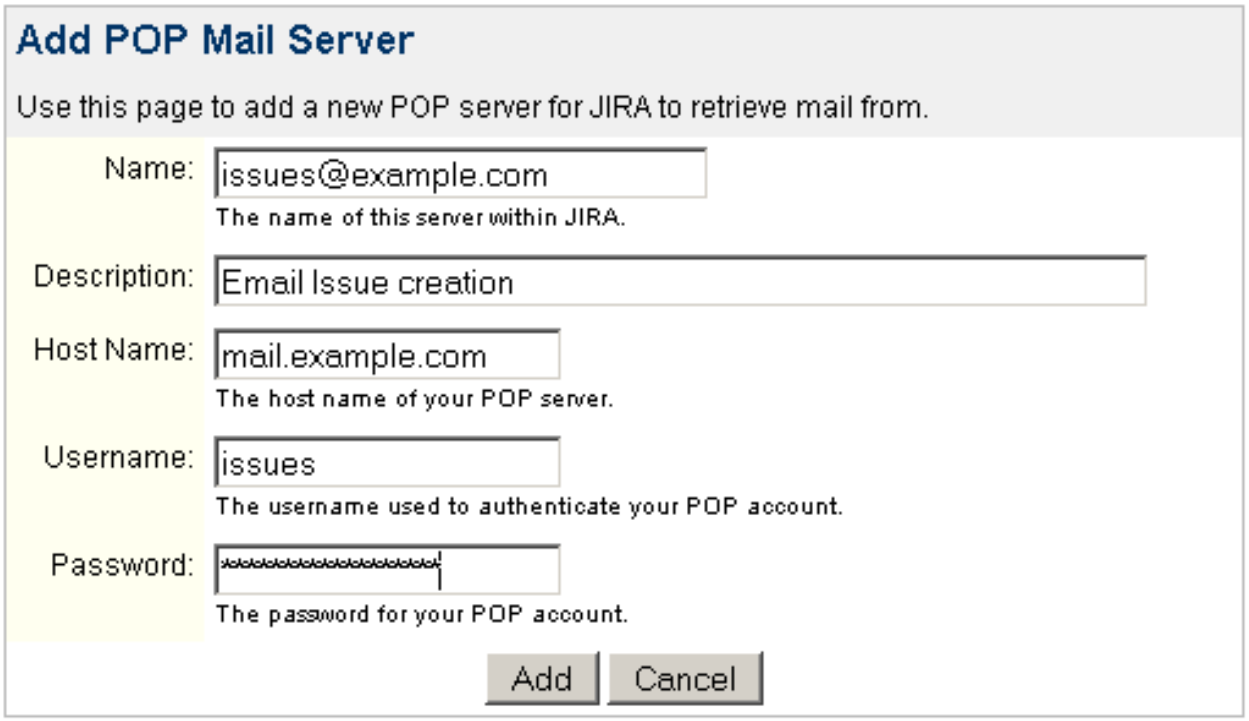


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title "Global Settings", click on the link labelled "Mail Servers".
4. Click on the "Configure new POP mail server" link.
5. This will bring up the "Add POP Mail Server" page.



Add pop server

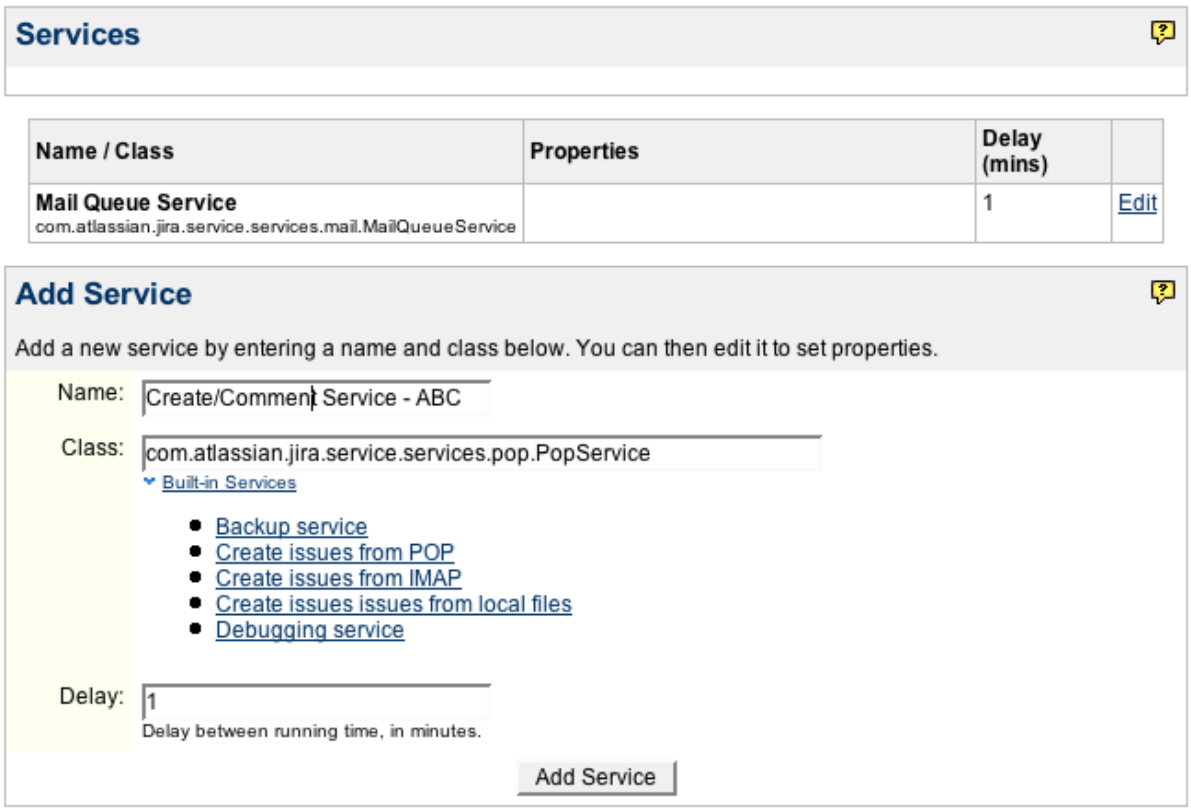
Fill in as follows:

- **Name:** put a short descriptive name, possibly just the email address that will be collected by this service
- **Description:** put a short phrase that describes this service, probably 'Email Issue Creation/Comments for <Project>'.
- **Hostname:** put the name of your POP server
- **Username** and **Password** use the email account details as created in step 1.

Note that the use of SSL is specified later in the service, not here in the Mail Server.

If you need to set a non-standard port, this will need to be done by setting a `-Dmail.pop3.port=<port>` property (instead of `pop3` you can specify `pop3s`, `imap` or `imaps`). See JRA-11037 for more on this.

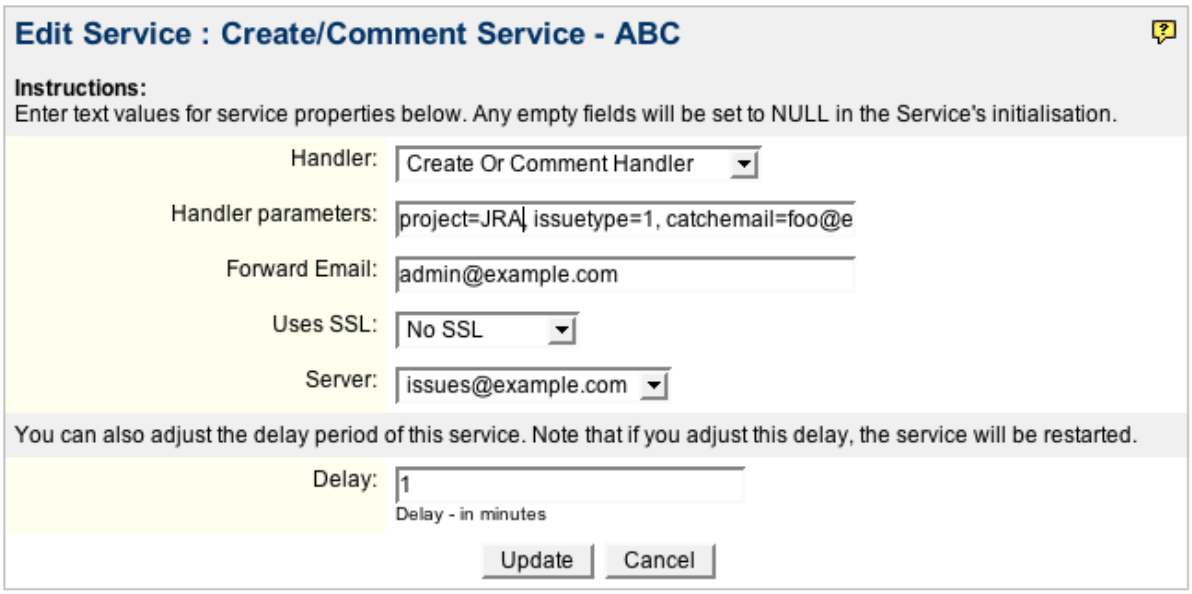
6. This should bring you back to the Email Servers page, where you should see a new POP server listed. You can edit and delete this server here.
7. On the panel on the left, under the title "System", click on the link labelled "Services".
8. This will bring up the "Services" page. It lists the current services running on this system. On a vanilla system there should be one service running - **Mail Queue Service**. You cannot delete the **Mail Queue Service**. Additionally, if you have enabled the option to automatically backup JIRA's data, you will also see the **Backup Service** listed here too.
9. Fill in the "Add Service" form as follows:



Add service

For **Name** enter a descriptive name, eg "Create Issue/Comment Service for <Project>". For **Class**, select the appropriate option presented in the drop down list, or enter **com.atlassian.jira.service.services.pop.PopService**, and the **Delay** is best left as 1 minute. Click Add Service.

10. This will bring up the "Edit Service" screen to configure the service.



Editing a service

For Handler, select "Create Or Comment Handler" from the drop down box. Set Handler parameters to something like:

```
project=JRA, issuetype=1, createusers=true, bulk=forward
```

Further details on the **handler parameters** are available below.

If you choose to connect over SSL, you will need to import and verify the server's SSL key before JIRA will be able to connect. See Connecting to SSL Services for more information.

The **Forward Email** parameter specifies an email address to which error notifications and (optionally) unhandled emails can be forwarded (see "bulk" parameter below). Any unhandled mails or failures encountered in this process are logged and forwarded in an email to this address.

Click the "update" button and the service will be in effect.

1.4.4.1. Issue/Comment Creation

JIRA examines the email subject and the in-reply-to message for an existing issue reference to determine whether a new issue or comment should be created. A new issue is created if an existing issue reference is not found - otherwise, a comment is added to the issue referenced in the email. The email to foo@atlassian.com will be processed as follows:

- Issue Creation:
  - The subject of the email will become the issue summary.
  - The body of the email will be the issue description.
  - A bug (since issue type has been set to 1 in this example) will now be created for project "JRA" with the above information.
  - Any attachments to the email will become attachments to the issue (assuming attachments have been enabled in JIRA). Note that, to ensure compatibility with various operating systems, any of the following characters in the filename will be replaced with an underscore character: \, /, ", %, :, $, ?, *, <, |, >.
- Comment Creation:
  - The body of the email will become a comment on the issue
  - Any attachments to the email will become attachments to the issue (assuming attachments have been enabled in JIRA)

Note:

The **Subject** of the email becomes the issue summary. As all issues require a summary, each email intended for issue creation should include a **Subject**.

1.4.4.2. Handler Parameters

project parameter is the project key.

Note:

The **project** parameter is only relevant for issue creation, not for issue commenting. If an email contains an issue key in the email subject, and that issue exists in the JIRA instance, the handler will add the email as a comment on the issue, regardless of which project the issue is in.

These are the numbers associated with the default **issue types**:

- **Bug:** `issuetype=1`
- **New Feature:** `issuetype=2`
- **Task:** `issuetype=3`
- **Improvement:** `issuetype=4`
- **Sub-task:** `issuetype=5`

You can use the method described here to determine what numbers are mapped to your issue types.

Besides **project** and **issuetype**, the following parameters are allowed:

createusers

If **createusers** is set to true, people who don't currently have an account in JIRA will have it created for them. In JIRA Enterprise, this allows the creator to be notified of subsequent updates to the issue, by configuring the notification scheme to notify the 'Reporter' of updates.

reporterusername

This sets which user will be the "reporter" of created issues, for emails whose sender does *not* match that of an existing user. Normally JIRA will ignore emails from addresses not matching an existing user. For instance, to allow anonymous users to create issues via email, you can create an anonymous user or dummy account on JIRA and set the **reporterusername** to point to this account. When the "reporterusername" parameter is specified, the "from" address of the email is added at the end of the comment of the issue, so you can identify the sender.

notifyusers

This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added.

ccassignee

If the To, Cc, or Bcc field of an email contains the address of a user already present in JIRA, then by default JIRA will **assign** the issue created from the email to that user. JIRA will attempt to assign the issue to a user from the To field first, then the Cc field and finally the Bcc field, if it cannot find a match in the To or Cc fields. If you do not wish JIRA to automatically assign issues in this way, then set **ccassignee** to **false**.

bulk

This parameter determines how to handle "bulk" emails (those sent by an automated service, notably JIRA itself), indicated by a "Predecence: bulk" header or an "Auto-Submitted" header that is not set to "no". Possible values are:

1. *ignore* - Ignore the email and do nothing
2. *forward* - Forward the email to the address set in the "Forward Email" text field
3. *delete* - Delete the email permanently

It is generally a good idea to set **bulk=forward** and set a Forward Email address, to prevent mail loops between JIRA and another automated service (eg. another JIRA installation).

catchemail

This causes JIRA to only process emails sent *to* the specified email address. All other emails are ignored. This is useful if you have multiple aliases for the same email Inbox, eg. `foo-support@example.com` and `bar-support@example.com` aliases for `support@example.com`, and you want one email service each, eg. to create issues in FOO and BAR projects respectively. Please note that **this parameter is rarely useful**, and should not be confused with the more common **reporterusername**.

You can only specify one catch email address and one issue type per listener.

1.4.4.3. Other Handlers

For more information on other handlers that are shipped with JIRA please refer to this document.

1.4.4.4. Email preprocessing

For production use, we recommend that you set up the following email preprocessing:

- Ensure mail is sent to a backup folder, so there is a record of what JIRA processed.
-

If the POP box contains email replies to JIRA notifications, set up rules filtering out email auto-replies and bounces.

If you do not do this, there is a strong possibility of mail loops between JIRA and autoresponders like vacation scripts. JIRA sets a 'Precedence:bulk' header (unless you've disabled this) and an 'Auto-Submitted' header on outgoing email, but some autoresponders ignore it.

There is no bulletproof way of detecting whether an email is a bounce or autoreply. The following rules (in procmail format) will detect most autoreplies:

```
^From:.*mailer-daemon@
^Auto-Submitted:.auto-
^Content-Type:\ multipart/report;\ report-type=delivery-status
^Subject:\ Delivery\ Status\ Notification
^Subject:\ Undeliverable
^Subject: Returned Mail:
^From:\ System\ Administrator
^Precedence:\ auto_reply
^Subject:.*autoreply
^Subject:.*Account\ signup
```

Even with these rules, you may encounter autoreplies with nothing in the headers to distinguish it from a regular mail, In these cases you will just need to manually update the filters to exclude that sender.

- Set up a filter to catch email with huge attachments. JIRA uses the standard JavaMail library to parse email, and it quickly runs out of memory on large attachments (eg. > 50Mb given 512Mb heap). As the unhandled mail isn't deleted, it will be reprocessed (causing another OutOfMemoryError) each time the mail service runs.

  In practice this problem is rarely seen, because most mail servers are configured to not accept email with huge attachments. Unless you're sure yours won't pass a huge attachment on to JIRA, it is best to configure a filter to prevent JIRA encountering any huge attachments.
- Set up spam filtering rules, so JIRA is not having to process (and possibly create issues from) spam.

1.4.4.5. Troubleshooting

A useful tip for debugging mail-related problems in JIRA is to set the `-Dmail.debug=true` property on startup. This will cause protocol-level details of JIRA's email interactions to be logged. Additionally, turning up JIRA's log level will show when the service is running and how mails are processed.

If you find some incoming emails simply disappear, check that you haven't accidentally started a second copy of JIRA (eg. in a staging environment) which is downloading and deleting mails. See the Restoring Data page for flags you should set to prevent mail being processed.

If you receive email with non-ASCII attachment names, particularly from Thunderbird users, you will need to configure JavaMail to support RFC 2231-encoded attachments. See JRA-12525 for details.

1.4.4.6. Additional Resources

- Creating comments and issues via email tutorial video — Watch this short tutorial video to see how to create a comments and issues in JIRA via email. Please note the JIRA version and JIRA edition of the tutorial video before watching.

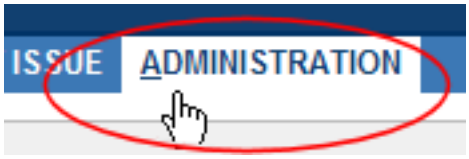
1.4.5. Configuring JIRA to send SMTP mail

To enable JIRA to send notifications about various events, you need to first configure JIRA to send

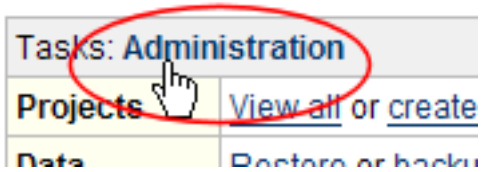
SMTP email.

1.4.5.1. 1. Define the SMTP Mail Server

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



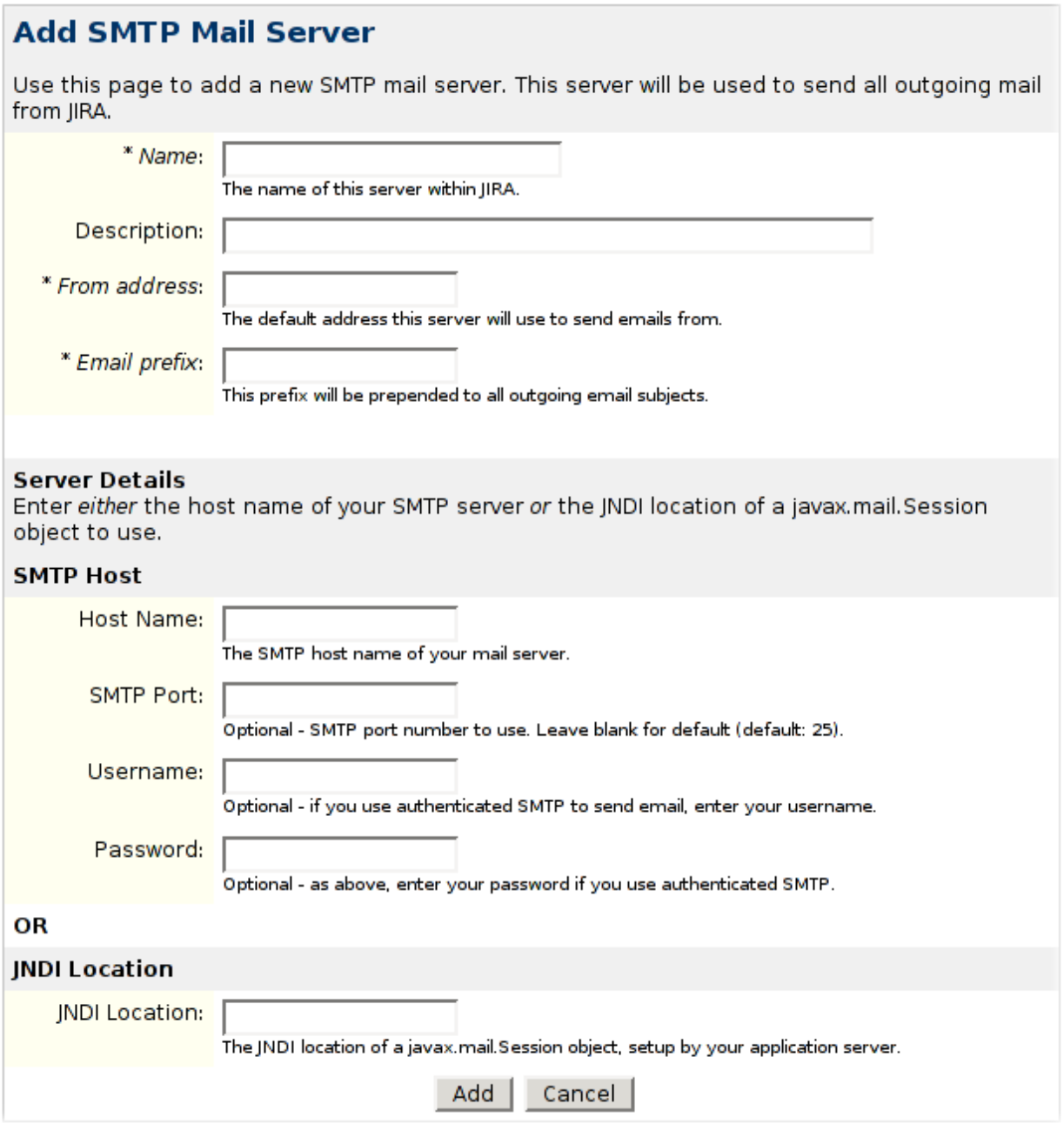
Link to Administration section



Link to Administration section

3. Click "Mail Servers" in the left-hand column (under "Global Settings").
4. Click "Configure new SMTP mail server".
5. This will display the "Add SMTP Mail Server" screen. Complete the top section as follows:

Name	An arbitrary name to associate with this email server configuration
Description	(Optional) Email server description
From address	The email address that outgoing mails will appear to have come from (unless overridden per project in JIRA Enterprise). Note that this is just the address part ("jira@company.com"). JIRA will use it in constructing the full From header based on the current user ("Joe Bloggs (JIRA) <jira@company.com>"). To change the From header, go to the "Administration" menu, select "General Configuration" (under "Global Settings") and edit the **Email From Header** field.
Email prefix	The subject of emails sent from this server will use this string as a prefix. This is useful for your users so that they can filter their email.



Adding an SMTP mail server in JIRA

1.4.5.2. 2. Specify the Host Name or JNDI Location

The second part of the screen specifies the **Server Details** of the SMTP server to which JIRA will send mail. There are two ways you can do this. Either:

- specify the **Host Name** of your mail server;
  or:
- specify the **JNDI Location** — that is, use JNDI to look up a mail server that you have preconfigured in your application server. This has the following advantages:
  - Better security: the mail details are not available to JIRA administrators through the JIRA administration interface, and are not stored in JIRA backup files.
  - More SMTP options: if you want to use SMTP over **SSL** (see below), you will need to use JNDI.
  - Centralised management: mail details are configured in the same place as database details,

and may be configured through your application server administration tools.

To specify the Host Name,

Most people configure SMTP details directly in JIRA. The form fields are as follows:

Host Name	Hostname or IP address of your SMTP server. Eg. `mail.yourcompany.com`
SMTP Port	The SMTP port, usually 25
Username	Username to connect as, if your SMTP host requires authentication. (Most company servers require authentication to relay mail to non-local users.)
Password	Password for username (if required by your SMTP host).

Note:

If your server's startup script uses the "**-Dmail**" system properties (e.g. "**mail.smtp.host**" or "**mail.smtp.port**"), they will override the settings that you specify in the above form. Additionally, if necessary you can manually specify the host name that JIRA reports itself as to the SMTP server by setting **-Dmail.smtp.localhost**

Once done, click 'Update' and then "Send a Test Email" to test the connection details.

To specify and configure a JNDI Location,

As an alternative to specifying mail details directly in JIRA, you can configure them in your application server, and then look up a preconfigured mail session via JNDI.

Complete the following form field

JNDI Location	The JNDI location of a javax.mail.MailSession object to use when sending email.

The **JNDI Location** will depend on your application server and configuration. For example, in Tomcat 5.5 (the default application server that is bundled with JIRA Standalone), your **JNDI Location** would be **java:comp/env/mail/JiraMailServer**, and you would add the following section in `conf/server.xml`, inside the <Context> node:

```
<Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
  ....

  <Resource name="mail/JiraMailServer"
    auth="Container"
    type="javax.mail.Session"
    mail.smtp.host="mail.yourcompany.com"
    mail.smtp.port="25"
    mail.transport.protocol="smtp"
    mail.smtp.auth="true"
    mail.smtp.user="jirauser"
    password="mypassword"
    />
    </Context>
```

Or if you don't require authentication (e.g. if you are sending via localhost, or only internally within the company):

```
<Context path="" docBase="${catalina.home}/atlassian-jira" reloadable="false">
  ....
```

```
      <Resource name="mail/JiraMailServer"
        auth="Container"
        type="javax.mail.Session"
        mail.smtp.host="localhost"
        mail.smtp.port="25"
        mail.transport.protocol="smtp"
        />
        </Context>
```

The format for other application servers will be similar. For details please see the Transaction Factory documentation.

If you have problems connecting, add a **mail.debug="true"** parameter, which will let you see SMTP-level details when testing the connection.

You will also need to ensure that the JavaMail classes are present in your application server's classpath, and do not conflict with JIRA's copy. Most J2EE application servers (eg. **JBoss, Orion, Weblogic, Websphere**) come with JavaMail, and this may conflict with JIRA's copy, resulting in errors like:

```
java.lang.NoClassDefFoundError: javax/mail/Authenticator
```

or:

```
java.lang.IllegalArgumentException: Mail server at location [java:comp/env/mail/JiraMailSe
          of required type javax.mail.Session.
```

To fix this, **remove** `WEB-INF/lib/javamail-1.3.2.jar` and `WEB-INF/lib/activation-1.0.2.jar` from the JIRA webapp.

Lighter app servers (**Tomcat, Resin, Jetty (but not JettyPlus)**) do not come with JavaMail. For these, you should **move** `WEB-INF/lib/javamail-1.3.2.jar` and `WEB-INF/lib/activation-1.0.2.jar` into the application server's lib/ directory, eg. `common/lib/` for Tomcat. This is necessary because the application server is establishing the SMTP connection, not JIRA, and the application server won't see the jars in JIRA's classloader.

SMTP over SSL

You can encrypt email communications between JIRA and your mail server via SSL, provided your mail server supports SSL.

To do this, edit your mail server connection properties and specify `starttls` and `SSLSocketFactory`, e.g.:

```
<Resource name="mail/GmailSmtpServer"
  auth="Container"
  type="javax.mail.Session"
  mail.smtp.host="smtp.gmail.com"
  mail.smtp.port="465"
  mail.smtp.auth="true"
  mail.smtp.user="myusername@gmail.com"
  password="mypassword"
  mail.smtp.starttls.enable="true"
  mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"
/>
```

Please note that there is a known bug in some versions of Tomcat 5.5.x (please see JRA-12180).

Additionally, as you are connecting to an SSL service, you will need to **import the SMTP server certificate** into a Java keystore. The process is described on the Connecting to SSL Services page.

For example, on Linux, you could import a certificate as follows:

```
$JAVA_HOME/jre/bin/keytool -import -alias jiramailserver -keystore ~/.keystore  -file /etc
```

```
Enter keystore password:  changeit
Owner: O=Atlassian, L=Sydney, ST=NSW, C=AU
Issuer: O=Atlassian, L=Sydney, ST=NSW, C=AU
Serial number: 0
Valid from: Wed Dec 29 13:02:52 EST 2004 until: Sat May 15 12:02:52 EST 2032
Certificate fingerprints:
         MD5:  91:EC:6E:EA:73:7A:7C:4F:88:92:A2:A0:2B:F7:BC:CC
         SHA1: D8:7C:09:8A:8D:D8:7D:59:C2:28:2A:09:85:90:82:46:78:06:38:D5
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

You would also need to tell Tomcat where the keystore file is located by adding the following to `bin/setenv.sh`:

```
export JAVA_OPTS="-Djavax.net.ssl.trustStore=$HOME/.keystore"
```

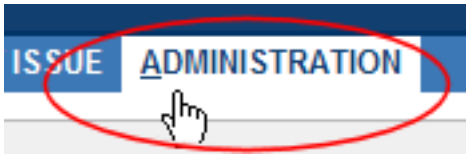
1.5. User Management

1.5.1. Managing Users

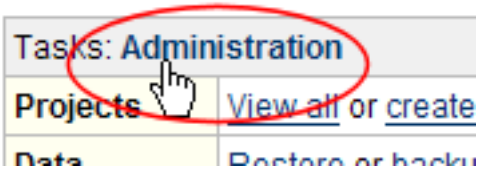
1.5.1.1. Viewing Users

To view a list of JIRA users:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

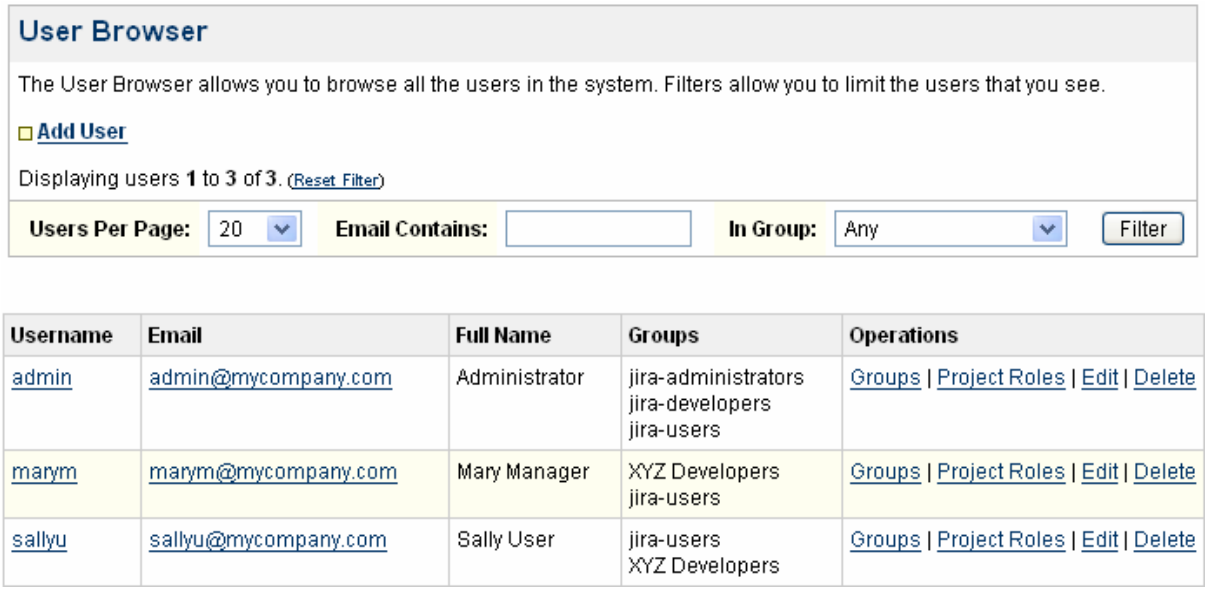


Link to Administration section



Link to Administration section

3. Select **User Browser** from the **Users, Groups & Roles** section of the administration menu. This will display the **User Browser** screen:



User Browser

4. To restrict the list of users shown in the **User Browser**, use the Filter form at the top of the User Browser. Specifying (part of) the user's email and/or group, then clicking the **Filter** button, will reduce the list to only those users who match those criteria.

1.5.1.2. Adding a User

1. Open the **User Browser** (see 'Viewing Users' above) and click the **Add User** link.
2. This will display the **Create New User** form. Enter the **Username** (note that a user's **Username** cannot be changed once the user is created), **Password**, **Full Name** and **Email Address**; and (optionally) tick the box to send the user an email containing their account details. Then click the **Create** button.

Note:

Users can also be created via:
- **Signup** — see "Enabling Public Signup"
- **Email** — e.g. you can use the CreateIssueHandler (see "Services") to have JIRA create a user based on the sender's email address.

Note:

If you have a user limited license (e.g. personal license) and have reached your user limit, any further users added will not have permission to log in to JIRA.

1.5.1.3. Assigning a User to a Group

When a user is created, they will be added to any groups that are set up to have new users automatically added to them.

To change a user's group membership:

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click the **Groups** link in the **Operations** column.
2. This will display two lists; the one on the left shows all available groups, and the one on the right shows all groups to which the user currently belongs. Use the **Join** and **Leave** buttons to add the user to or remove them from your selected group.

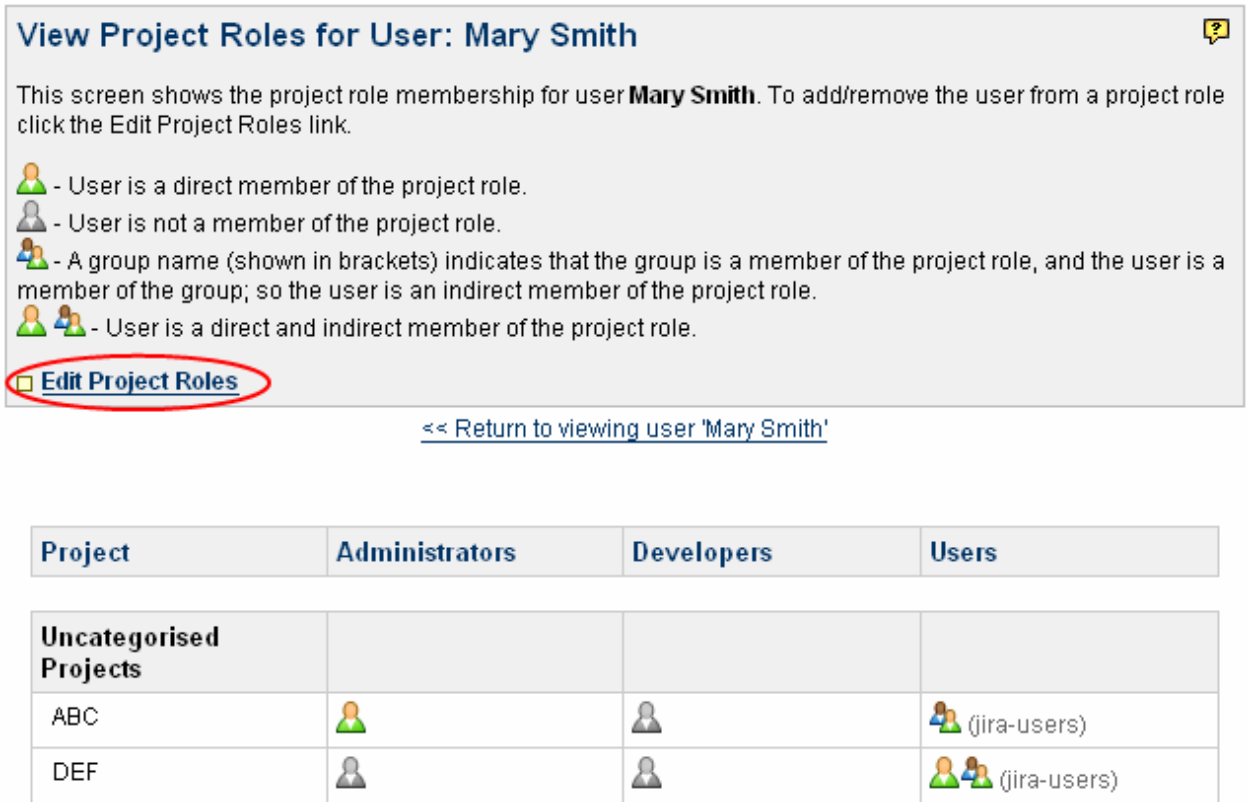
Note:

If you have a user limited license (e.g. personal license) and have reached your user limit, you will not be able to assign any further users to groups with login permissions (i.e. jira-users permission) without first reducing the number of users with login permissions.

1.5.1.4. Assigning a User to a Project Role

Assigning a user to a project role enables them to fulfil a particular function in a particular project.

To view a user's project role membership, locate the user in the **User Browser** (see 'Viewing Users' above) and click the **Project Roles** link in the **Operations** column. This will display a table showing all the projects and project roles that exist in JIRA, and the user's current project role membership for each project:



User's Project Role Membership

E.g. this screenshot shows that, for the ABC project:

- Mary is a member of the 'Administrators' project role.
- Mary is not a member of the 'Developers' project role.
- Mary is indirectly a member of the 'Users' project role, through being a member of the 'jira-users' group.

(Also note that, for the DEF project, May is both a direct *and* an indirect member of the 'Users' project role.)

Click the **Edit Project Roles** button. The check-boxes will then be available for you to tick (to add the user to a project role) or un-tick (to remove the user from a project role).

1.5.1.5. Changing a User's Name or Email Address

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click their **Edit** link in the **Operations** column.
2. This displays a form where you can change the user's Full Name or Email Address. Click **Update** to confirm the change.

1.5.1.6. Changing a User's Password

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click their **Username**.
2. This displays the user's details, below which are several links. Click the **Set Password** link.
3. This displays the **Set Password**screen. Enter and confirm the new password; then click the **Update** button.

1.5.1.7. Adding a Property to a User

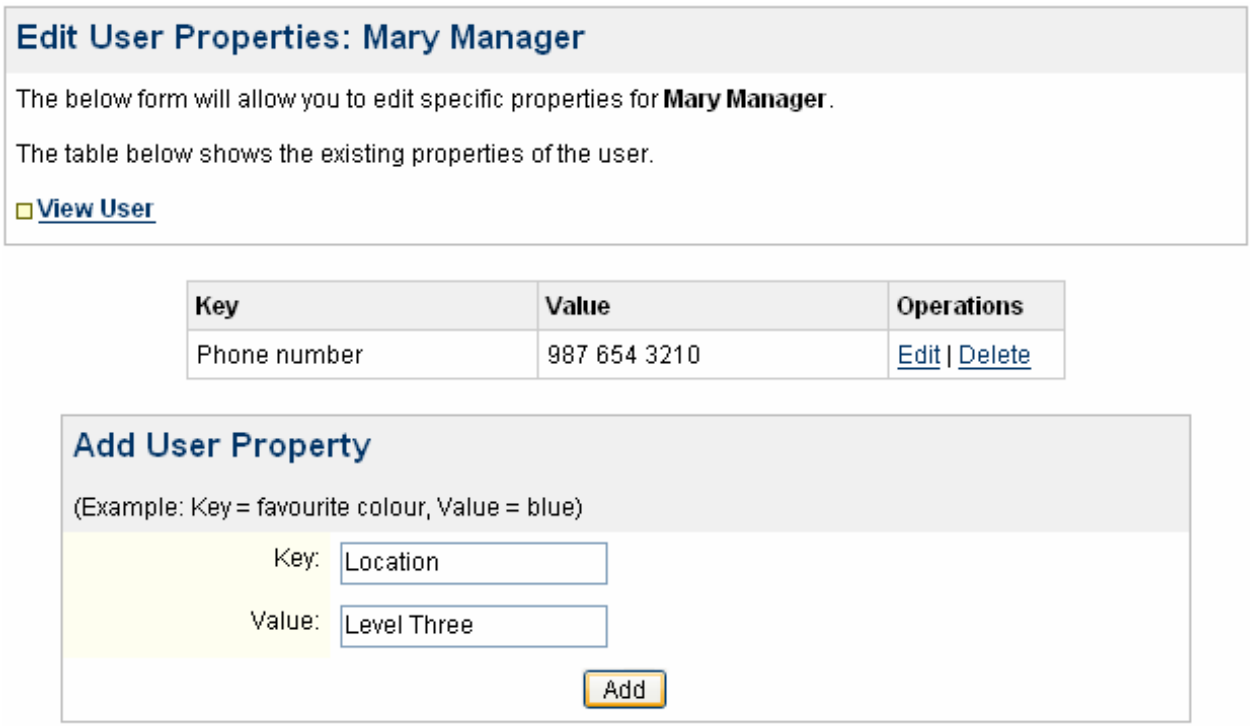
A '**Property**' is an extra piece of information that you can store regarding a user. A Property consists of a **Key** of your choice (eg. 'Phone number', 'Location') plus a corresponding **Value** (eg. '987 654 3210', 'Level Three').

To create a new Property for a user:

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click their **Username**.
2. This displays the user's details, below which are several links. Click the **Edit Properties** link.

User Properties

3. This displays the **Edit User Properties** screen, showing any previously-created properties:



User Properties

4. Enter the new **Key** and its **Value**, then click the **Add** button.

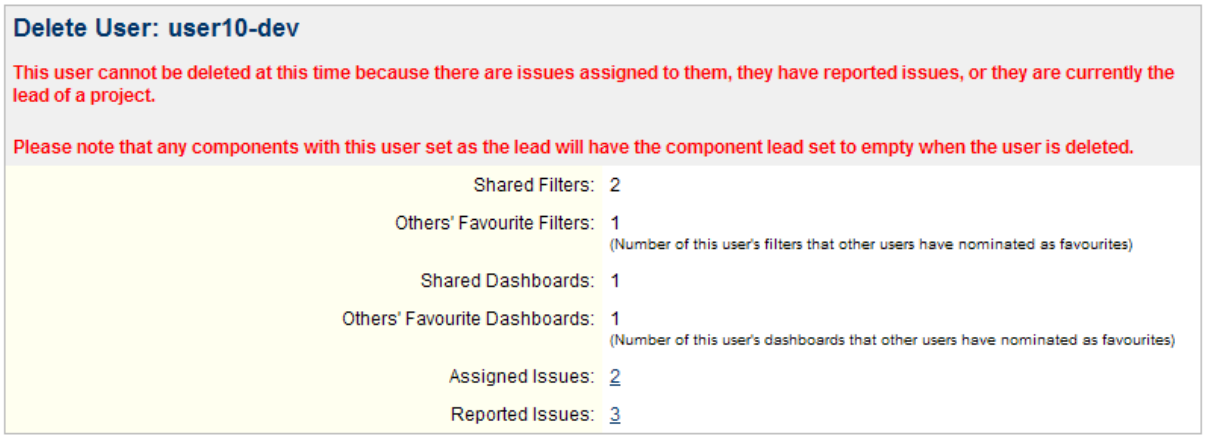
1.5.1.8. Deleting a User

Note:

Rather than deleting a user, it is recommended to *disable their account* by removing them from all groups (see 'Assigning a User to a Group', above). This prevents the user's account from being used. It is still important to reassign any issues assigned to that user, but there is no need to modify the 'Reporter' as described below.

To delete a user,

1. Locate the user in the **User Browser** (see 'Viewing Users' above) and click the **Delete** link in the **Operations** column.
2. The confirmation screen that follows will summarise any involvement of that user in the system by showing current issues assigned to and reported by that user, etc. These connections between the user and other parts of the system may prevent the deletion of that user. For example, attempting to delete a user called *bob* results in the following screen, which prevents deletion due to the presence of 10 assigned issues:



Deletion Prevented

   As well as reassigning any issues, you may need to bulk-edit the issues created by the user and change the 'Reporter' to someone else. You'll need the 'Modify Reporter' permission to do this.
3. If there are no issues assigned to, or reported by the user, the confirmation screen will display a **Delete** button; click this to proceed with the deletion.

Note:

Please note that the filters and dashboards of a user will be deleted when the user is deleted, regardless of whether the filters or dashboards are shared with other users.

Note:

If you are using External User Management, you will not be able to create, edit or delete users from within JIRA; but you can still assign users to project roles, and create/edit/delete user properties.

1.5.2. Managing groups

A JIRA group is a convenient way to manage a collection of users. Users can belong to many groups. Groups are used throughout JIRA; for example, they can:

- be granted global permissions.
- be used in project permission schemes.
- be used in email notification schemes.
- be used in issue security levels.

- be given access to issue filters *(in JIRA Professional and Enterprise editions)*.
- be given access to dashboards *(in JIRA Professional and Enterprise editions)*.
- be used in workflow conditions *(in JIRA Professional and Enterprise editions)*.
- belong to project roles*.

*Project roles are somewhat similar to groups, the main difference being that group membership is global whereas project role membership is project-specific.

1.5.2.1. JIRA's default groups

When you install JIRA, three groups are automatically created:

- **jira-administrators** — typically contains people who are JIRA system administrators. By default, this group:
  - is a member of the '**Administrators**' project role.
  - has the '**JIRA Administrators**' and the '**JIRA System Administrators**' global permissions. (Note: if you need to give these permissions to separate people, you will need to create an additional group and grant the permissions separately, as described in *'About 'JIRA System Administrators' and 'JIRA Administrators''*.)
- **jira-developers** — typically contains people who perform work on issues. By default, this group:
  - is a member of the '**Developers**' project role.
  - has the '**Browse Users**', '**Create Shared Filter**' and '**Manage Group Filter Subscriptions**' global permissions.
- **jira-users** — typically contains every JIRA user in your system. By default, this group:
  - is a member of the '**Users**' project role.
  - has the '**JIRA Users**' and '**Bulk Change**' global permissions.

You can create and delete groups according to your organisation's requirements.

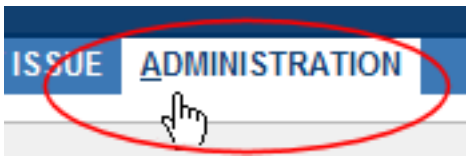
Note:

If you are using External User Management, you will not be able to create, delete or edit groups or group membership from within JIRA; and 'Automatic Group Membership' (see below) will not apply. However, you can still assign groups to project roles.

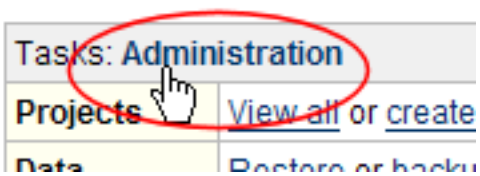
1.5.2.2. Viewing groups

To see what groups exist, and where they are used:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

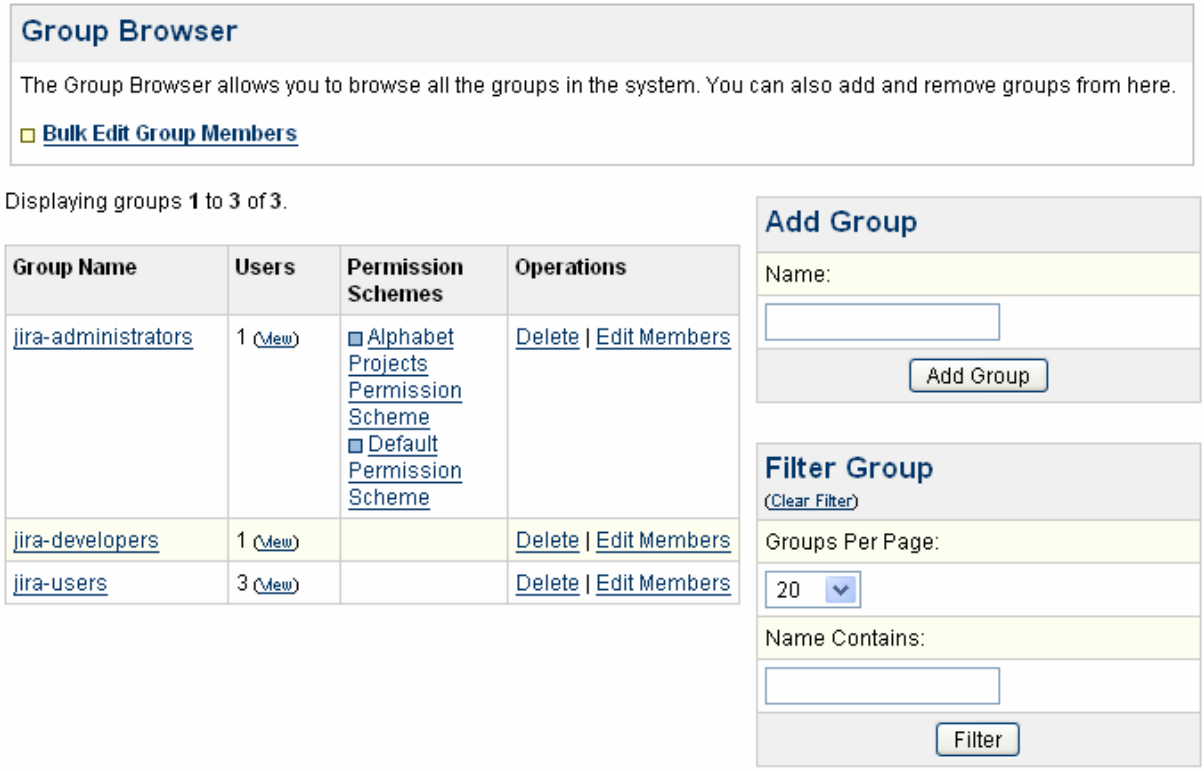


Link to Administration section



Link to Administration section

3. Select '**Group Browser**' from the '**Users, Groups & Roles**' section of the 'Administration' menu.
4. You will then see a page containing the '**Group Browser**' as shown below.



Group Browser

Note:

The '**Filter Group**' form restricts the list of groups shown to those that match the '**Name Contains**', with a specified maximum per page. Click the '**Filter**' button to refresh the list with the restricting filter.

5. To see which permission schemes, email notification schemes, issue security levels and saved filters are using this group, click the group name.

1.5.2.3. Adding a group

To create a group, enter the new group '**Name**' in the '**Add Group**' form in the '**Group Browser**' (see *'Viewing groups'* above) and click the '**Add Group**' button.

1.5.2.4. Deleting a group

To delete a group, click the '**Delete**' link for that group in the '**Group Browser**' (see *'Viewing groups'* above). The confirmation screen that follows explains that users will be removed from the group through its deletion. Be aware of the impact this may have on users in that group. For example, if that group membership was the sole conveyor of a permission for a user, then the user will no longer have that permission.

Note:

Before deleting a group it is recommended that you check whether the groups is being used by any permission schemes, email notification schemes, issue security levels or saved filters. See *'Viewing groups'* (above).

1.5.2.5. Editing group membership

To edit a group's membership, click the '**Edit Members**' link in the row for that group in the '**Group Browser**' (see *'Viewing groups'* above). This takes you to a form allowing you to add users to or remove them from the group.

Note:

If the group has the '**JIRA System Administrators**' global permission, you cannot edit its membership unless you have the '**JIRA System Administrators**' global permission.

Note:

If you have a user limited license (e.g. personal license) and have reached your user limit, you will not be able to assign any further users to groups with login permissions (i.e. jira-users permission) without first reducing the number of users with login permissions.

1.5.2.6. Automatic group membership

To automatically add newly-created users to a particular group, grant the group the '**JIRA Users**' global permission.

To do this, navigate to the '**Administration**' section and select '**Global Permissions**' from the '**Global Settings**' menu. Add the '**JIRA Users**' permission to the relevant group, as described in *'Granting global permissions'*.

JIRA Permissions	
JIRA System Administrators Ability to perform all administration functions. There must be at least one group with this permission. Note: People with this permission can always log in to JIRA.	jira-administrators (View Users | Delete)
JIRA Administrators Ability to perform most administration functions (excluding Import & Export, SMTP Configuration, etc.). Note: People with this permission can always log in to JIRA.	jira-administrators (View Users | Delete)
JIRA Users Ability to login to JIRA. They are a 'user'. Any new users created will automatically join these groups. Note: All users need this permission to login to JIRA, even if they have other permissions.	jira-users (View Users | Delete)
Browse Users Ability to select a user or group from a popup window. Users with this permission will be able to see names of all users and groups in the system.	jira-developers (View Users | Delete)
Create Shared Filter Ability to share a filter globally or with group of users.	jira-developers (View Users | Delete)
Manage Group Filter Subscriptions Ability to manage (create and delete) group filter subscriptions.	jira-developers (View Users | Delete)
Bulk Change Ability to modify a collection of issues at once. For example, resolve multiple issues in one step.	jira-users (View Users | Delete)

Global permissions screen

1.5.3. Managing project roles

Project roles are a flexible way to associate users and/or groups with particular projects. In JIRA Enterprise, project roles also allow for delegated administration:

- Global administrators define JIRA's project roles - that is, all projects have the same project roles available to them.
- Project administrators* assign members to project roles specifically for their project(s).

Project roles can be used in:

- permission schemes.
- email notification schemes.
- issue security levels.

- comment visibility.
- workflow conditions.

Project roles can also be given access to:

- issue filters *(in JIRA Professional and Enterprise editions)*.
- dashboards *(in JIRA Professional and Enterprise editions)*.

Project roles are somewhat similar to groups, the main difference being that group membership is global whereas project role membership is project-specific. Additionally, group membership can only be altered by JIRA administrators, whereas project role membership can be altered by project administrators*.

Note:

*A project administrator is someone who has the project-specific 'Administer Project' permission, but not necessarily the global 'JIRA Administrator' permission. In JIRA Enterprise, a project administrator can manage project role membership. In JIRA Professional and Standard editions, only global administrators can manage project role membership.

1.5.3.1. Using project roles

Project roles enable you to associate users with particular functions. For example, if your organisation requires all software development issues to be tested by a Quality Assurance person before being closed, you could do the following:

1. Create a project role called **Quality Assurance**
2. Create a permission scheme called **Software Development**, in which you assign the 'Close Issue' permission to the **Quality Assurance** project role.
3. Associate the **Software Development** permission scheme with all software development projects.
4. For each software development project, add the appropriate Quality Assurance people to the **Quality Assurance** project role.

Note:

JIRA versions prior to 3.7 did not have project roles. If you previously used JIRA 3.6.x (or earlier), please see Using 'Scheme Tools' to migrate to Project Roles.

1.5.3.2. JIRA's default project roles

When you install JIRA, three project roles are automatically created:

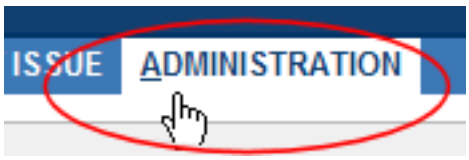
- **Administrators** - typically contains people who administer a given project.
- **Developers** - typically contains people who work on issues in a given project.
- **Users** - typically contains people who log issues in a given project.

You can create, edit and delete project roles according to your organisation's requirements.

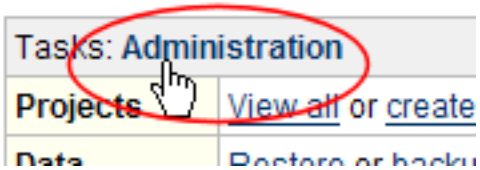
1.5.3.3. Viewing project roles

To see what project roles exist, and where they are used:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

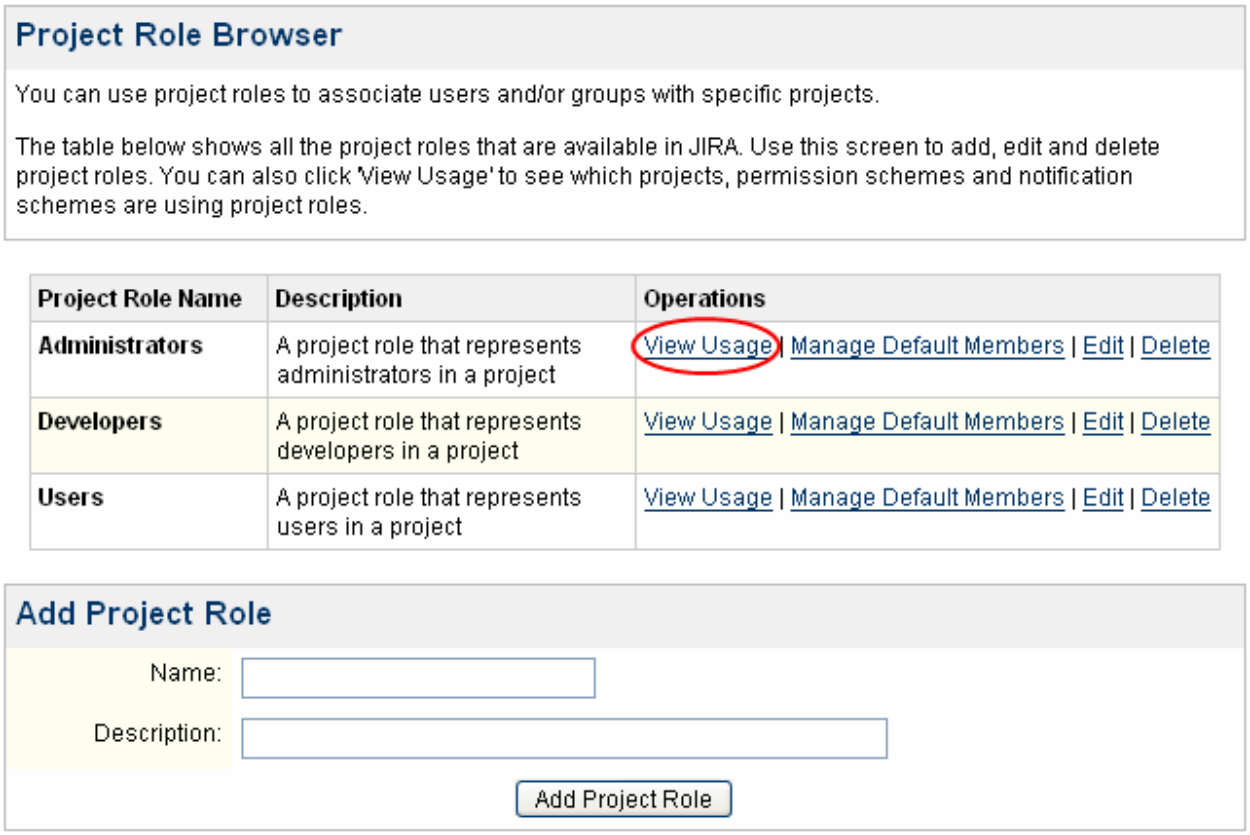


Link to Administration section



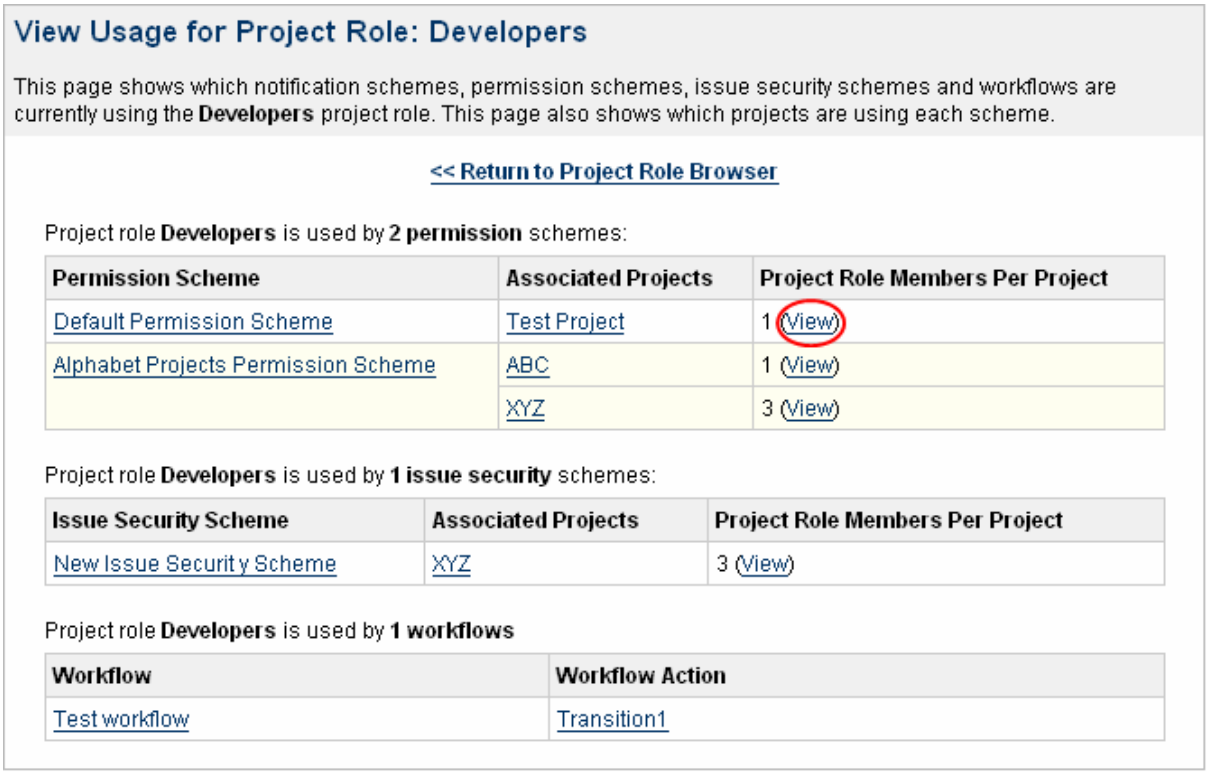
Link to Administration section

3. Select **Project Role Browser** from the **Users Groups & Roles** section of the **Administration** menu. You will then see the Project Role Browser, which contains a list of all the project roles in your JIRA system. To see where a project role is used, click the **View Usage** link:



Project Role Browser

4. This will display a list of the project role's associated permission schemes, email notification schemes, issue security levels and workflow conditions. To see which users/groups are associated with a project role for a particular project, click the **View** link:



Project Role Usage

1.5.3.4. Adding a project role

To define a new project role, enter its Name and a Description in the **Add Project Role** form in the Project Role Browser (see 'Viewing Project Roles' above), and click the **Add Project Role** button. Note that project role names must be unique.

Once a new project role is created, it is available to all projects. Project administrators can then assign members to the project role for their project (see Managing project role membership).

1.5.3.5. Deleting a project role

To delete a project role, locate the project role in the Project Role Browser (see 'Viewing Project Roles' above), and click the **Delete** link. The confirmation screen that follows lists any permission schemes, email notification schemes, issue security levels and workflow conditions that use the project role.

Note that deleting a project role will remove any assigned users and groups from that project role, for all projects. Be aware of the impact this may have; for example, if the project role membership was the sole conveyor of a permission for a user, then the user will no longer have that permisison.

Note:

If a project role has been used to specify who can view a comment, deleting the project role will mean that noone can see that comment any more.

1.5.3.6. Editing a project role

To edit the **Name** and **Description** of a project role, locate the project role in the Project Role Browser (see 'Viewing Project Roles' above), and click the **Edit** link. This takes you to a form where you can modify the project role's **Name** and **Description**.

1.5.3.7. Assigning members to a project role

A project role's members are assigned on a project-specific basis. To assign users/groups to a project role for a particular project, please see Managing project role membership.

To see/edit *all* the project roles to which a particular user belongs, for all projects, click the **Project Roles** link in the User Browser.

1.5.3.8. Specifying 'default members' for a project role

The default members for a project role are users and groups that are initially assigned to the project role for all newly created projects. The actual membership for any particular project can then be modified by the project administrator.

The default members consist of the **Default Users** plus the **Default Groups** shown in the Project Role Browser (see 'Viewing Project Roles' above).

To add to the **Default Users** or the **Default Groups** for a project role, click the corresponding **Edit** link.

For example, if a user called Susie needs to have administration permissions for all newly created projects, you could add her to the **Default Users** for the 'Administrator' project role as follows:

1. Open the Project Role Browser.
2. Click the **Edit** link in the **Administrators** column (next to '*None selected*').
3. In the 'Assign Default Users to Project Role' screen, click the 'User Picker' icon.
4. Locate Susie in the 'User Picker' popup window, then click the **Select** button.
5. In the 'Assign Default Users to Project Role' screen, click the **Add** button.

Note:

Changing a project role's default members does not affect the actual project role members for projects already created.

1.5.4. Using 'Scheme Tools' to migrate to Project Roles

Project roles are a flexible way of associating particular users and groups with a particular project.

1.5.4.1. Why migrate to Project Roles?

- **Ease of management**

  JIRA versions prior to 3.7 did not have project roles. If you previously used JIRA 3.6.x (or earlier), your system may contain multiple, project-specific groups, permission schemes and notification schemes. By implementing project roles, you may be able to reduce the number of groups, permission schemes and notification schemes in your JIRA system. This can make your system easier to manage.

- **Delegated administration**

  In JIRA Enterprise, a project administrator (that is, someone who has the 'Administer Project' permission, but not neceessarily the global 'JIRA Administrator' permission) can assign users and groups to project roles for their project. If their project's permission scheme and notification scheme are using project roles, the project administrator can control who may access their project and who receives email notifications.

The instructions on this page will help you use Scheme Tools to:

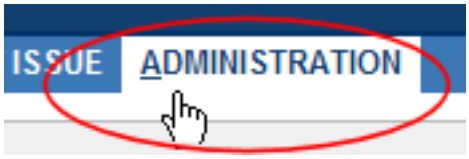
- update your permission schemes and notification schemes so that they use project roles instead

of groups; then
- minimise the number of permission schemes and notification schemes in your JIRA system.

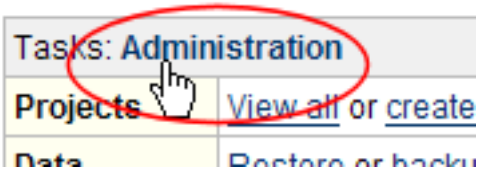
1.5.4.2. Updating Permission Schemes and Notification Schemes to use Project Roles instead of Groups

Note:
Before you begin, please perform a full backup.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

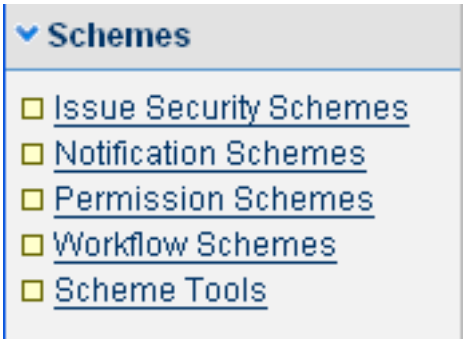


Link to Administration section



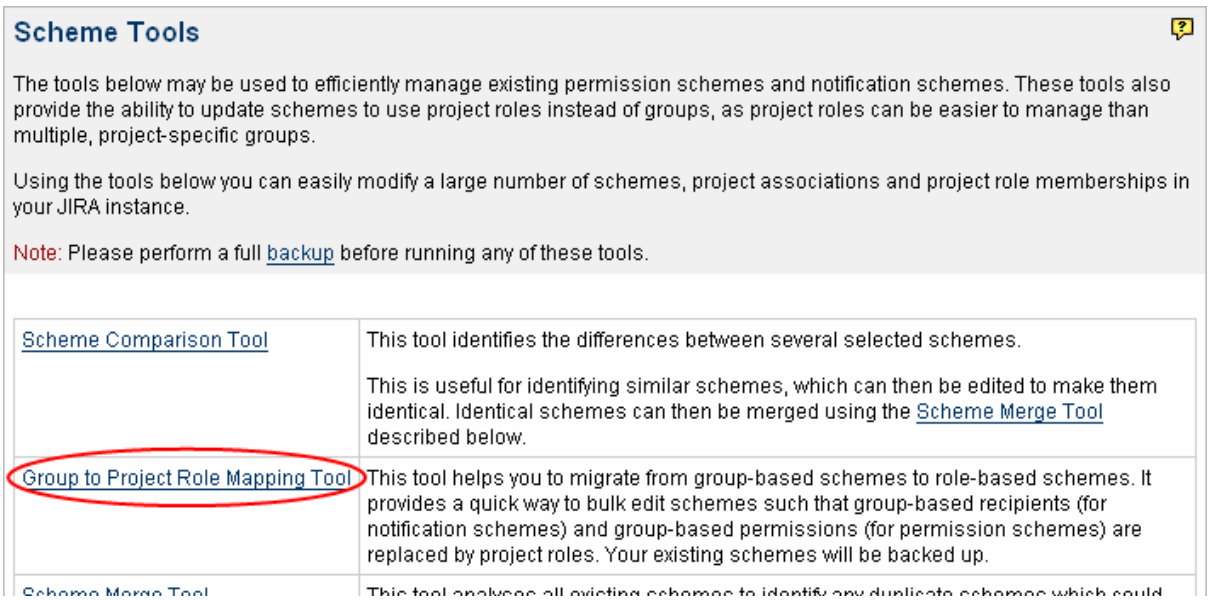
Link to Administration section

3. On the panel on the left, under the title "Schemes", click the link labelled "Scheme Tools".



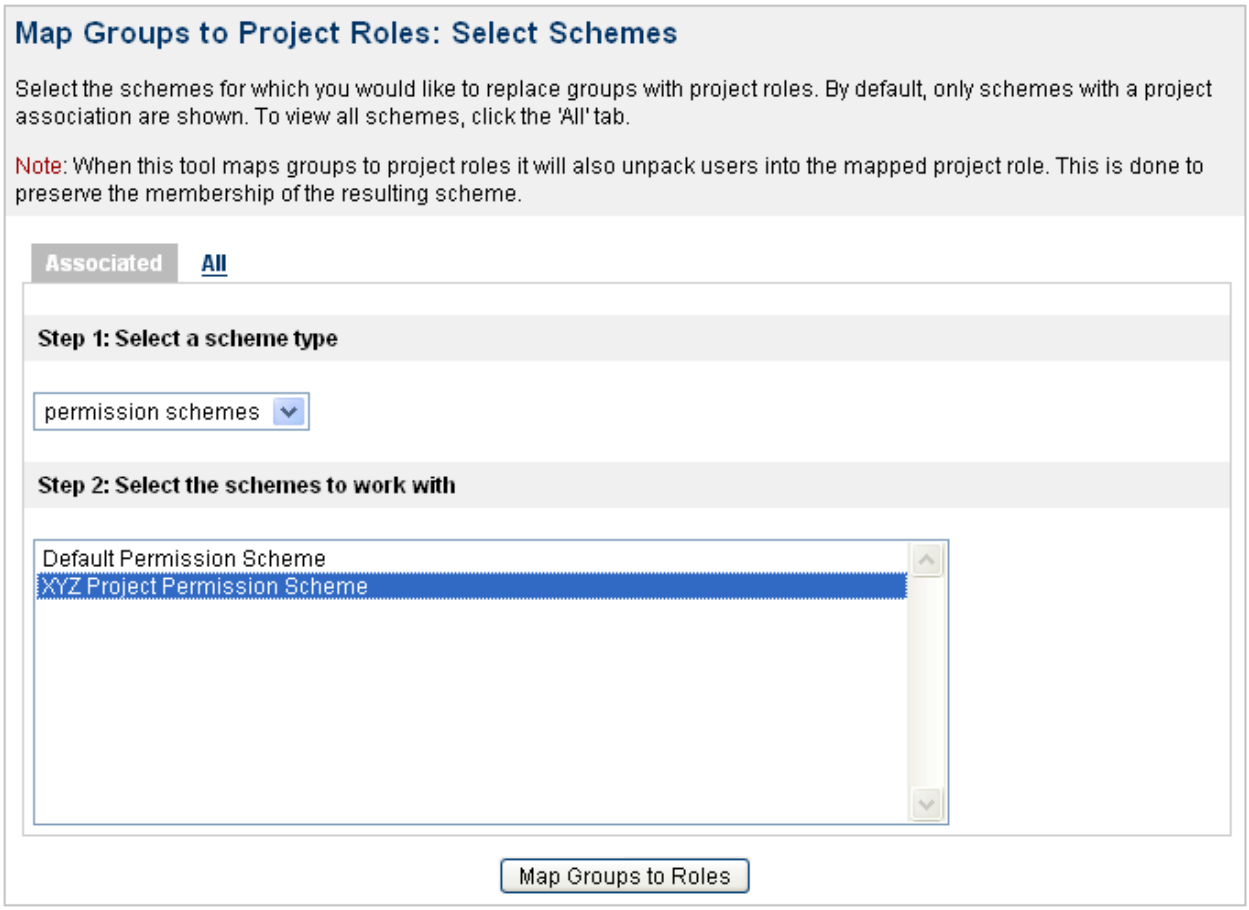
Scheme Tools

4. This will display the "Scheme Tools" page. Click the **"Group to Project Role Mapping Tool"** link.



Scheme Tools

5. This will display the "Map Groups to Project Roles: Select Schemes" page:

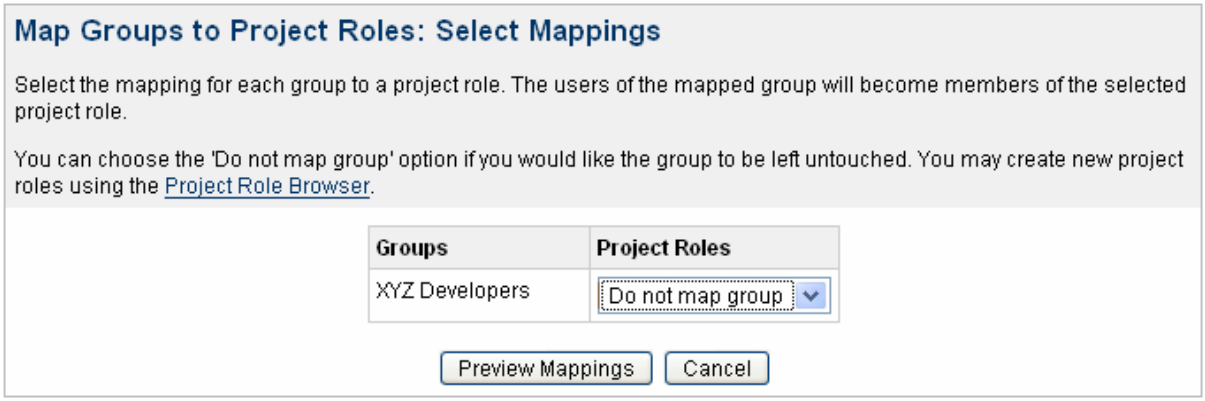


Select schemes to update

- Note that schemes that are not associated with any projects need not usually be included in this process; but if you wish to select from all schemes in your system (including unused schemes), click 'All'.
- Under **Step 1: Select a scheme type**, select whether you want to update permission schemes or notification schemes. (You can only do one type of scheme at a time, but you can easily come back and do the other type later.)

- Under **Step 2: Select the schemes to work with**, select the schemes you want to update to use project roles instead of groups. You can use the **Ctrl** key to select multiple schemes.
- Then click the "Map Groups to Roles" button.

6. This will display the "Map Groups to Project Roles: Select Mappings" page:

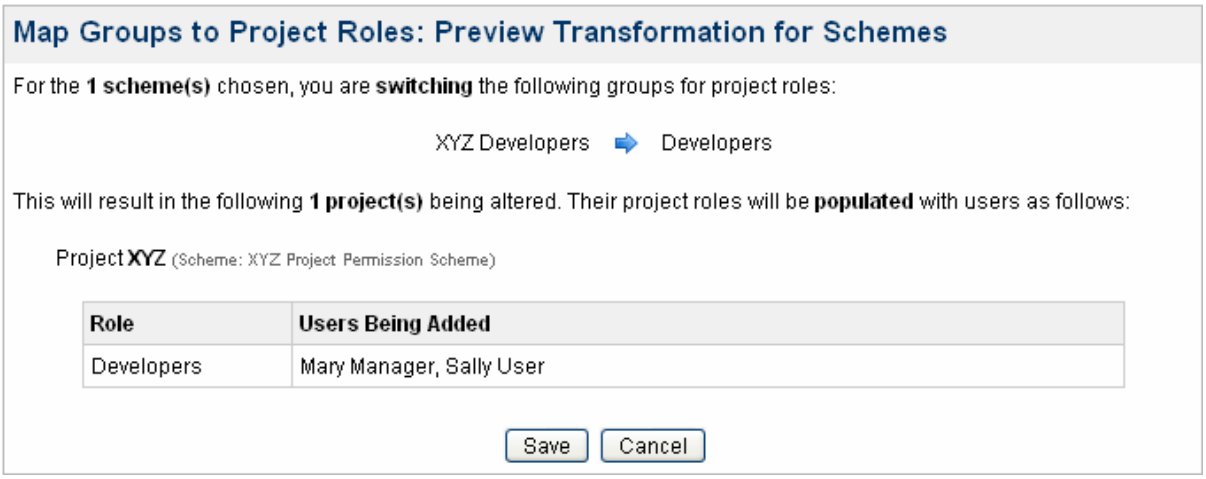


Select groups to update

For each group, select the project role that will replace it; or, for any groups that you do not want to migrate, choose the "Do not map group" option. Then click the "Preview Mappings" button.

- For ease of maintenance, it is recommended that you do not migrate any groups to which JIRA users are automatically added (that is, groups which have the 'JIRA Users' global permission). If you migrate these groups to project roles, and you still want all new users to have access to particular projects, you will need to manually add new users to the relevant project role for each project.

7. You will now see the "Map Groups to Project Roles: Preview Transformation for Schemes" page:



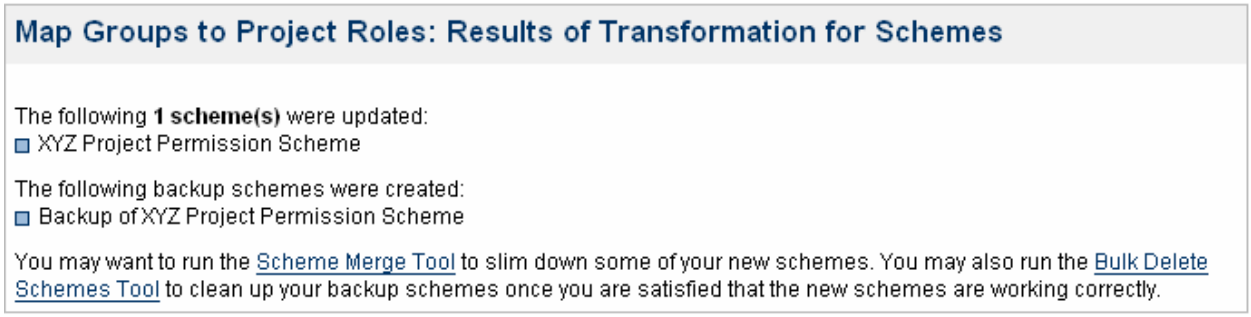
Preview roles to update

If you are satisfied that the information shown on this page is correct, click the "Save" button to:

- create a backup of the scheme(s) that you selected in step 5 (you can later delete this backup scheme by using the "Bulk Delete Schemes Tool", available from the "Scheme Tools" page shown in step 4). This backup scheme will not be associated with any projects.
- update the scheme(s) that you selected in step 5 to use the role (left of the blue arrow) instead of the group (right of the blue arrow)

- add the users (in the right column of the table) to the project role (in the left column of the table) for each project that uses the scheme. This ensures that all users will continue to have the same permissions and notifications.

8. You will now see confirmation of the above changes on the "Map Groups to Project Roles: Results of Transformation for Schemes" page:



Results of groups-to-roles update

After updating your permission schemes and notification schemes to use project roles instead of groups, you may find that many of your schemes are now very similar. To identify such schemes, merge them, and delete any redundant ones, please see Minimising the number of Permission Schemes and Notification Schemes (below).

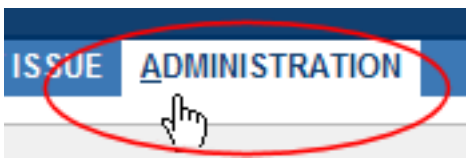
You may also find that some groups are no longer required. You can use the Group Browser to identify and delete groups that are not used by any permission schemes or notification schemes.

1.5.4.3. Minimising the number of Permission Schemes and Notification Schemes

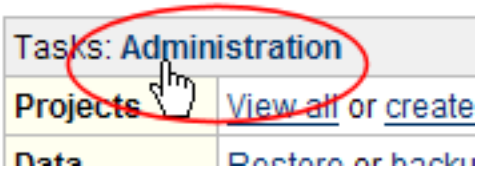
Minimising the number of permissions schemes and notification schemes can make your JIRA system easier to manage. To identify and remove unnecessary schemes, follow the steps below:

Note:
Before you begin, please perform a full backup.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

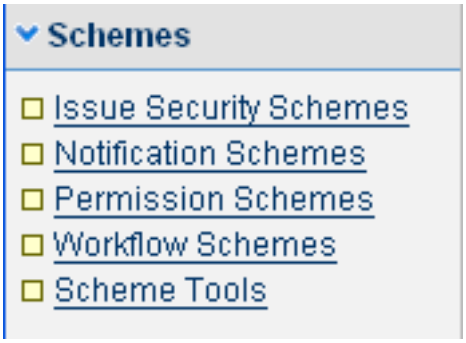


Link to Administration section



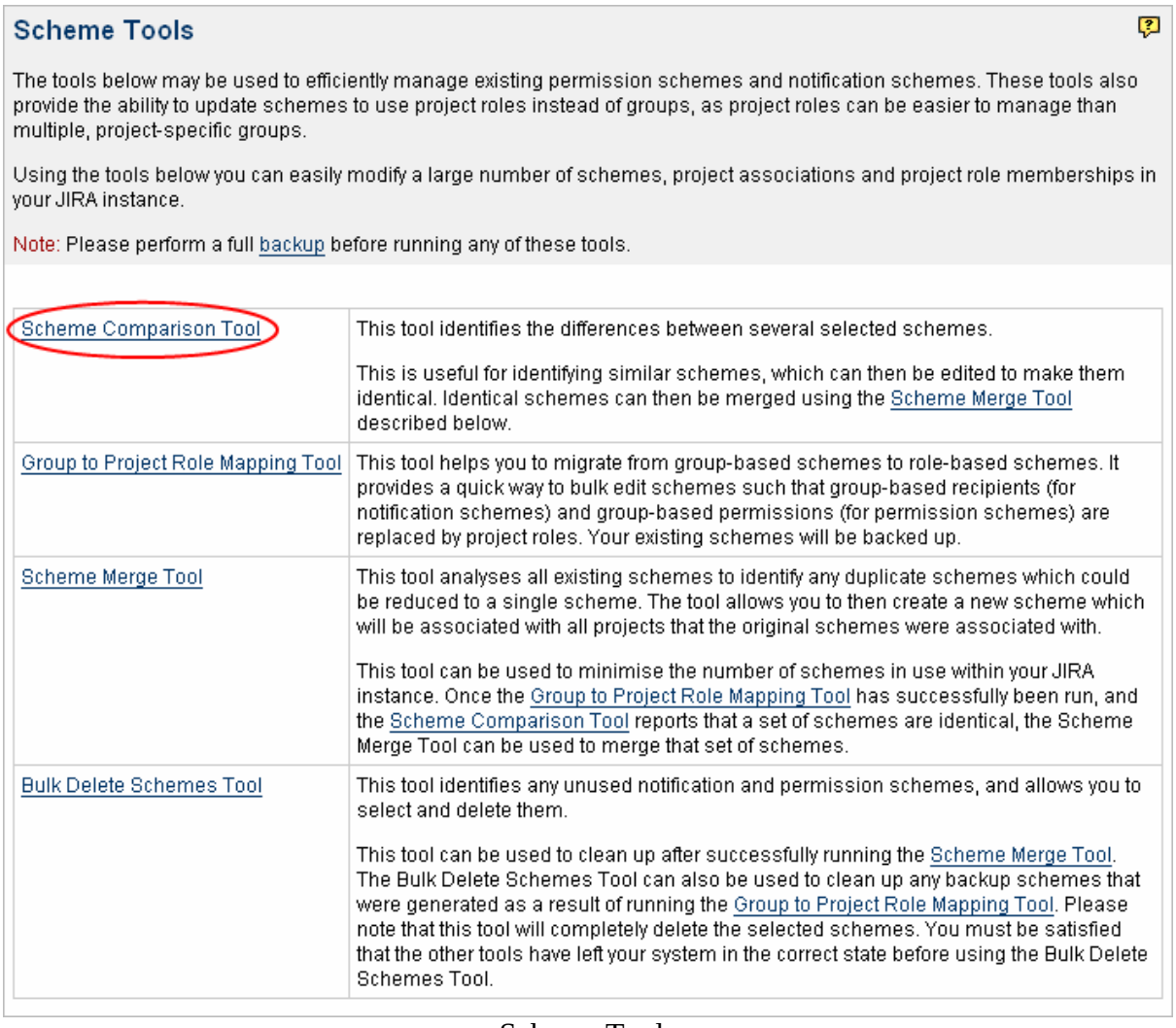
Link to Administration section

3. On the panel on the left, under the title "Schemes", click the link labelled "Scheme Tools".



Scheme Tools

4. This will display the "Scheme Tools" page. Click the "Scheme Comparison Tool" link.



Scheme Tools

The tools below may be used to efficiently manage existing permission schemes and notification schemes. These tools also provide the ability to update schemes to use project roles instead of groups, as project roles can be easier to manage than multiple, project-specific groups.

Using the tools below you can easily modify a large number of schemes, project associations and project role memberships in your JIRA instance.

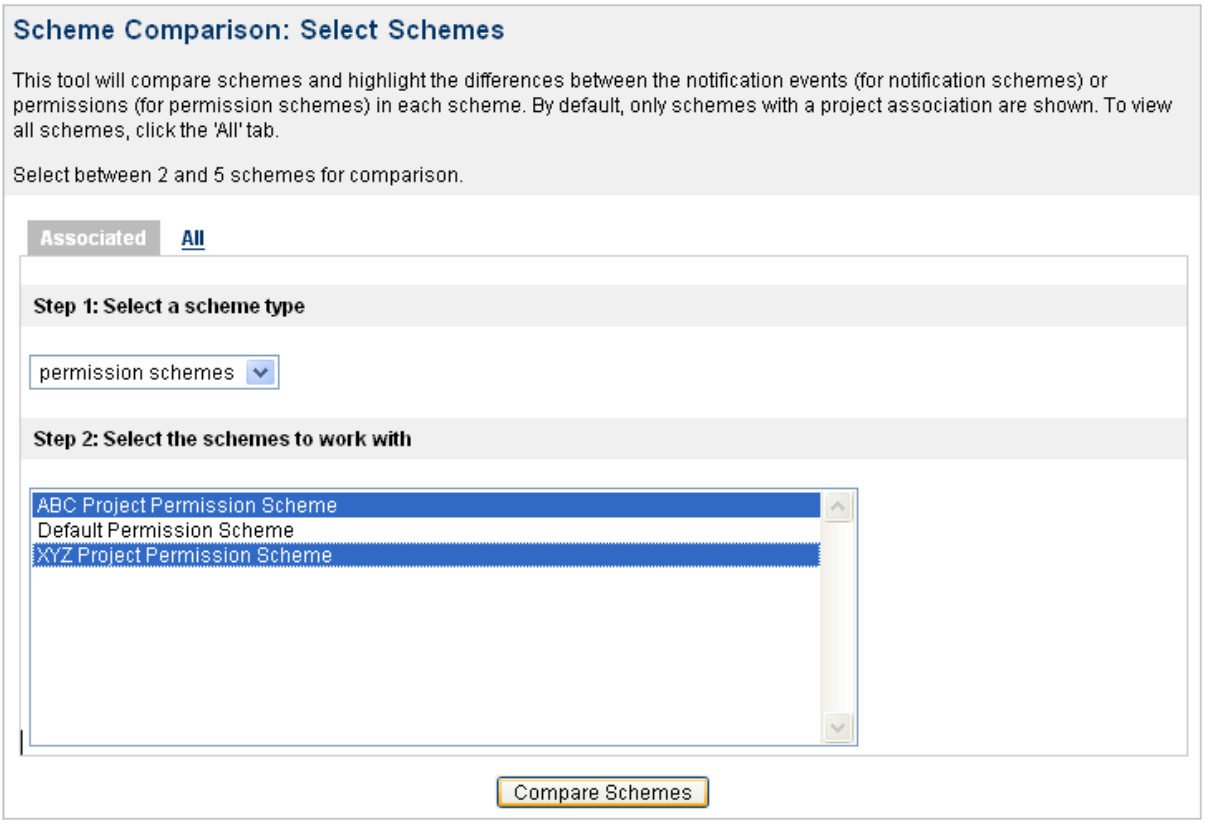
Note: Please perform a full backup before running any of these tools.

Scheme Comparison Tool	This tool identifies the differences between several selected schemes. This is useful for identifying similar schemes, which can then be edited to make them identical. Identical schemes can then be merged using the Scheme Merge Tool described below.
Group to Project Role Mapping Tool	This tool helps you to migrate from group-based schemes to role-based schemes. It provides a quick way to bulk edit schemes such that group-based recipients (for notification schemes) and group-based permissions (for permission schemes) are replaced by project roles. Your existing schemes will be backed up.
Scheme Merge Tool	This tool analyses all existing schemes to identify any duplicate schemes which could be reduced to a single scheme. The tool allows you to then create a new scheme which will be associated with all projects that the original schemes were associated with. This tool can be used to minimise the number of schemes in use within your JIRA instance. Once the Group to Project Role Mapping Tool has successfully been run, and the Scheme Comparison Tool reports that a set of schemes are identical, the Scheme Merge Tool can be used to merge that set of schemes.
Bulk Delete Schemes Tool	This tool identifies any unused notification and permission schemes, and allows you to select and delete them. This tool can be used to clean up after successfully running the Scheme Merge Tool. The Bulk Delete Schemes Tool can also be used to clean up any backup schemes that were generated as a result of running the Group to Project Role Mapping Tool. Please note that this tool will completely delete the selected schemes. You must be satisfied that the other tools have left your system in the correct state before using the Bulk Delete Schemes Tool.

Scheme Tools

- The Scheme Comparison Tool assists you in identifying similar schemes, and if appropriate, making them identical.
- Identical schemes can later be merged using the Merge Duplicate Schemes Tool (see step 9 below).

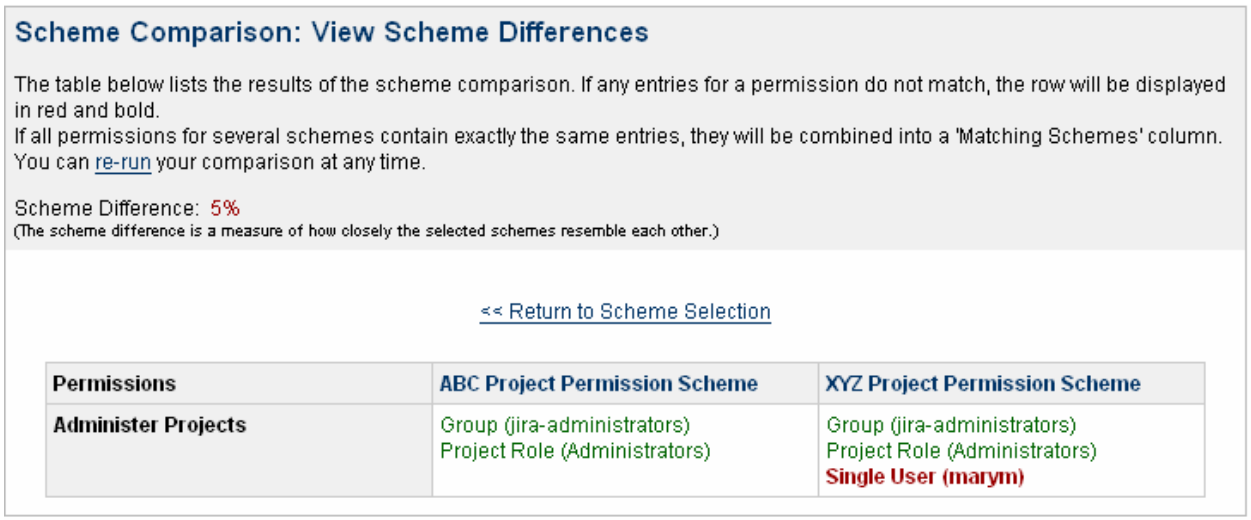
5. This will display the "Scheme Comparison: Select Schemes" page:



Selecting schemes to compare

- Note that schemes which are not associated with any projects need not usually be included in this process; but if you wish to select from all schemes in your system (including unused schemes), click 'All'.
- Under **Step 1: Select a scheme type**, select whether you want to compare permission schemes or notification schemes. (You can only do one type of scheme at a time, but you can easily come back and do the other type later.)
- Under **Step 2: Select the schemes to work with**, select the schemes you want to compare. Select at least 2 (and no more than 5) schemes, using the **Ctrl** key to select multiple schemes.
- Then click the "Compare Schemes" button.

6. This will display the "Scheme Comparison: View Scheme Differences" page:

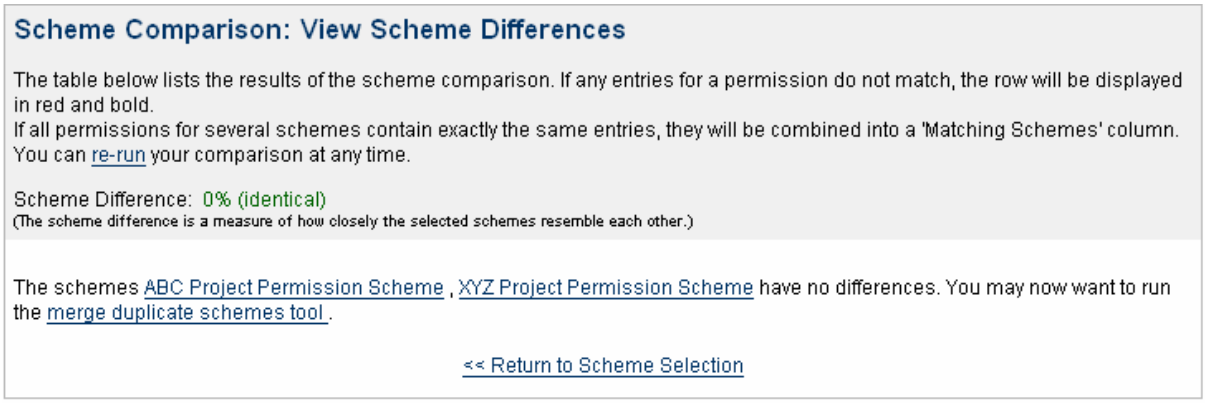


Comparing schemes

- Only the **differences** between the selected schemes are displayed. For example, in the

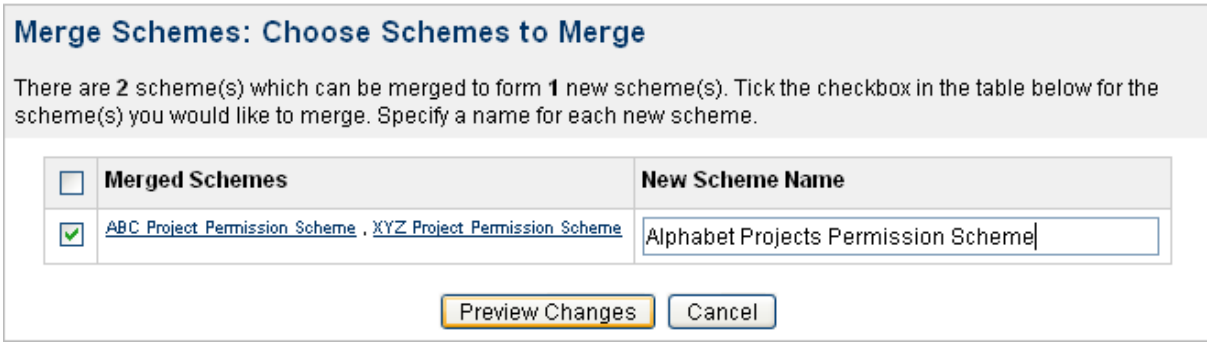
screenshot above, only the "Administer Projects" permission is displayed; this means that all the other permissions in these two permission schemes ("ABC Project Permission Scheme" and "XYZ Project Permission Scheme") are identical.

7. If you decide it is appropriate to edit a scheme to make it the same as another one, you can edit the scheme by clicking the scheme name. For example, it may be appropriate to delete **Single User (marym)** from the "XYZ Project Permission Scheme" if she is a member of the "Administrators" project role for the XYZ project.
8. Then repeat the steps above, and verify that you have achieved a batch of 2 or more identical permission schemes, eg:



Identical schemes

9. Click the "Merge Duplicate Schemes Tool" link. (Note: this link is also available from the "Scheme Tools" page shown in step 4.)
10. You will now see the "Merge Schemes: Choose Schemes to Merge" page:

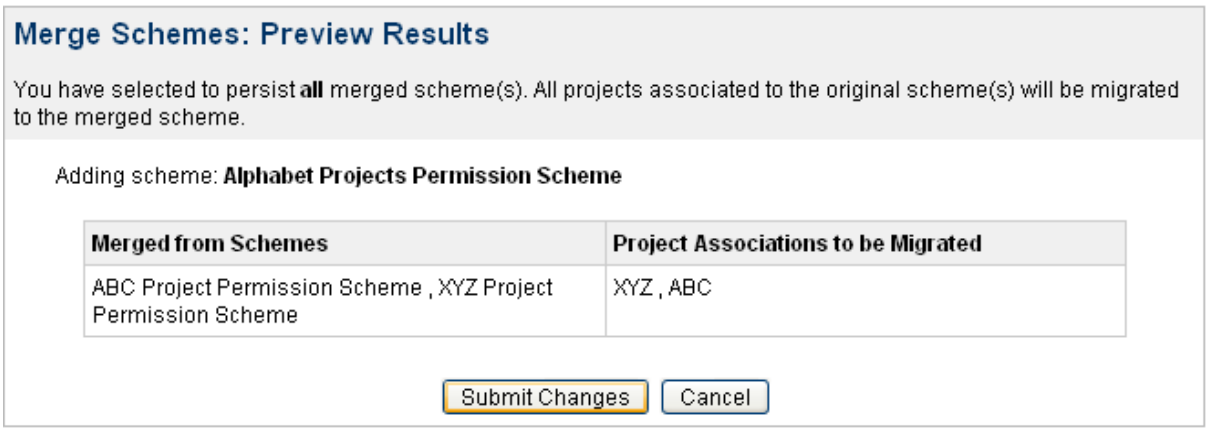


Choose schemes to merge

If you decide it is appropriate to merge the displayed schemes:

- Check the box next to the scheme names
- Type a name for the new scheme in the "New Scheme Name" box
- Click the "Preview Changes" button.

11. You will now see the "Merge Schemes: Preview Results" page:

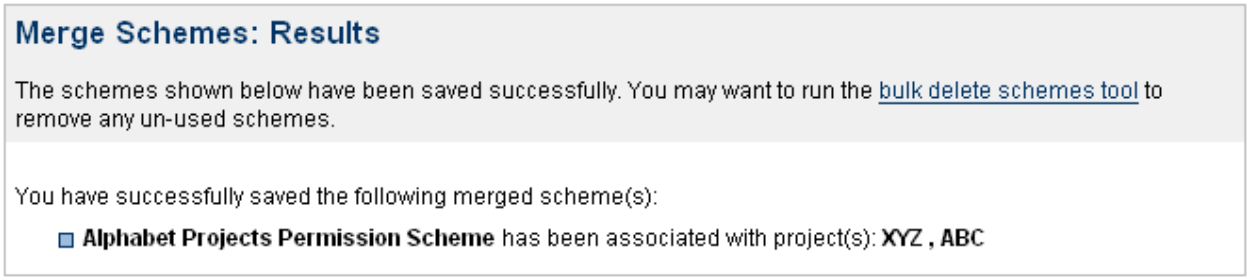


Preview scheme merge

If you are satisfied that the information shown on this page is correct, click the "Submit Changes" button to:

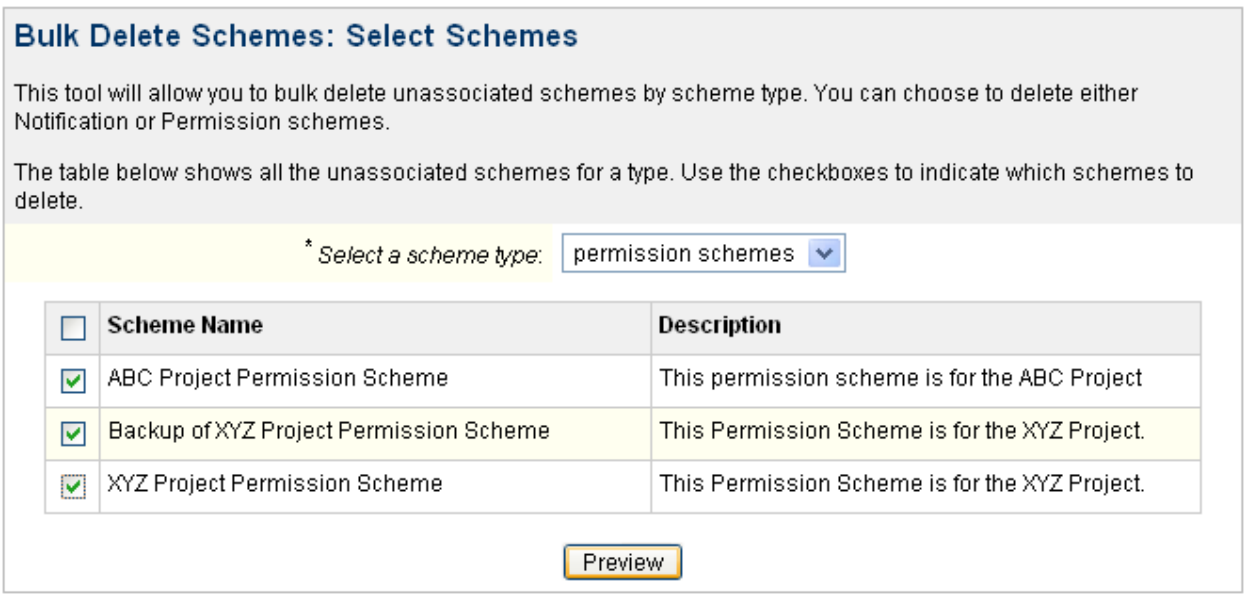
- create the new scheme whose name is shown in bold
- associate the projects (in the right column of the table) with the new scheme
- unassociate the existing schemes (in the left column of the table) from the projects. These schemes can then be deleted using the "Bulk Delete Schemes Tool (see step 15).

12. You will now see confirmation of the above changes on the "Merge Schemes: Results" page:



Results of merging schemes

13. Click the "Bulk Delete Schemes Tool" link. (Note: this link is also available from the "Scheme Tools" page shown in step 4.)
14. You will now see the "Bulk Delete Schemes: Select Schemes" page:



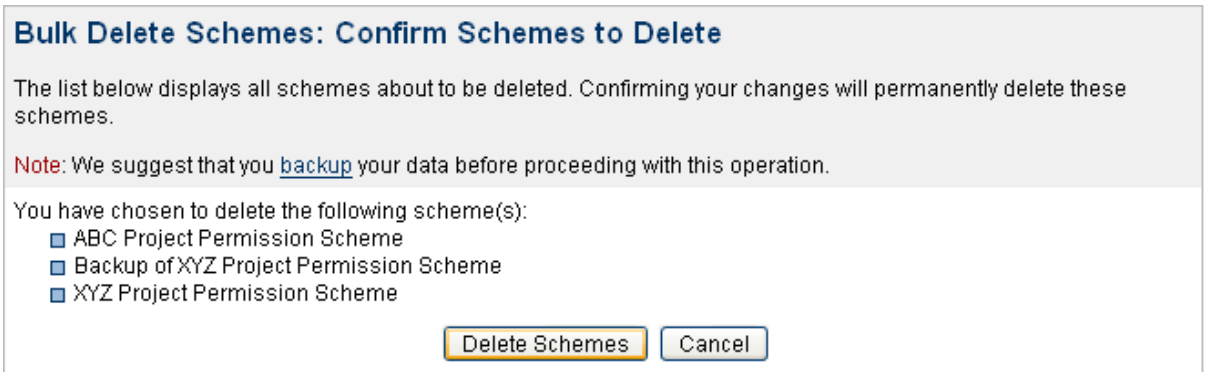
Choosing schemes to delete

If you decide it is appropriate to delete the displayed schemes:

- Check the box next to the scheme names
- Type a name for the new scheme in the "New Scheme Name" box
- Click the "Preview" button.

Note that deleting these schemes will not affect any projects, as this page only displays schemes that are not associated with projects.

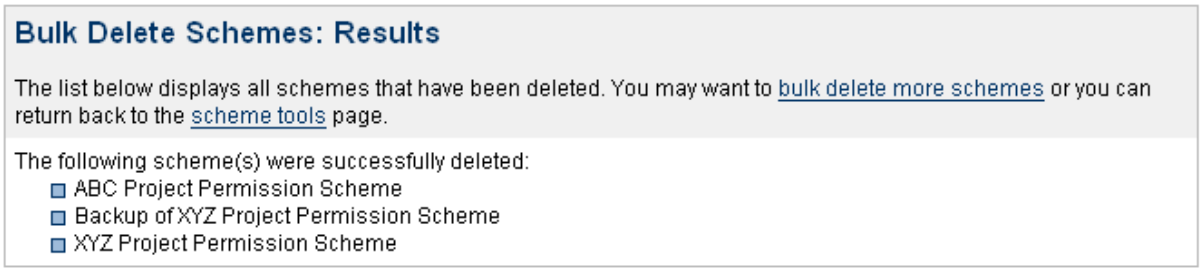
15. You will now see the "Bulk Delete Schemes: Confirm Schemes to Delete" page:



Confirming bulk delete

If you are satisfied that the information shown on this page is correct, click the "Delete Schemes" button.

16. You will now see the "Bulk Delete Schemes: Results" page, confirming that the unused schemes have been deleted:



Schemes deleted

1.5.5. Enabling Public Signup and CAPTCHA

For some organisations it is appropriate to enable *signup*, which allows visitors to immediately create their own JIRA user accounts. If signup is not enabled, then only a JIRA administrator can create new user accounts.

For example, enabling signup can be useful if you are using JIRA as a support system and have a very large number of potential users, of which only some will need to log support tickets.

Note:

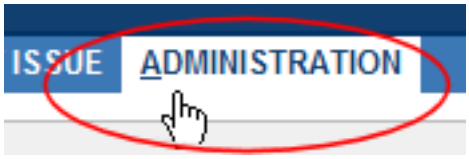
For security reasons, even if you enable signup, it is still necessary for users to have the appropriate project permissions before they can see or create issues. Note that you can use automatic group membership to add all new users to appropriate groups.

If your JIRA server is accessible from outside your organisation's firewall, and you have enabled signup, then you may want to also enable *CAPTCHA*. CAPTCHA helps ensure that only real humans (and not automated spam systems) can sign themselves up to JIRA. When CAPTCHA is enabled, visitors will need to recognise a distorted picture of a word (see example below), and must type the word into a text field. This is easy for humans to do, but very difficult for

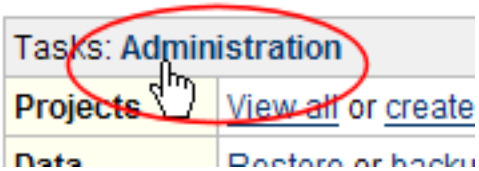
computers.

1.5.5.1. Enabling Public Signup

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

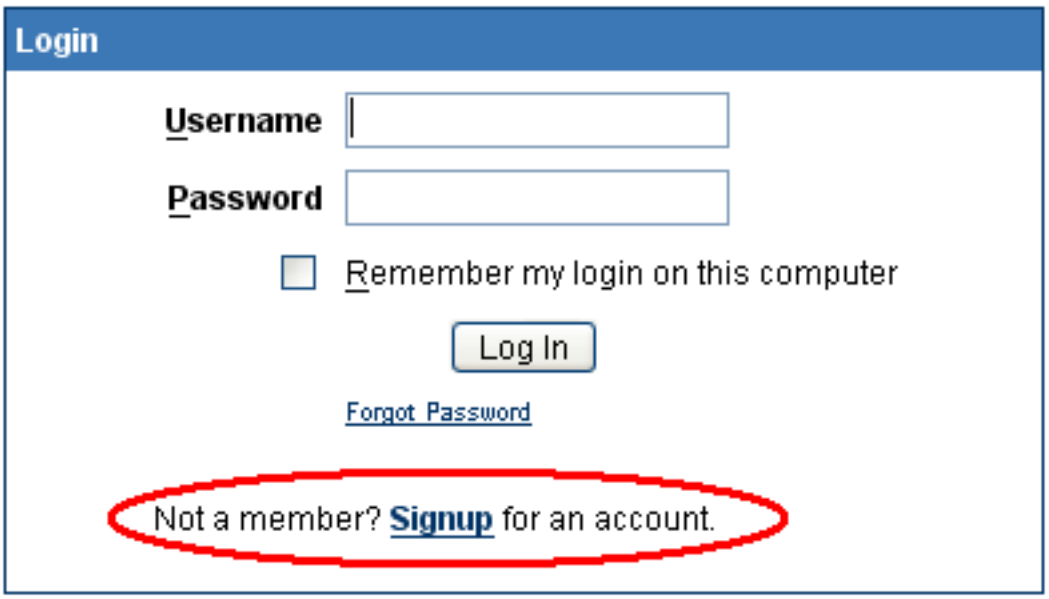


Link to Administration section



Link to Administration section

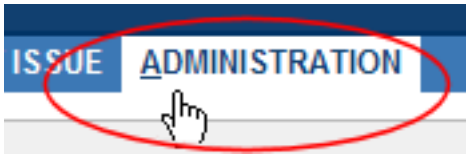
3. In the left navigation column, click "General Configuration".
4. This will display the "General Configuration" screen. Click the "**Edit Configuration**" link at the bottom of the screen.
5. In the **"Mode"** drop-down, select **"Public"**.
6. Click the **"Update"** button at the bottom of the screen.
7. Log out of JIRA, then click the "Log In" link at the top right of the screen and verify that the **"Signup"** link is displayed at the bottom of the login screen:



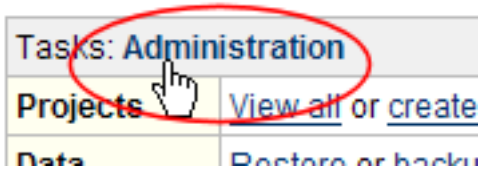
Login Screen (with Signup enabled)

1.5.5.2. Enabling CAPTCHA

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

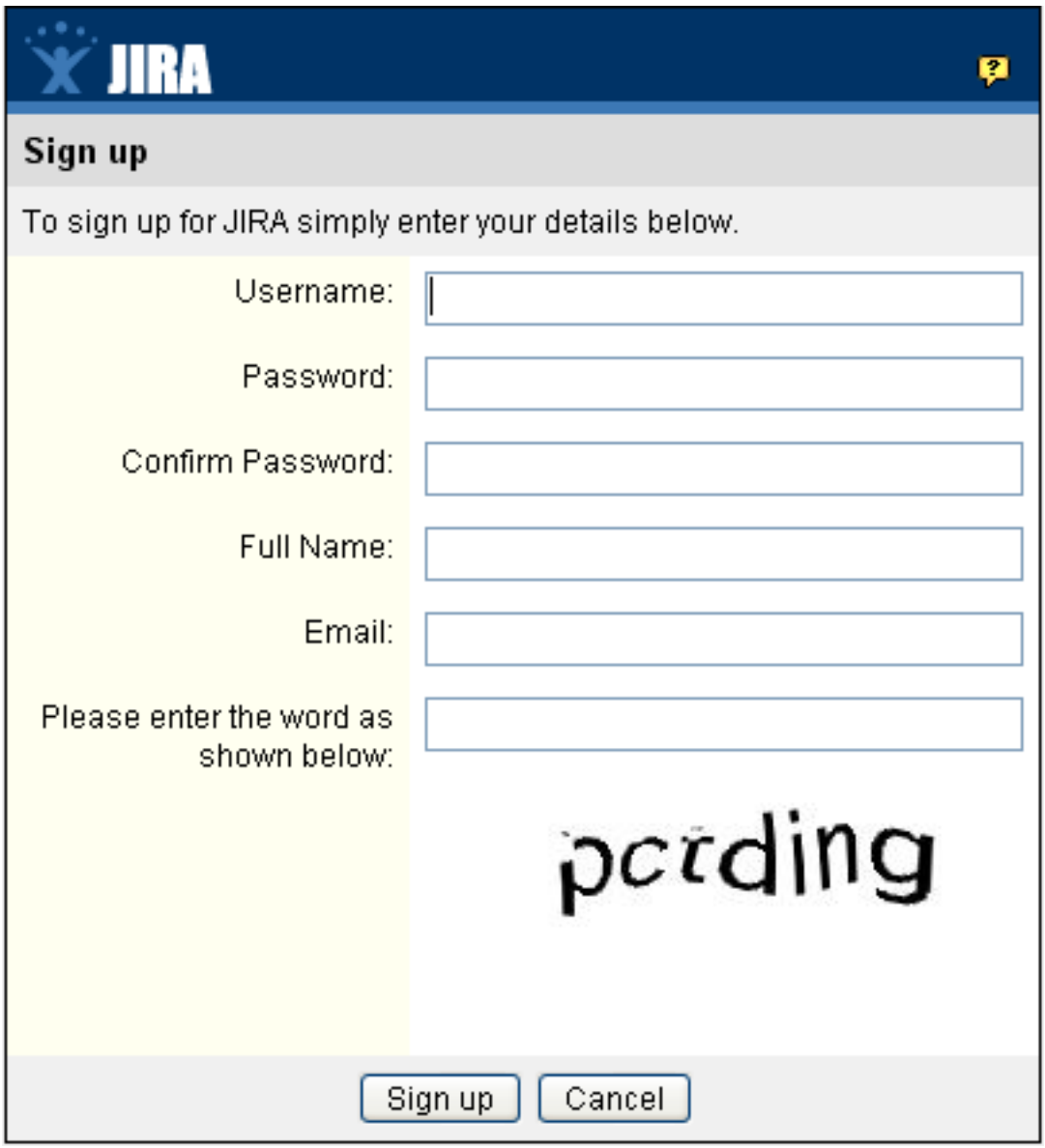


Link to Administration section



Link to Administration section

3. In the left navigation column, click "General Configuration".
4. This will display the "General Configuration" screen. Click the "**Edit Configuration**" link at the bottom of the screen.
5. Locate **"CAPTCHA on signup"** and select **"On"**.
6. Click the **"Update"** button at the bottom of the screen.
7. Log out of JIRA, click the "Log In" link at the top right of the screen, then click the **"Signup"** link and verify that a random sequence of letters is displayed at the bottom of the **"Signup"** screen — e.g. "pctding" in the following screenshot:



Signup Screen (with CAPTCHA enabled)

1.5.6. LDAP Integration

Many organisations have an LDAP directory acting as a centralised database of system users. JIRA is able to authenticate users against their LDAP password.

1.5.6.1. About JIRA's LDAP Integration

In JIRA, user management is handled by OSUser, a pluggable user management framework. OSUser is configured through the **WEB-INF/classes/osuser.xml** file.

Only password-checking for LDAP users is done in JIRA

The main point to realise is that user profiles are *still managed in JIRA* (the OFBizProfileProvider in osuser.xml). Only the password lookup is done against LDAP, and only if the JIRA username coincides with a LDAP username.

Technically, this behaviour is due to Credentials (password) checking being a separate operation to user-profile lookups. The profile can be loaded from the JIRA database, but the password looked up

from LDAP. Furthermore, multiple credentials providers can be specified (here, LDAP and OSUser), and if one fails, the other will be used. This allows non-LDAP users to log in with their JIRA password.

Not all LDAP users have JIRA access

Another effect of this implementation is that LDAP users *do not automatically have access to JIRA*. A JIRA account must be created for each user wishing to use JIRA. You can bulk-create users from LDAP with this LDAP user importer.

This is because each JIRA user has a set of **groups** (for example, 'jira-users') stored in their profile. Without an associated group, that user can do nothing; not even browse JIRA (they lack the 'use' permission).

Thus, for an LDAP user to be able to use JIRA, a JIRA administrator must create an account for them, and assign them to a group (typically 'jira-user'). The password in this JIRA account will be ignored, as the LDAP password will override it.

Planned Improvements

In future, we plan to more tightly integrate LDAP into JIRA, so that LDAP groups can be mapped to JIRA groups, and user management can be fully externalised (see Plans for JIRA's LDAP integration). We'd like feedback as we go further with this; please see this issue report for current status, and add your use-case as a comment. Thanks!

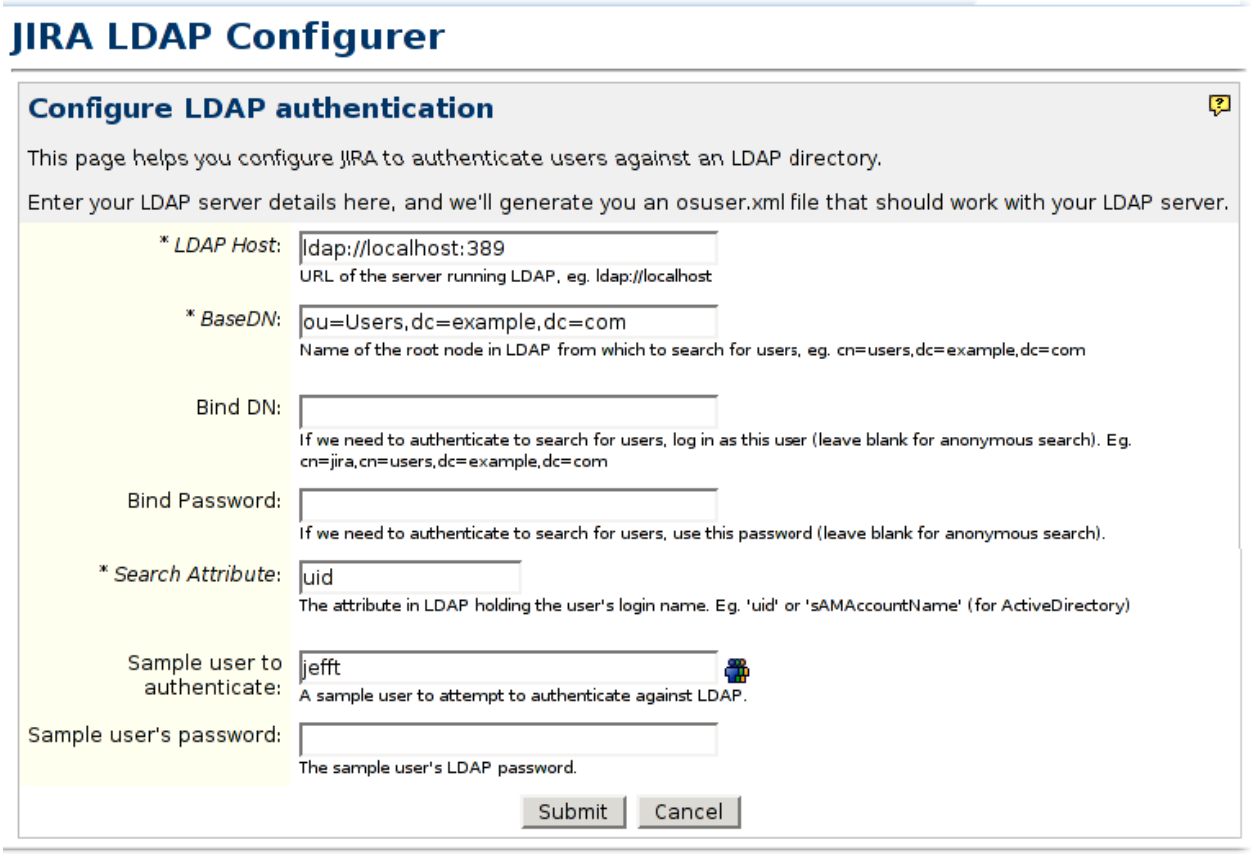
Atlassian Crowd

Atlassian Crowd is Atlassian's single sign-on product. Crowd provides an interface for managing users and groups, and provides an OSUser implementation that allows JIRA to fully delegate users and groups to Crowd, and hence LDAP. Thus while we still plan to develop tighter JIRA-LDAP integration, purchasers of Crowd can achieve this right now by using Crowd as an intermediary. See Crowd's JIRA integration documentation for details.

1.5.6.2. Step 1. Configuring LDAP Integration

JIRA Enterprise Edition

JIRA Enterprise contains a configuration utility which lets you auto-generate a valid **osuser.xml** file. This can be accessed from **Admin -> System -> LDAP**:



LDAP Configurer in JIRA Enterprise

JIRA Professional & Standard Editions

In **JIRA Professional** and **JIRA Standard**, LDAP integration is configured by manually editing the **WEB-INF/classes/osuser.xml** file.

In the JIRA Standalone distribution, this file can be edited directly in **atlassian-jira/WEB-INF/classes/osuser.xml**. In the WAR/webapp distribution, it should be copied from **webapp/WEB-INF/classes/osuser.xml** to **edit-webapp/WEB-INF/classes/**, edited there, and then rebuilt into an updated .war file with the '`ant war`' command.

The default osuser.xml contains:

```
<opensymphony-user>
  <authenticator
      class="com.opensymphony.user.authenticator.SmartAuthenticator" />

  <provider
      class="com.atlassian.core.ofbiz.osuser.CoreOFBizCredentialsProvider">
      <property name="exclusive-access">true</property>
  </provider>

  <provider
      class="com.opensymphony.user.provider.ofbiz.OFBizProfileProvider">
    <property name="exclusive-access">true</property>
  </provider>

  <provider
      class="com.opensymphony.user.provider.ofbiz.OFBizAccessProvider">
    <property name="exclusive-access">true</property>
  </provider>

</opensymphony-user>
```

CredentialsProviders are responsible for checking usernames and passwords, which is what we are interested in here. The default **CoreOFBizCredentialsProvider** looks in the JIRA database. We are going to add a **LDAPCredentialsProvider**, so that LDAP users can also be authenticated:

```
<opensymphony-user>
  <authenticator class="com.opensymphony.user.authenticator.SmartAuthenticator" />

<provider
      class="com.opensymphony.user.provider.ldap.LDAPCredentialsProvider">
 <property name="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</property
 <property name="java.naming.provider.url">ldap://localhost:389</property>
 <property name="searchBase">dc=atlassian,dc=com</property>
 <property name="uidSearchName">uid</property>
 <!--
 <property name="java.naming.security.principal">cn=Manager,dc=atlassian,dc=com</proper
 <property name="java.naming.security.credentials">secret</property>
 <property name="exclusive-access">true</property>
-->
</provider>

<provider class="com.atlassian.core.ofbiz.osuser.CoreOFBizCredentialsProvider">
  <property name="exclusive-access">true</property>
</provider>

<provider class="com.opensymphony.user.provider.ofbiz.OFBizProfileProvider">
  <property name="exclusive-access">true</property>
</provider>

<provider class="com.opensymphony.user.provider.ofbiz.OFBizAccessProvider">
  <property name="exclusive-access">true</property>
</provider>

</opensymphony-user>
```

It is necessary to use *both* the LDAP and OFBiz providers, and the order must be as shown (LDAP first).

Some LDAP properties that are commonly set here are:

Property	Required	Description	Example
java.naming.factory.initial	Yes	Specifies that JNDI (the directory API) should use the LDAP implementation	Should always be **com.sun.jndi.ldap.LdapCtxFactory**
java.naming.provider.url	Yes	LDAP URL of your server	ldap://localhost:389
searchBase	Yes	The node in the LDAP tree to search below for usernames.	The root LDAP node is typically called 'dc=*companyname*,dc=com', and user account nodes are usually stored in a subtree, like 'cn=Users,dc=*companyname*,dc=com'.
uidSearchName	Yes	Attribute expected to contain username	Typically **uid**, or **sAMAccountName** for MS ActiveDirectory.
java.naming.security.principal	No	Username to initially log in to LDAP as. Not required if anonymous user lookups are allowed.	Eg. 'cn=Administrator,cn=Users,dc=*company*

java.naming.security.credentials	No	Password for initial login. Not required if anonymous lookups are allowed.	
providerName	No	unique name for this LDAP provider. Useful when you specify multiple LDAP providers (allowing fallback), and need to distinguish them in the debug logs.	
cacheTimeout	No	The value in milliseconds for duration of password caching. Password caching reduces the load JIRA will put on the LDAP server. Only successful authentication attempts are cached. The default value is 30 minutes (1,800,000 ms).	0 (to disable caching)

The full list of properties is specified in the JNDI documentation.

Once you have made this modification and restarted JIRA, JIRA users **whose username also exists in LDAP** will be authenticated against their LDAP password.

1.5.6.3. Step 2. Disabling JIRA's Password Management

Once you have LDAP-based password checking working, you should go to **Admin -> General Configuration**, and turn on **External password management** (see Configuring JIRA documentation). This will disable the "change my password" links in the JIRA interface, ensuring that passwords are now only managed via LDAP.

1.5.6.4. Configuration Notes

LDAP over SSL

With plain LDAP, passwords may be passing over the network unencrypted, which (depending on your network security) may be a security problem. If you wish to connect to LDAP over SSL, see the Connecting to SSL services guide for details on how to import the SSL server's public key. In **osuser.xml**, you would need to use **ldaps://** in the URL if you have port 636 dedicated to LDAP over SSL.

Multiple LDAP trees (eg. ActiveDirectory domains)

If you wish to authenticate users from multiple LDAP directories or different trees in the same directory, simply edit the **OSUser** file and add a **LDAPCredentialsProvider** section for each (see 'Configuring LDAP Integration' above). JIRA will query them in order, and the first one containing the requested user will be used for password checking. As soon as a user is found, the password is checked and no further processing is done (ie. only one password will work).

If you have more than one **LDAPCredentialsProvider** it is a good idea to give each a unique **providerName** attribute for debugging purposes.

ActiveDirectory users note: a better approach to searching multiple trees is to set up an Active Directory **Global Catalog**. This is an AD instance which mirrors records in other instances. Searching the Global Catalog is thus equivalent to searching all mirrored LDAP directories. This is faster and more reliable than JIRA's LDAP fallback.

If you have a Global Catalog set up, it can be searched via LDAP on port 3268 (eg. ldap://adserver:3268) or 3269 for SSL (eg. ldaps://adserver:3269). See this guide for more information.

LDAP on Linux

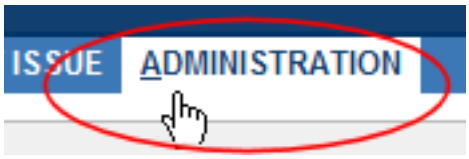
See these notes on how to set up a LDAP directory on a Linux server for use with JIRA.

1.5.6.5. Debugging

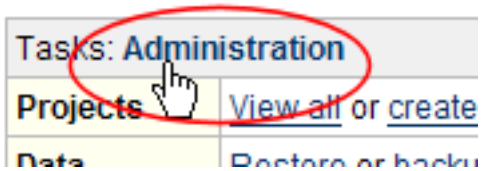
Cannot create osuser.xml file via JIRA LDAP Configurer

To see exactly why the JIRA LDAP Configurer is failing, follow these steps:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Under the '**System**' sub-menu in the left-hand navigation column, click the '**Logging & Profiling**' link.
4. The '**Logging & Profiling**' page will display. Click the '**Edit**' link next to '**com.atlassian.jira.web.action.util.LDAPConfigurer**'
5. Change the logging level to 'DEBUG', then click '**Update**'. This temporarily turns up logging for the JIRA LDAP Configurer.
6. When you next try to submit your LDAP server details via the JIRA LDAP Configurer, you should see extra logs on stdout.

Cannot authenticate against LDAP password

If JIRA does not authenticate against the LDAP password, then something is probably wrong with your setup. First, ensure that the user you are trying to connect as has a JIRA account (see above). Make sure you have the LDAP connection details correct (basename, uid, username/password). These details are best discovered with the help of an LDAP browser such as Apache Directory Studio or JXplorer.

To see exactly why LDAP authentication is failing, follow these steps:

1. Edit `log4j.properties` (instructions), locate the lines:

```
log4j.category.com.opensymphony = WARN, console
log4j.additivity.com.opensymphony = false
```

Duplicate these, and change the first copy to:

```
log4j.category.com.opensymphony.user.provider.ldap = DEBUG, console
log4j.additivity.com.opensymphony.user.provider.ldap = false
```

Restart JIRA to apply your change. This change will turn up logging for the LDAP authentication module.

2. When next trying to log in, you should see extra logs on stdout. A successful authentication looks like this:

```
DEBUG [user.provider.ldap.LDAPCredentialsProvider] LDAPCredentialsProvider $Revision: 1
DEBUG [user.provider.ldap.LDAPCredentialsProvider] 'jturner' will be handled by LDAP
DEBUG [user.provider.ldap.LDAPCredentialsProvider] 'jturner' will be handled by LDAP
DEBUG [user.provider.ldap.LDAPCredentialsProvider] 'jturner' will be handled by LDAP
DEBUG [user.provider.ldap.LDAPCredentialsProvider] Doing initial search:
 username='cn=admin,dc=atlassian,dc=com', password='secret', base='ou=People,dc=atlassi
DEBUG [user.provider.ldap.LDAPCredentialsProvider] Found users
DEBUG [user.provider.ldap.LDAPCredentialsProvider] Searching below 'uid=jturner,ou=Peop
DEBUG [user.provider.ldap.LDAPCredentialsProvider] User 'jturner' successfully authenti
```

This log was generated with the following to osuser.xml:

```
<provider class="com.opensymphony.user.provider.ldap.LDAPCredentialsProvider">
<property name="java.naming.factory.initial">com.sun.jndi.ldap.LdapCtxFactory</property
<property name="java.naming.provider.url">ldap://localhost:389</property>
<property name="searchBase">ou=People,dc=atlassian,dc=com</property>
<property name="uidSearchName">uid</property>
<property name="java.naming.security.principal">cn=admin,dc=atlassian,dc=com</property>
<property name="java.naming.security.credentials">secret</property>
<property name="exclusive-access">true</property>
</provider>
```

3. If you have problems, try emulating the operations performed by the LDAP authentication provider. The LDAP authentication provider works by first doing an search for the specified username, searching from base **searchBase**, using query **uidSearchName**=*username*, authenticating using the principal and credentials properties if present, or doing an anonymous search otherwise. If an entry is found, it then tries to log in to LDAP using the specified matching username and specified password.

Note:

If you get an error message **javax.naming.PartialResultException: Unprocessed Continuation Reference(s)**, try adding `<property name="java.naming.referral">follow</property>` to the LDAPCredentialsProvider section.

1.5.7. Integrating with Crowd

Atlassian's Crowd identity management system can be integrated with JIRA. For more information please see the chapter titled 'Integrating JIRA with Crowd', in the Crowd documentation.

1.5.8. Trusted Applications

A 'trusted application' is an application that JIRA will allow to access specified functions on behalf of any user — without the user logging in to JIRA.

For example, when Confluence is configured as a trusted application, every Confluence user will see *exactly the same list of issues* when they view the Confluence 'JIRA Issues' macro as they see when they use the JIRA Issue Navigator as a logged-in JIRA user. Likewise, the Confluence 'JIRA Portlet' macro will appear exactly the same as it does on the user's JIRA Dashboard.

At this time, Confluence (version 2.7 or later) is the only application that can be configured as a trusted application.

Trusted applications are a potential security risk. When you configure a trusted application, you are allowing the application to access JIRA as *any user*. By doing this, you are bypassing all the built-in JIRA security measures. Do not configure a trusted application unless you trust all code in this application to behave itself at all times, and are sure that the application will maintain the security of its private key.

Note:

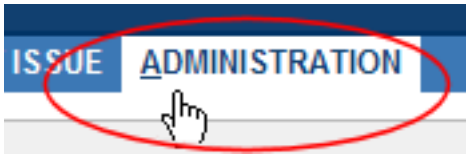
All of your trusted application's users must also be JIRA users, and the usernames in both systems must be identical.

1.5.8.1. Adding a trusted application

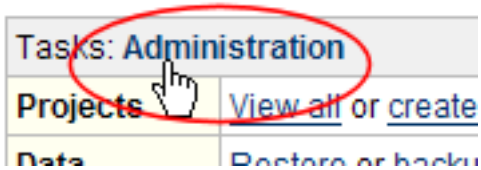
Before you begin: Note that configuring a trusted application requires the transmission of sensitive data. To prevent 'man-in-the-middle' attacks, it is recommended that you use SSL while configuring a trusted application.

To add a trusted application,

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

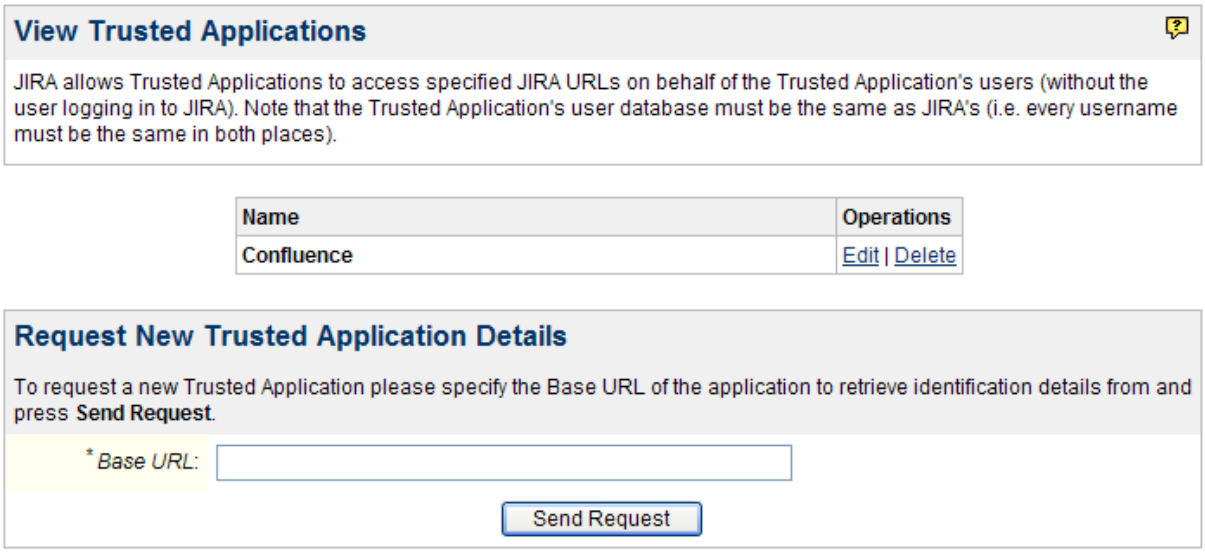


Link to Administration section



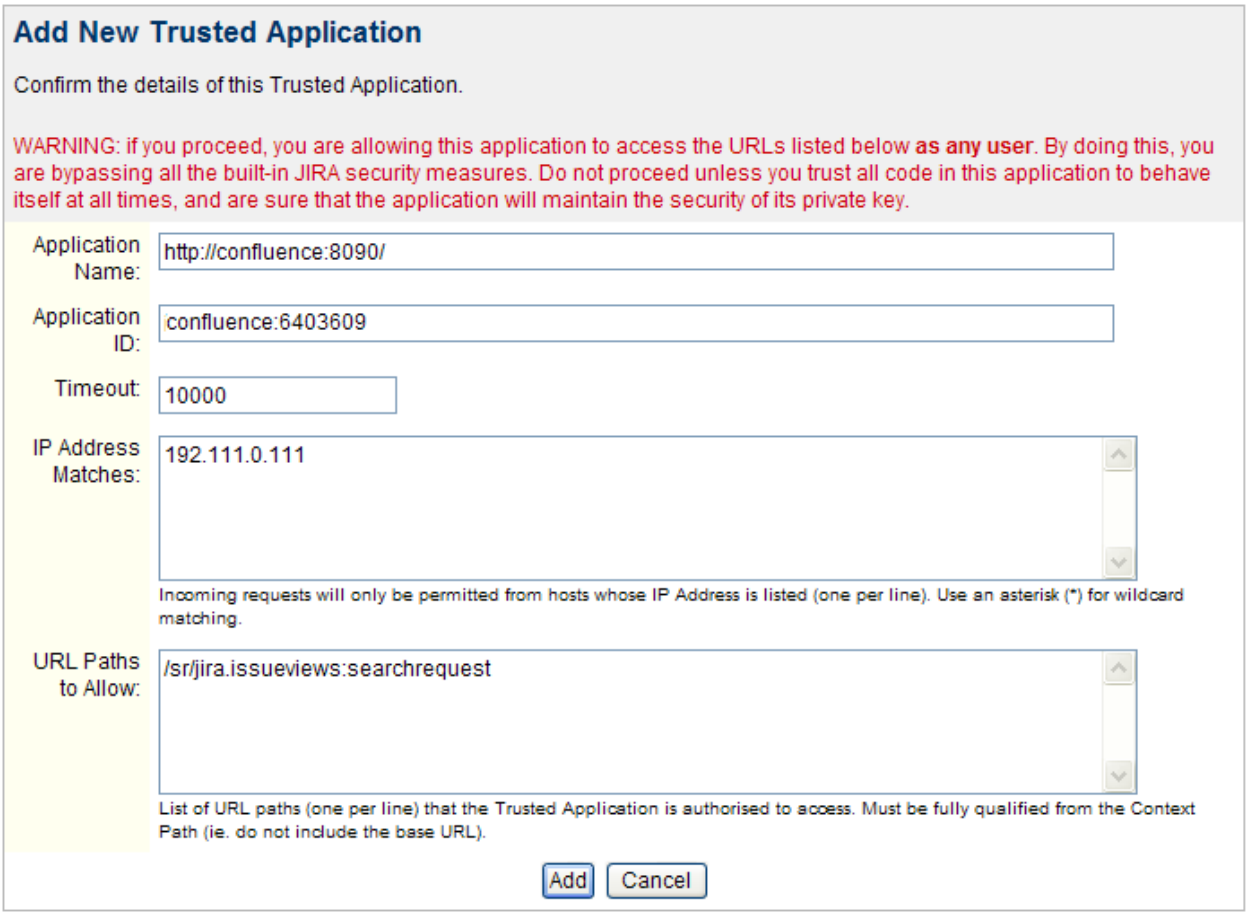
Link to Administration section

3. In the left-hand panel, under the title '**System**', click the '**Trusted Applications**' link. The '**Trusted Applications**' page will be displayed, showing a list of configured trusted applications (if any). The '**Request New Trusted Application Details**' box is shown below the list.



Trusted Applications

4. In the '**Base URL**' field, type the URL that you use to access the application you wish to add (e.g. 'http://confluence.mycompany.com:8080' or 'http://www.mycompany.com/confluence').
5. Click the '**Send Request**' button to retrieve the application's ID and public key. The '**Add New Trusted Application**' screen will be displayed:



Edit a Trusted Application

6. In the '**Application Name**' field, the URL which you typed on the previous page will be displayed. You can optionally change this if you wish (e.g. if you typed 'http://confluence.mycompany.com:8090' on the previous page, you might want to change it to just 'Confluence').

    - Note: the '**Application ID**' is generated automatically by JIRA and cannnot be edited.
7. In the '**Timeout**' field, type the number of milliseconds that JIRA will wait for a response when communicating with the trusted application.
8. In the '**IP Address Matches**' field, type the IP address (or multiple addresses, one per line) from which JIRA will accept requests on behalf of the trusted application. You can specify wildcard matches by using an asterisk (*), e.g. '`192.111.*.*`'.
    - If you are using a proxy server that makes an HTTP request on the client's behalf (e.g. Squid, mod_proxy), you need to add the proxy server's IP address to this field *as well as* all the clients' IP addresses.
    - If you are using a proxy server that passes the client's IP address directly via an application server's API (e.g. AJP for Tomcat, such as mod_jk or IIS's Tomcat Connector, or mod_caucho for Resin) — or if you are not using a proxy server — then you only need to enter the clients' IP addresses.

    If you are configuring a clustered instance of Confluence as a trusted application, you need to set up JIRA to receive requests from each Confluence node. If you do not set up each node appropriately, users may not be able to view any JIRA information in Confluence (e.g. a jiraissue macro request). You can set this up by either:
    - specifying each individual IP address for each node of the cluster separated by commas, e.g. `172.16.0.10, 172.16.0.11, 172.16.0.12`, or
    - specifying the IP address for your clustered Confluence instance using wildcards e.g. `172.16.0.*`
9. In the '**URL Paths to Allow**' field, type the JIRA URLs that the application will be allowed to access. Each URL corresponds to a particular JIRA function.

    By default, the following will be included:
    - '`/sr/jira.issueviews:searchrequest`' — This allows the application to search for JIRA issues.
    - '`/secure/RunPortlet`' — This allows the application to access JIRA dashboard portlets.
10. Click the '**Add**' button.
11. The '**Trusted Applications**' page will be displayed, with your new trusted application now shown in the list.

1.6. Security

1.6.1. Security overview

When configuring security for your JIRA instance, there are two areas to address:

- security within JIRA itself
- security in the external environment

1.6.1.1. Configuring security within JIRA

JIRA has a flexible security system which allows you to configure who can access JIRA, and what they can do/see within JIRA.

There are five types of security within JIRA:

1. Global permissions — these apply to JIRA as a whole (eg. who can log in).
2. Project permissions — organised into permission schemes, these apply to projects as a whole. (Eg. who can see the project's issues ('Browse' permission), create, edit and assign them.)
3. Issue security levels *(Enterprise Edition only)* — organised into security schemes, these allow the visibility of individual issues to be adjusted, within the bounds of the project's permissions.

4. Comment visibility — allows the visibility of individual comments (within an issue) to be restricted.
5. Work-log visibility — allows the visibility of individual work-log entries (within an issue) to be restricted.

1.6.1.2. Configuring security in the external environment

If your JIRA instance contains sensitive information, you may want to configure security in the environment in which your JIRA instance is running. Some of the main areas to consider are:

- Database:
  - If you are using an external database as recommended for production systems (i.e. you are not using the HSQL database that is bundled with JIRA Standalone), you should restrict access to the database that your JIRA instance uses.
  - If you are using the embedded HSQL database that is bundled with JIRA Standalone, you should restrict access to the directory in which you installed JIRA. (Note that the user which your JIRA instance is running as will require full access to this directory.)
- File system — you should restrict access to the following directories (but note that the user which your JIRA instance is running as will require full access to these directories):
  - Index directory
  - Attachments directory
- SSL — if you are running your JIRA instance over the Internet, you may want to consider using SSL.

1.6.2. Global Permissions

Global permissions are system wide.

> **Note:**
> See also project permissions, which apply to individual projects.

Global permissions are granted to groups of users.

This table lists the different global permissions and the functions they secure:

Global Permission	Explanation
JIRA System Administrators	Permission to perform all JIRA administration functions.
JIRA Administrators	Permission to perform most JIRA administration functions (see list of exclusions below).
JIRA Users	Permission to login to JIRA. (Note: Granting the **JIRA Users** permission to a group also means that all newly created users will be automatically added to that group.)
Browse Users	Permission to view a list of all JIRA user names and group names. Used for selecting users/groups in popup screens (such as the 'User Picker').
Create Shared Object	Permission to share a filter or dashboard globally or with groups of users. *(This applies only to JIRA Professional and Enterprise editions.)*
Manage Group Filter Subscriptions	Permission to manage (create and delete) group filter subscriptions. *(This applies only to JIRA*

	Professional and Enterprise editions.)
Bulk Change	Permission to execute the bulk operations within JIRA: Bulk Edit*, Bulk Move*, Bulk Workflow Transition, Bulk Delete* (*subject to project-specific permissions.) Note: The decision to grant the **Bulk Change** permission should be considered carefully. This permission grants users the ability to modify a collection of issues at once. For example, in JIRA installations configured to run in Public mode (ie. anybody can sign up and create issues), a user with the **Bulk Change** global permission and the **Add Comments** project permission could comment on *all* accessible issues. Undoing such modifications may not be possible through the JIRA application interface and may require changes made directly against the database (which is not recommended).

1.6.2.1. About 'JIRA System Administrators' and 'JIRA Administrators'

People who have the '**JIRA System Administrators**' permission can perform all of the administration functions in JIRA, while people who have only the '**JIRA Administrators**' permission cannot perform functions which could affect the application environment or network. This is useful for organisations which need to delegate some administrative privileges (e.g. creating users, creating projects) to particular people, without granting them complete rights to administer the JIRA system.

By default, the `jira-administrators` group has both the '**JIRA Administrators**' permission *and* the '**JIRA System Administrators**' permission. If you need some people to have only the '**JIRA Administrators**' permission (and not the '**JIRA System Administrators**' permission), you will need to use two separate groups, e.g.:

1. Create a new group (e.g. called `jira-system-administrators`).
2. Add to the `jira-system-administrators` group everyone who needs to have the '**JIRA System Administrators**' permission.
3. Grant the '**JIRA System Administrators**' permission to the `jira-system-administrators` group.
4. Remove the '**JIRA System Administrators**' permission from the `jira-administrators` group.
5. *(Optional, but recommended for ease of maintenance)* Remove from the `jira-administrators` group everyone who is a member of the `jira-system-administrators` group.

People who have the '**JIRA Administrators**' permission (and not the '**JIRA System Administrators**' permission) cannot do the following:

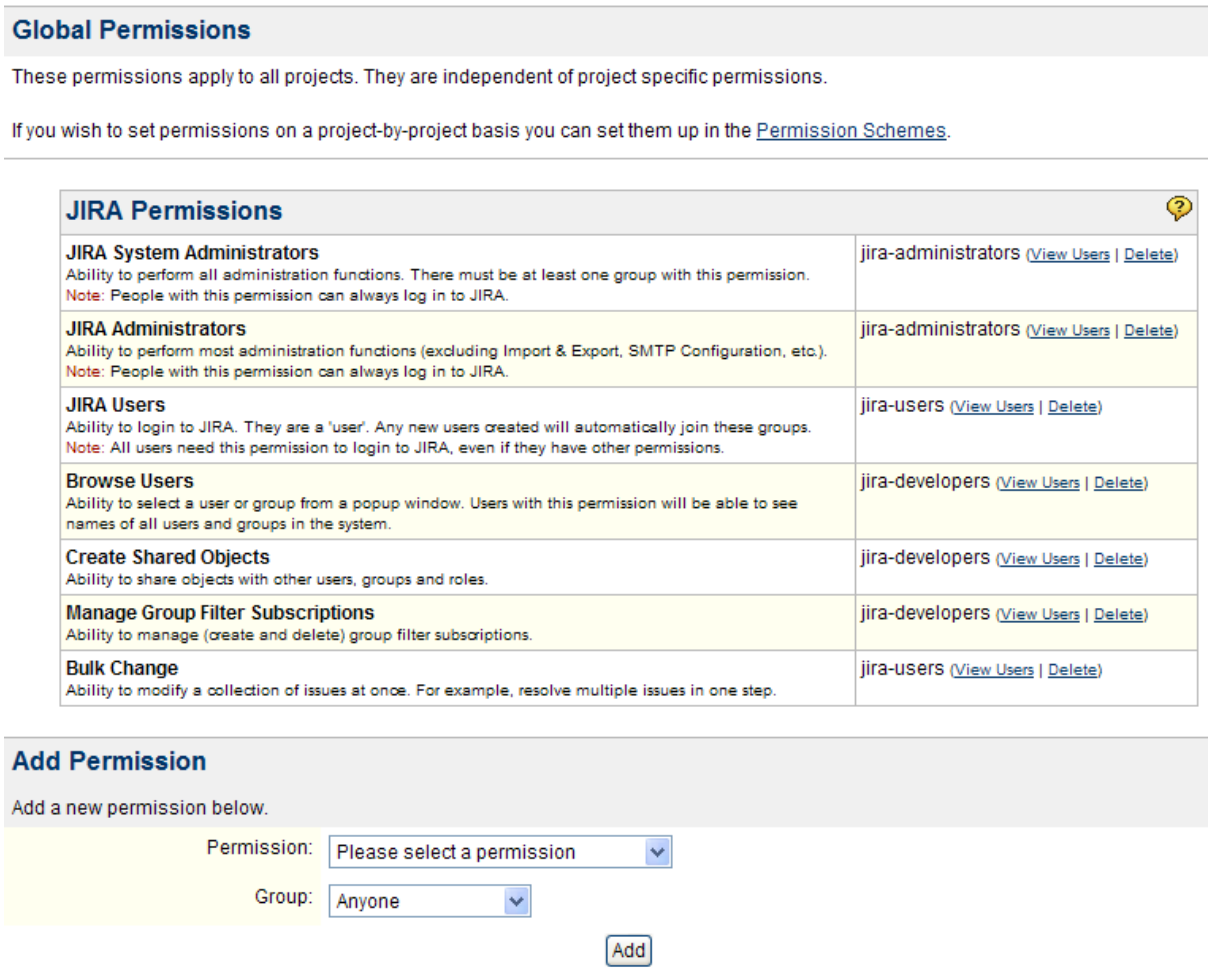
- Configure SMTP email (but note that they can fully administer email notification schemes).
- Configure a CVS source code repository (but note that they can associate a project with a configured repository).
- Configure listeners.
- Configure services.
- Change the index path (but note that they can reindex and optimise the index).
- Run the integrity checker.
- Access logging and profiling information.
- Access the scheduler.
- Export/backup JIRA data to XML.

- Import/restore JIRA data from XML.
- Import data from external systems (Bugzilla, Mantis, FogBugz, Excel/CSV).
- Import XML workflows into JIRA.
- Enable attachments, set the attachment path or size limit (but note that they can enable thumbnails).
- Enable trackbacks.
- Run Jelly scripts.
- Configure LDAP integration.
- Configure trusted applications.
- Access license details.
- Grant/revoke the '**JIRA System Administrators**' global permission.
- Edit (or Bulk Edit) groups that have the '**JIRA System Administrators**' global permission.
- Edit, change the password of or delete a user who has the '**JIRA System Administrators**' global permission.

It is recommended that people who have the '**JIRA Administrators**' permission (and not the '**JIRA System Administrators**' permission) are not given direct access to the JIRA filesystem or database.

1.6.2.2. Granting global permissions

1. Log in as a user with the '**JIRA Administrators**' global permission (or the '**JIRA System Administrators**' global permission, if you need to grant the '**JIRA System Administrators**' global permission).
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the **Administration** box on the dashboard.
3. In the panel on the left, under the title '**Global Settings**', click the link labelled '**Global Permissions**'. The '**Global Permissions**' page will be displayed, showing a list of the global permissions and which groups currently have them:



Global Permissions

The '**Add Permission**' box is shown below the list.

4. In the '**Permission**' drop-down list, select the global permission you wish to grant.
5. In the '**Group**' drop-down list, either:
   - select the group to which you wish to grant the permission; or
   - if you wish to grant the permission to non logged-in users, select '**Anyone**' (not recommended for production systems). Note that the '**JIRA Users**' permission (i.e. permission to log in) cannot be granted to '**Anyone**' (i.e. to non logged-in users) since this would be contradictory.

Note:

If you have a user limited license (e.g. personal license) and have reached your user limit, you will not be able to grant login permissions (i.e. jira-users permission) to any further groups without first reducing the number of users with login permissions.

1.6.3. Project permissions

Project permissions are created within Permission Schemes, which are then assigned to specific projects.

Project permissions can be granted to:

- Individual users
- Groups
- Project roles
- Issue roles such as 'Reporter', 'Project Lead' and 'Current Assignee' (JIRA Enterprise Edition only)

- 'Anyone' (eg. to allow anonymous access)
- A (multi-)user picker custom field.
- A (multi-)group picker custom field. This can either be an actual group picker custom field, or a (multi-)select-list whose values are group names.

The following table lists the different types of project permissions and the functions they secure. Note that, in JIRA Professional and Enterprise editions, project permissions can also be used in workflow conditions.

Project Permission	Explanation
Administer Projects	Permission to administer a project in JIRA. This includes the ability to edit project role membership (Enterprise Edition only), project components, project versions, and some project details ('Project Name', 'URL', 'Project Lead', 'Project Description').
Browse Projects	Permission to browse projects, use the Issue Navigator and view individual issues (except issues that have been restricted via Issue Security). Users without this permission will not know that the project exists.
View Version Control	Permission to view the version control information (CVS, Subversion, etc) for issues. Note that for CVS, to view the Version Control information the project needs to be associated with at least one Repository.
Create Issues	Permission to create issues in the project. (Note that the Create Attachments permission is required in order to create attachments.) Includes the ability to create sub-tasks (if sub-tasks are enabled).
Edit Issues	Permission to edit issues (excluding the 'Due Date' field — see the Schedule Issues permission). Includes the ability to convert issues to sub-tasks and vice versa (if sub-tasks are enabled). Note that the Delete Issue permission is required in order to delete issues. The Edit Issue permission is usually given to any groups or project roles who have the Create Issue permission (perhaps the only exception to this is if you give **everyone** the ability to create issues — it may not be appropriate to give everyone the ability to edit too). Note that all edits are recorded in the Issue Change History for audit purposes.
Schedule Issues	Permission to set and edit the 'Due Date' of issues.
Move Issues	Permission to move issues from one project to another, or (in Enterprise Edition only) from one workflow to another workflow within the same project. Note that a user can only move issues to a project for which they have Create Issue permission.
Assign Issues	Permission to assign issues to users. (See also Assignable User permission below)
Assignable User	Permission to be assigned issues. (Note that this does

	not include the ability to assign issues; see Assign Issue permission above.)
Resolve Issues	Permission to resolve and reopen issues. This also includes the ability to set the 'Fix For version' field for issues.
Close Issues	Permission to close issues. (This permission is useful where, for example, developers resolve issues and testers close them.)
Modify Reporter	Permission to modify the 'Reporter' of an issue. This allows a user to create issues 'on behalf of' someone else.
Delete Issues	Permission to delete issues. Think carefully about which groups or project roles you assign this permission to; usually it will only be given to administrators. Note that deleting an issue will delete all of its comments and attachments, even if the user does not have the Delete Comments or Delete Attachments permissions. However, the Delete Issues permission does not include the ability to delete individual comments or attachments.
Link Issues	Permission to link issues together. (Only relevant if Issue Linking is enabled.)
Set Issue Security	Permission to set the security level on an issue to control who can access the issue (Enterprise Edition only). Only relevant if issue security has been enabled.
View Voters and Watchers	Permission to view the voter list and watcher list of an issue.
Manage Watcher List	Permission to manage (i.e. view/add/remove users to/from) the watcher list of an issue.
Add Comments	Permission to add comments to issues. Note that this does not include the ability to edit or delete comments.
Edit All Comments	Permission to edit any comments, regardless of who added them.
Edit Own Comments	Permission to edit comments that were added by the user.
Delete All Comments	Permission to delete any comments, regardless of who added them.
Delete Own Comments	Permission to delete comments that were added by the user.
Create Attachments	Permission to attach files to an issue. (Only relevant if Attachments are enabled.) Note that this does not include the ability to delete attachments.
Delete All Attachments	Permission to delete any attachments, regardless of who added them.

Delete Own Attachments	Permission to delete attachments that were added by the user.
Work On Issues	Permission to log work done against an issue, i.e. create a worklog entry. (Only relevant if Time Tracking is enabled.)
Edit Own Worklogs	Permission to edit worklog entries that were added by the user. (Only relevant if Time Tracking is enabled.)
Edit All Worklogs	Permission to edit any worklog entries, regardless of who added them. (Only relevant if Time Tracking is enabled.)
Delete Own Worklogs	Permission to delete worklog entries that were added by the user. (Only relevant if Time Tracking is enabled.)
Delete All Worklogs	Permission to delete any worklog entries, regardless of who added them. (Only relevant if Time Tracking is enabled.)

1.6.3.1. Permission Schemes

What is a Permission Scheme?

A permission scheme is a set of user/group/role assignments for the project permissions listed above. Every project has a permission scheme. One permission scheme can be associated with multiple projects.

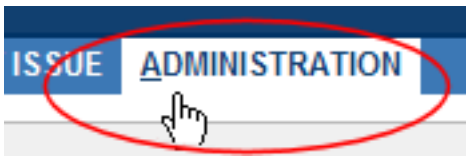
Why Permission Schemes?

In many organisations, multiple projects have the same needs regarding access rights. (For example, only the specified project team may be authorised to assign and work on issues.)

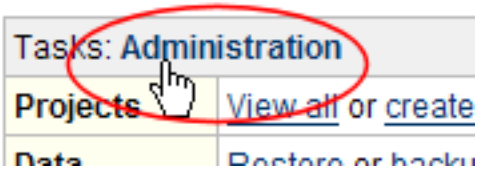
Permission schemes prevent having to set up permissions individually for every project. Once a permission scheme is set up it can be applied to all projects that have the same type of access requirements.

1.6.3.2. Creating a Permission Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

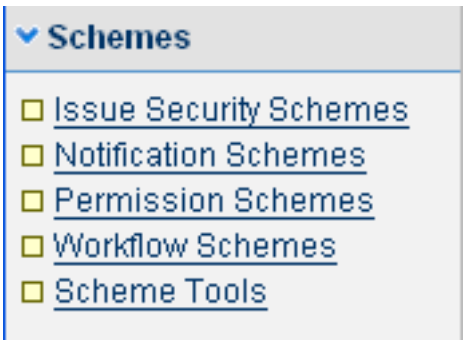


Link to Administration section



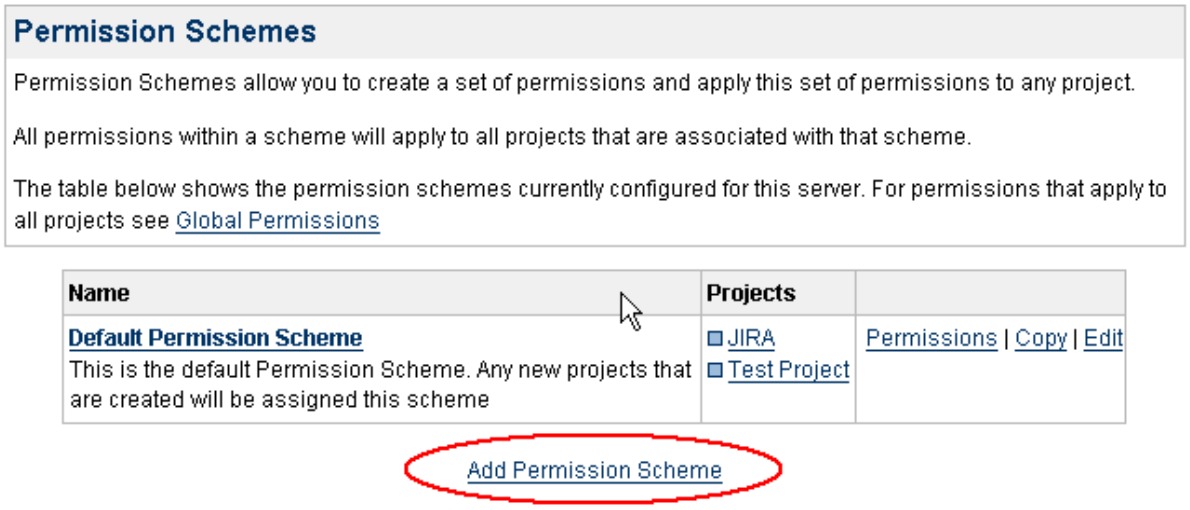
Link to Administration section

3. On the panel on the left, under the title "Schemes", click on the link labelled "Permission Schemes".



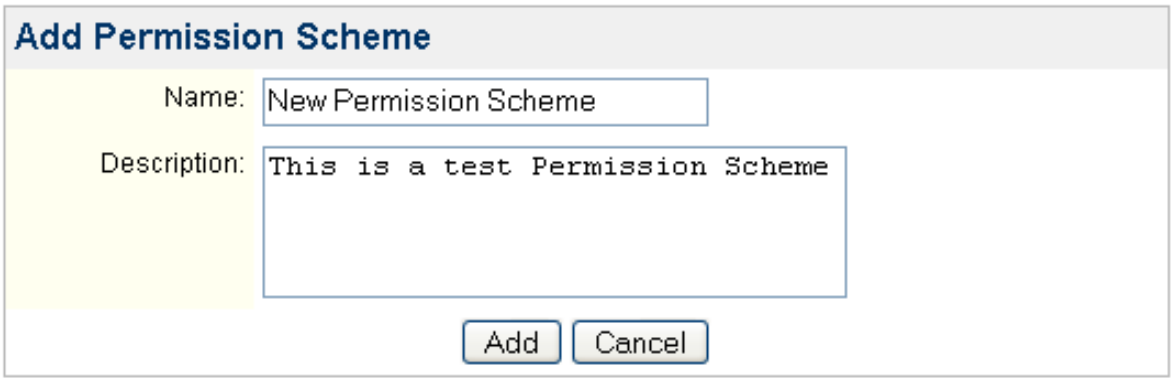
View Permission Schemes

4. This will display the "Permission Schemes" page. This page lists all of the Permission Schemes that JIRA currently has. Click on the "Add Permission Scheme" link.



View Permission Schemes

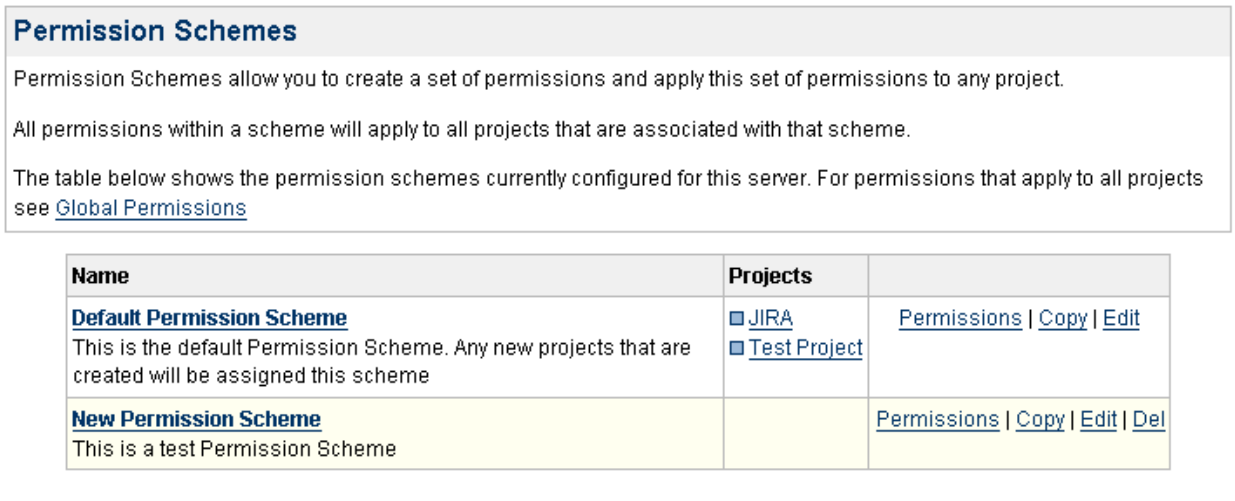
5. In the "Add Permission Scheme" form, enter a name for the scheme, and a short description of the scheme. Click on the "Add" button.



Add a Permission Scheme

6. You will return to the "Permission Schemes" page which now contains the newly added

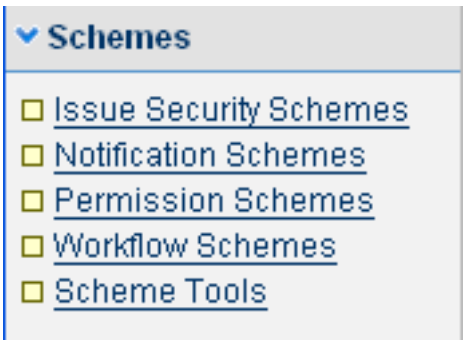
scheme.



View Permission Schemes

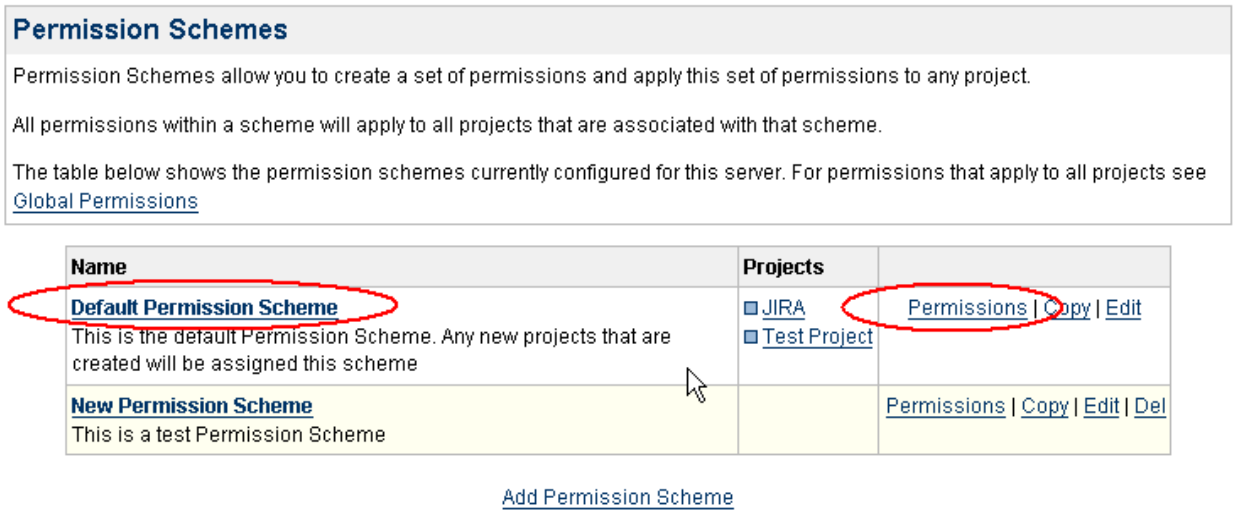
1.6.3.3. Adding Users, Groups or Roles to a Permission Scheme

1. On the panel on the left, under the title "Features", click on the link labelled "Permission Schemes".



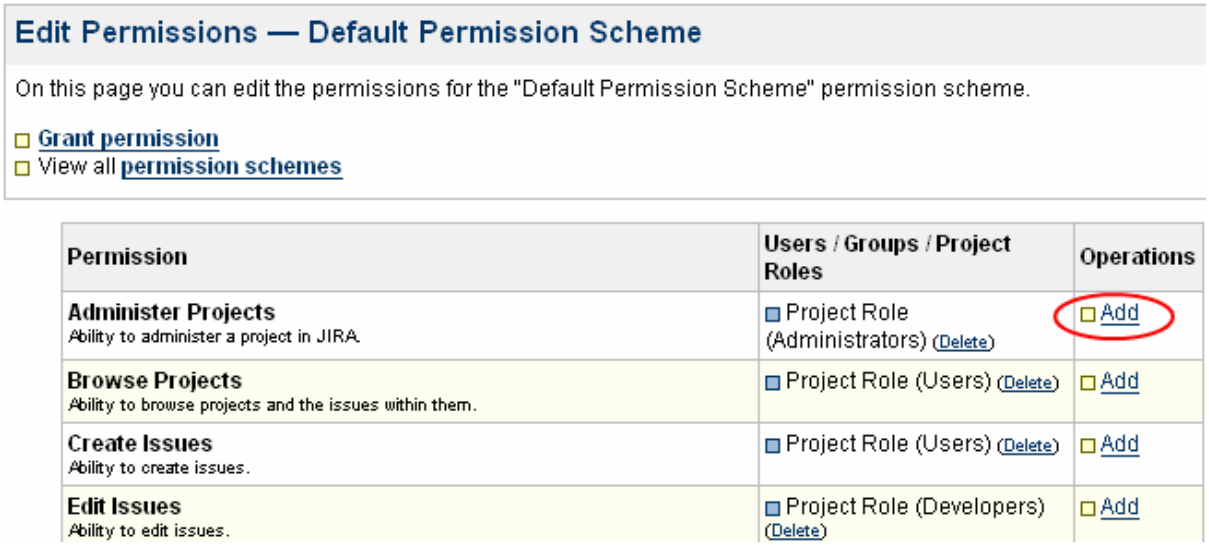
View Permission Schemes

2. Click on the "Permissions" link or on the name of the Permission Scheme to show a list of permissions



edit Permissions

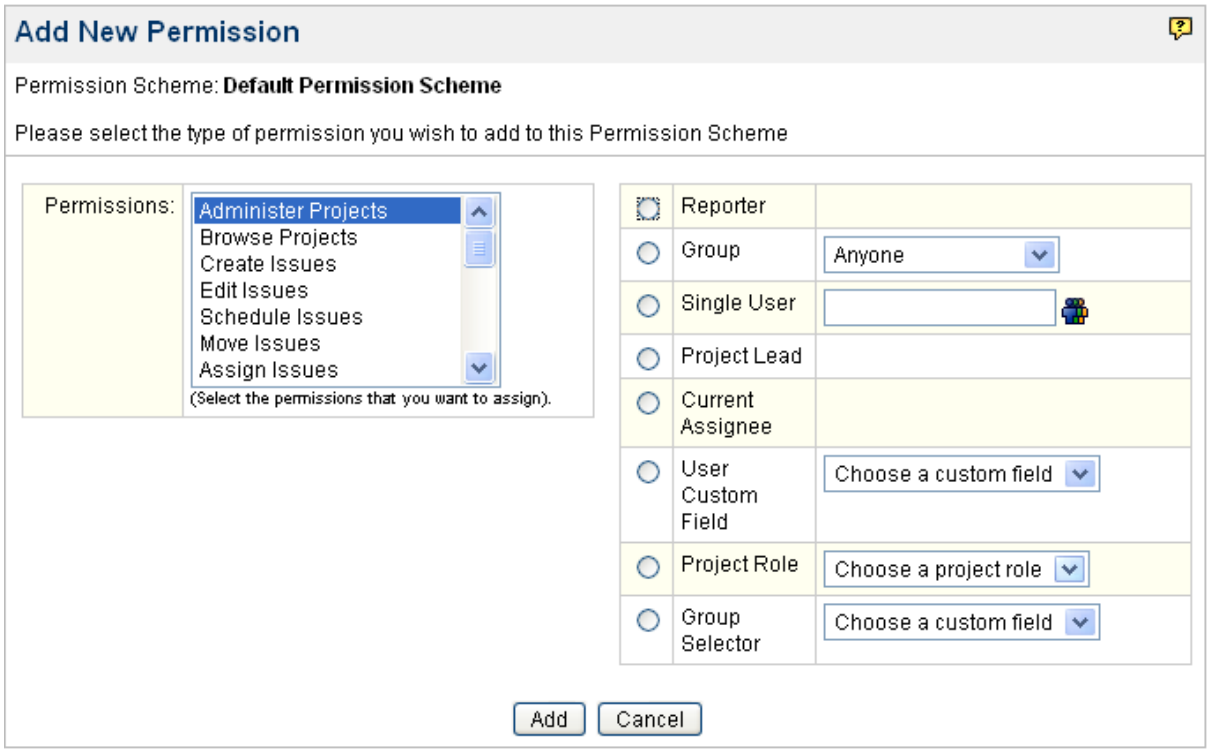
3. Click the "Add" link in the "Operations" column.



Add Users to Permissions

4. This will display the "Add Permission" page. After selecting one or more permissions to add and who to add the selected permissions to, click the "Add" button. The users/groups/roles will now be added to the selected permissions

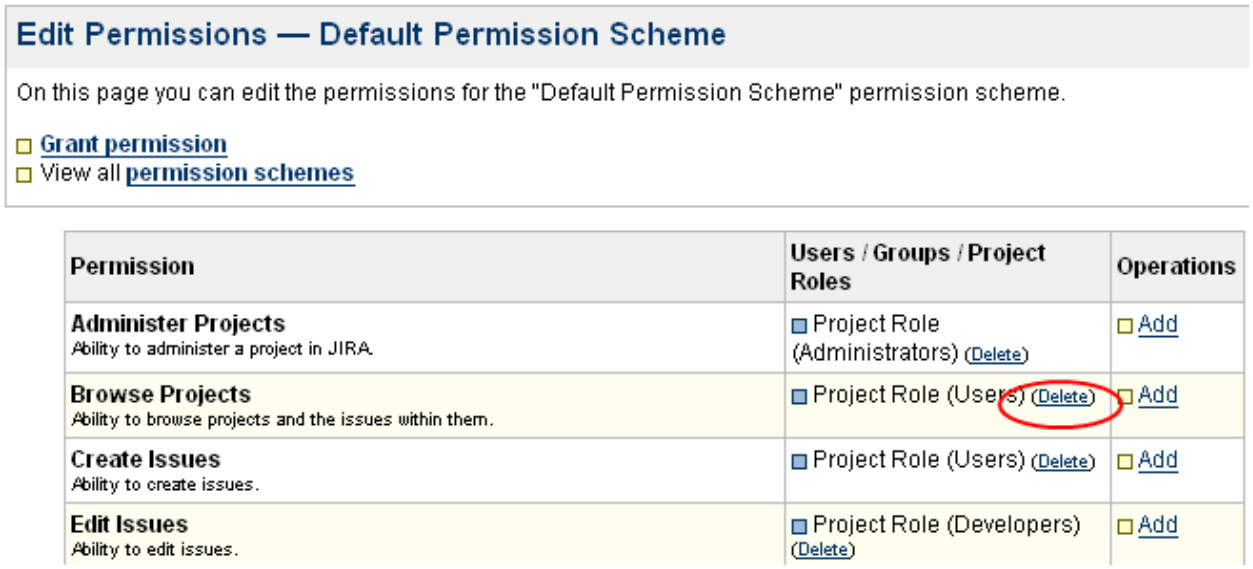
   Note that project roles are useful for defining specific team members for each project. Referencing project roles (rather than users or groups) in your permissions can help you minimise the number of permission schemes in your system.



Add Users to Permission

(Note: this screenshot is from JIRA Enterprise; in JIRA Professional/Standard only 'Group' and 'Project Role' are available)

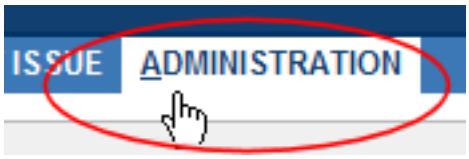
5. Repeat the last 2 steps until all required users/groups/roles have been added to the permissions
6. To delete a user/group/role from a permission click on the "Delete" link in the "Users / Groups / Roles" column



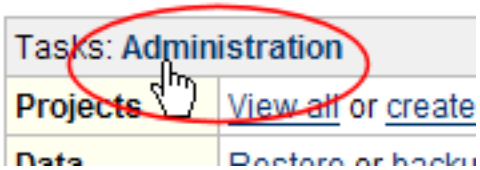
Add Users from Permission

1.6.3.4. Associating a Permission Scheme with a Project

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

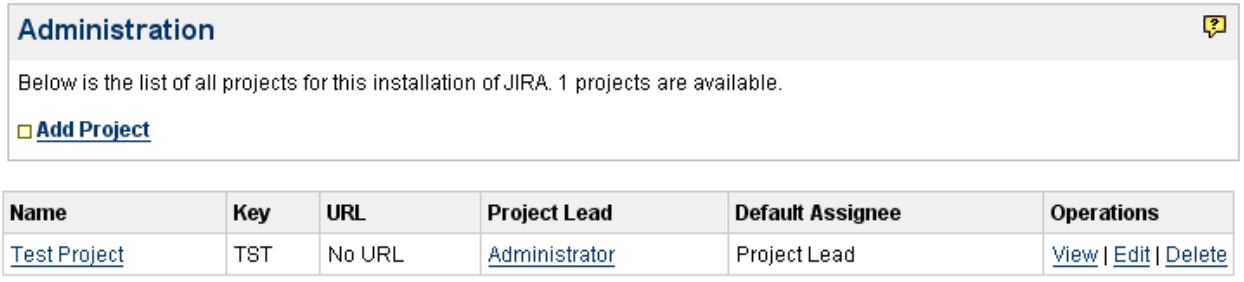


Link to Administration section



Link to Administration section

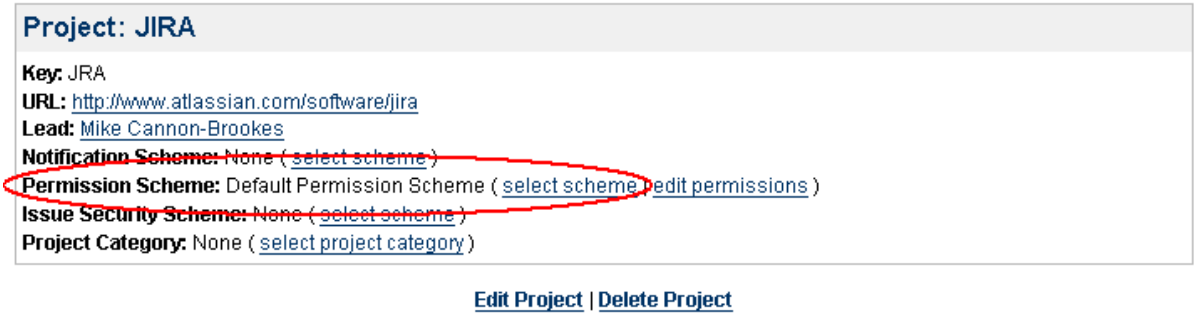
3. A list of projects is displayed



List of Projects

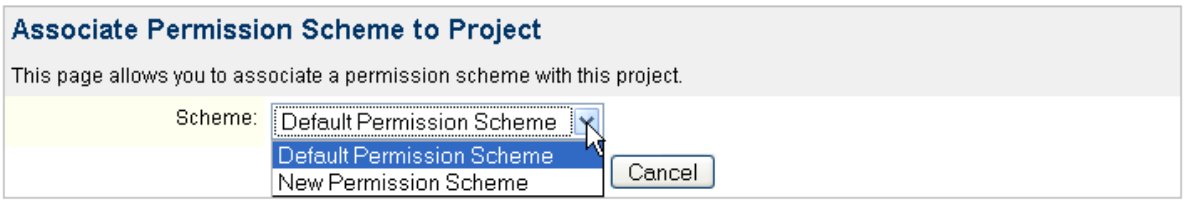
4. Select the project you want by clicking on the project name. This will display the project details

5. Click on the "select scheme" link beside the Permission Scheme caption.



Project Details

6. This will bring up a list of Permission Schemes. Select the Permission Scheme that you want to associate with this project.

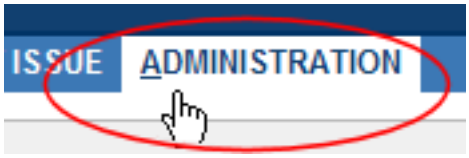


Select Scheme

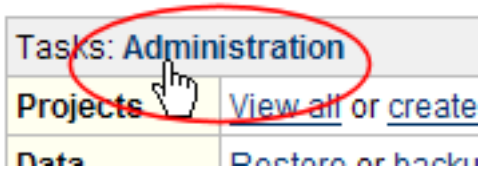
7. Click the "Associate" button to associate the project with the permission scheme.

1.6.3.5. Deleting a Permission Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

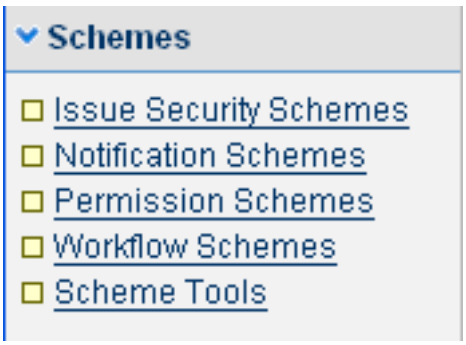


Link to Administration section



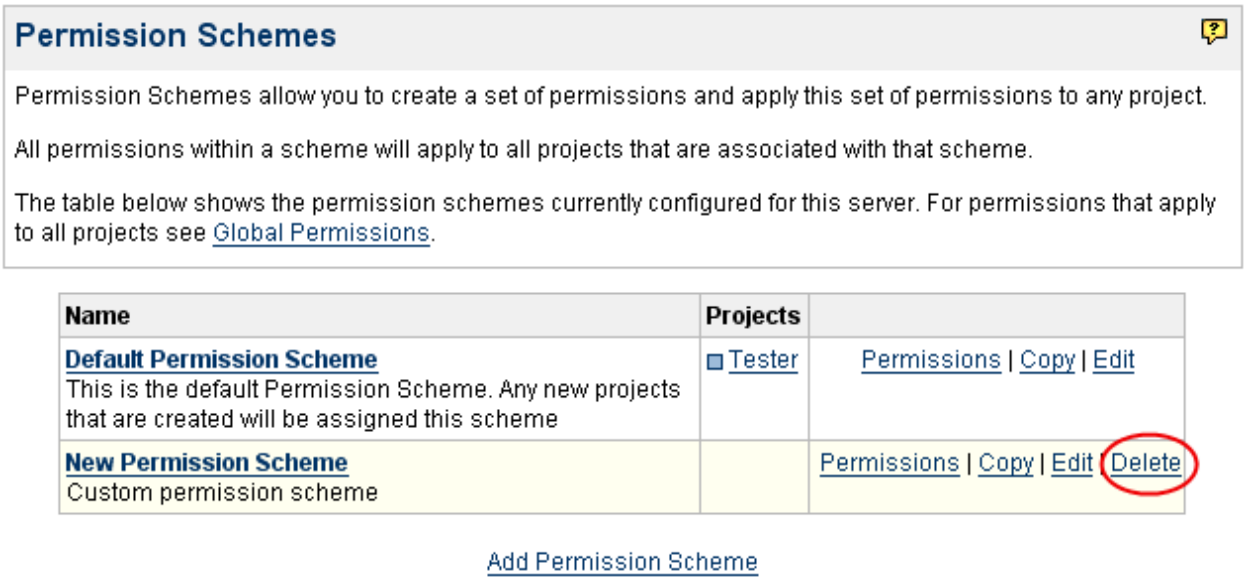
Link to Administration section

3. On the panel on the left, under the title "Features", click on the link labelled "Permission Schemes".



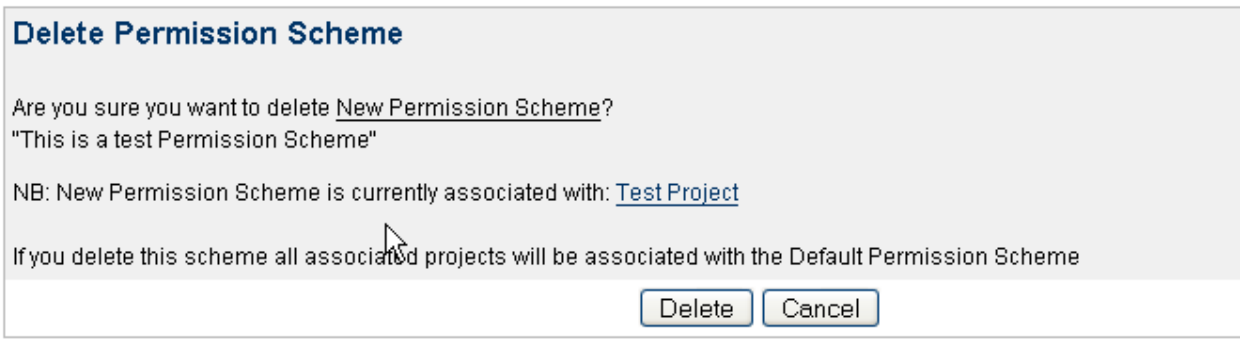
View Permission Schemes

4. This will display the 'Permission Schemes' page. This page lists all the Permission Schemes that are currently defined in your JIRA system. Click on the "Delete" link for the scheme that you want to delete.



Delete Permission Schemes

5. A confirmation screen will appear. To delete click "Delete" otherwise click "Cancel".



Delete Permission Scheme

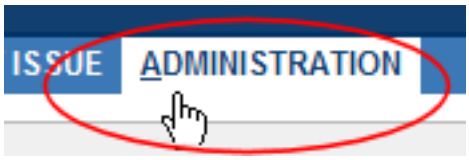
6. The scheme will be deleted and all associated projects will be automatically associated with the Default Permission Scheme. (Note that you cannot delete the Default Permission Scheme.)

Note:

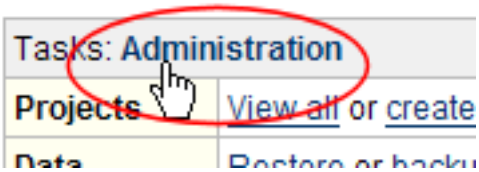
See also Minimising the number of Permission Schemes and Notification Schemes.

1.6.3.6. Copying a Permission Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

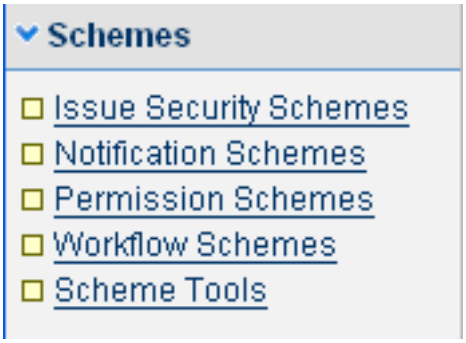


Link to Administration section



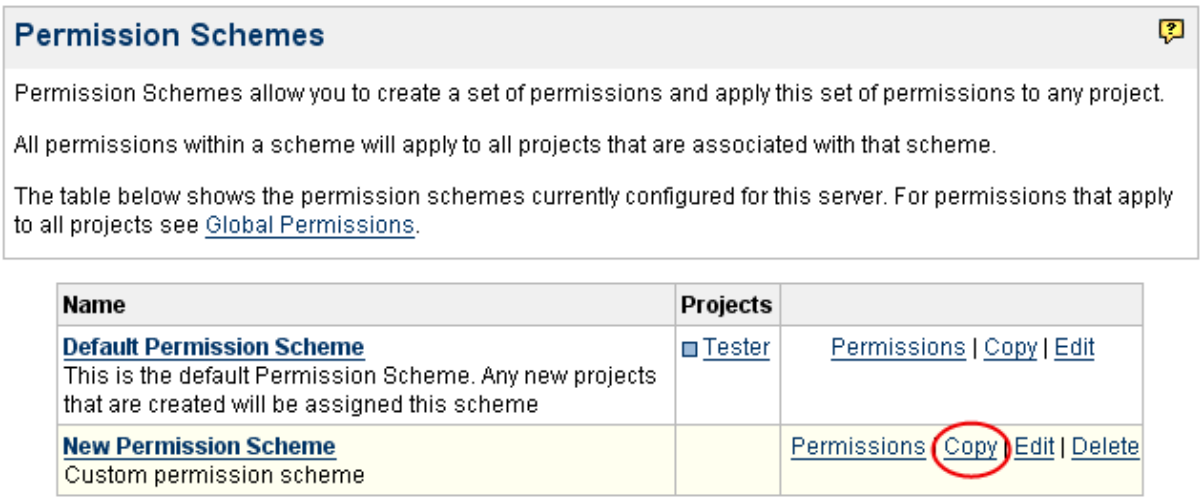
Link to Administration section

3. On the panel on the left, under the title "Features", click on the link labelled "Permission Schemes".



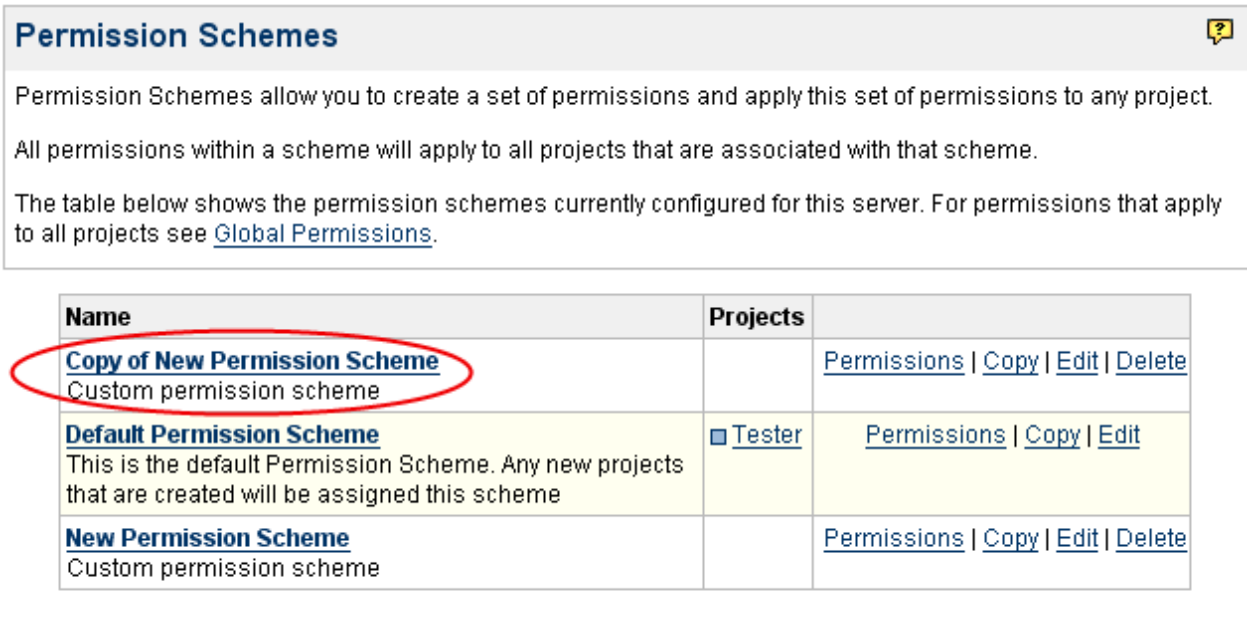
View Permission Schemes

4. This will display the "Permission Schemes" page. This page lists all of the Permission Schemes that JIRA currently has. Click on the "Copy" link for the scheme that you want to copy.



Copy Permission Schemes

5. A new scheme will be created with the same permissions and the same users/groups/roles assigned to them.



Copy Permission Schemes

1.6.3.7. Additional Resources

- Permission scheme overview tutorial video — Watch this short tutorial video to see how to set up a new permission scheme to control which users can perform which operations on an issue. Please note the JIRA version and JIRA edition of the tutorial video before watching.

1.6.4. Issue security

Issue security levels are a JIRA Enterprise feature that allows you to control who can see individual issues within a project (subject to the project's permissions).

An issue security level is a *named collection of users*. Issue security levels are created within issue security schemes, which are then associated with projects. Once an issue security scheme has been associated with a project, its security levels can be applied to issues in that project (note, sub-tasks will inherit the security level of their parent issue). Those issues will then only be accessible to members of that security level.

A security level's members may consist of:

- Individual users
- Groups
- Project roles
- Issue roles such as 'Reporter', 'Project Lead', and 'Current Assignee'
- 'Anyone' (eg. to allow anonymous access)
- A (multi-)user picker custom field.
- A (multi-)group picker custom field. This can either be an actual group picker custom field, or a (multi-)select-list whose values are group names.

Note:

Only users with the project-specific 'Set Issue Security' permission can apply a security level to an issue, regardless of whether they are members of the security level.

Please note that issue security levels are only available in the Enterprise edition of JIRA.

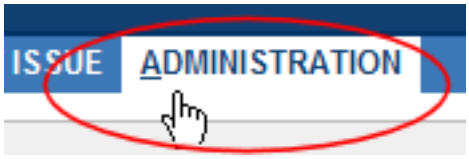
1.6.4.1. Why use issue security?

As an example, a company may have a public instance of JIRA running. Within this instance they may have several projects that external people (customers) can browse. However, it may not be appropriate to show all issues to the customers. To achieve this you could:

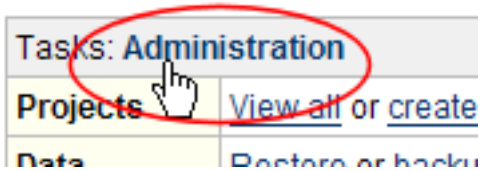
- Create an issue security scheme.
- Create an issue security level named 'Private' for this scheme.
- Add appropriate people to the 'Private' security level.
- Associate the issue security scheme with the relevant projects.
- Set the security level of specific issues to 'Private'.

1.6.4.2. Creating an Issue Security Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

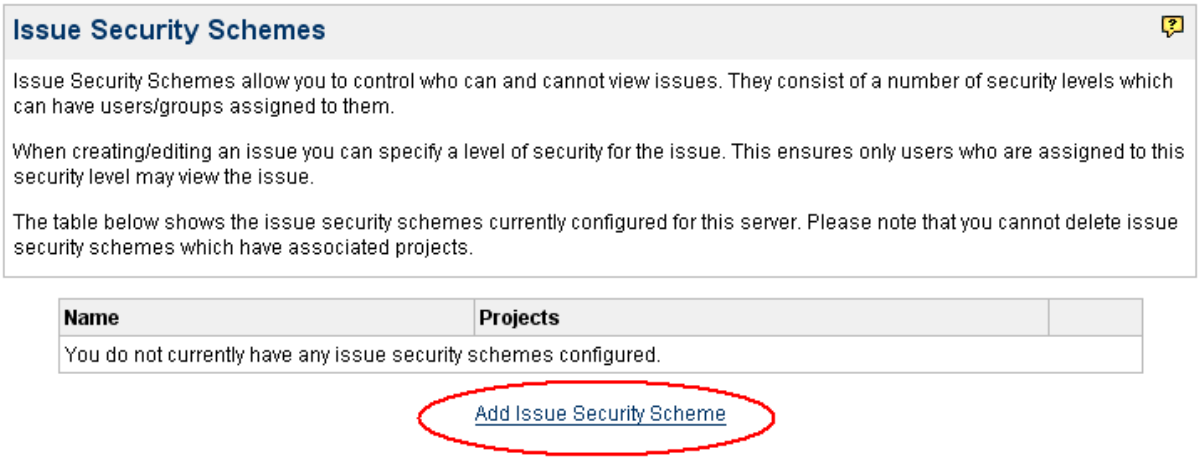


Link to Administration section

3. On the panel on the left, under the title 'Schemes', click the link labelled 'Issue Security Schemes'.

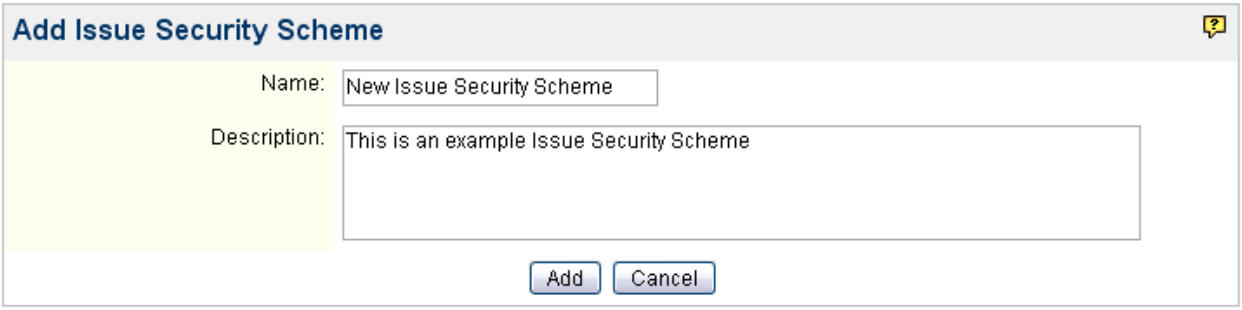
View Issue Security Schemes

4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has. Click the 'Add Issue Security Scheme' link.



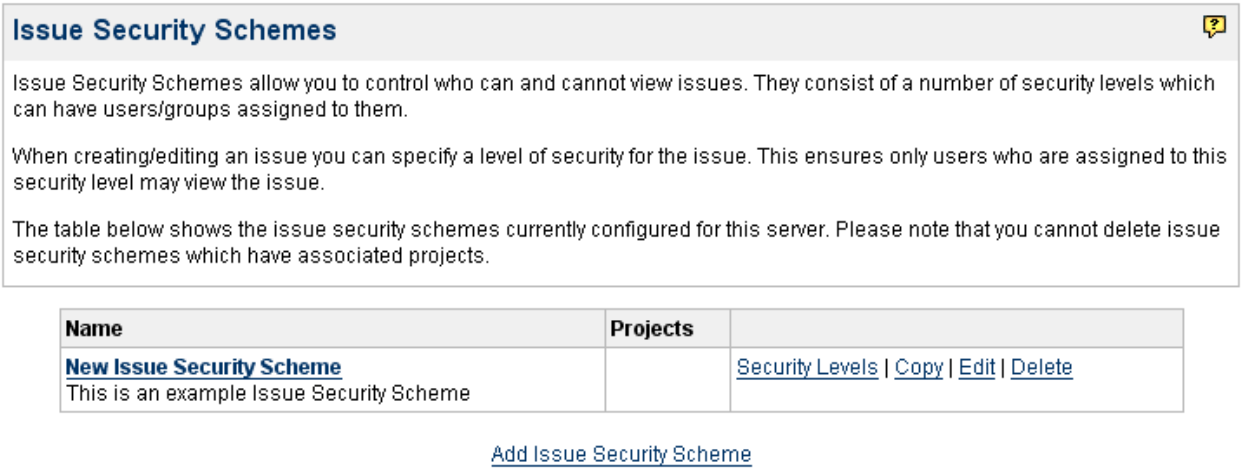
View Issue Security Schemes

5. In the 'Add Issue Security Scheme' form, enter a name for the issue security scheme, and a short description of the scheme. Then click the 'Add' button.



Add an Issue Security Scheme

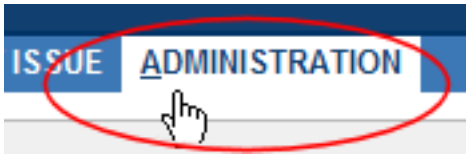
6. You will return to the 'Issue Security Schemes' page, which now contains the newly added scheme.



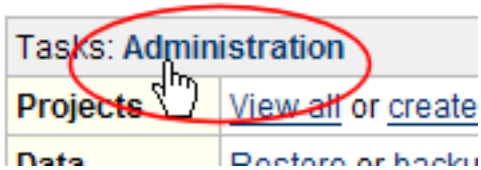
View Issue Security Schemes

1.6.4.3. Adding a Security Level to an Issue Security Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

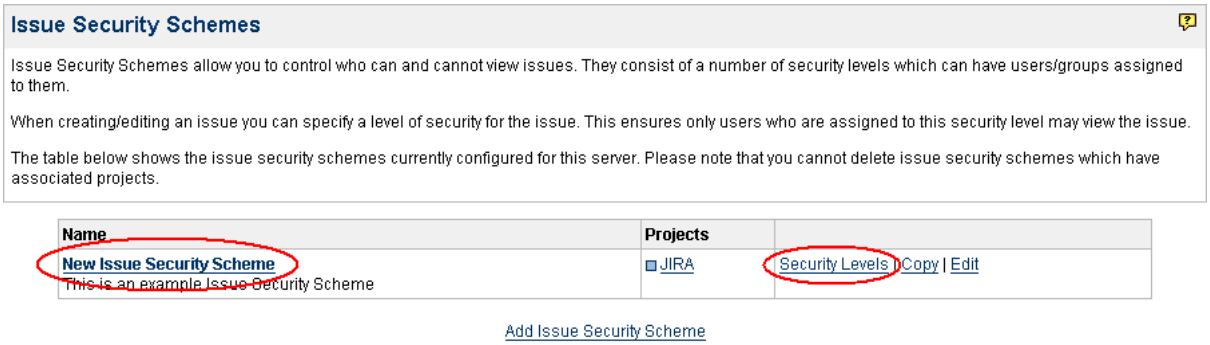


Link to Administration section

3. On the panel on the left, under the title 'Schemes', click on the link labelled 'Issue Security Schemes'.

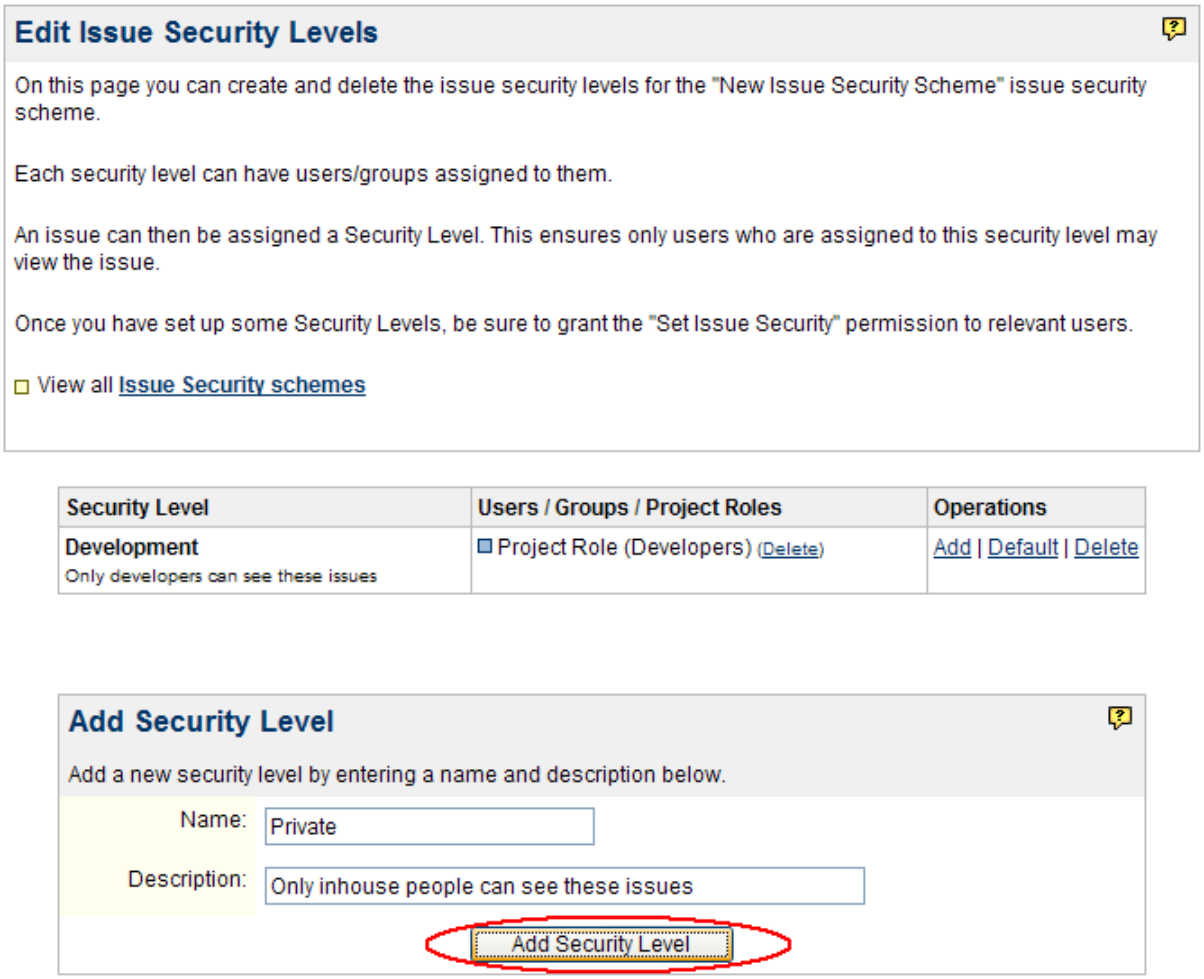
View Issue Security Schemes

4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has.
5. Click on the name of any scheme or the link 'Security Levels' to bring up the 'Edit Issue Security Levels' page.



View Issue Security Levels

6. In the "Add Security Level" box, enter a name and description for your new security level. Then click the button 'Add Security Level'.



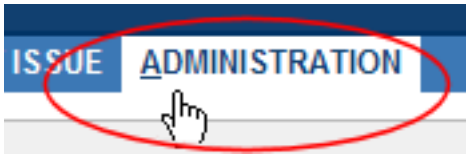
Add Issue Security Levels

1.6.4.4. Setting the Default Security Level for an Issue Security Scheme

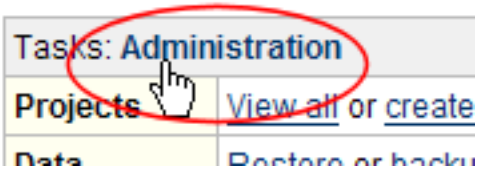
You can choose to specify a Default Security Level for your issue security scheme.

The Default Security Level is used when issues are created. If the reporter of an issue does not have the permission 'Set Issue Security', then the issue's security level will be set to the Default Security Level. If the project's issue security scheme does not have a Default Security Level, then the issue's security level will be set to 'None'. (A security level of 'None' means that anybody can see the issue.)

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

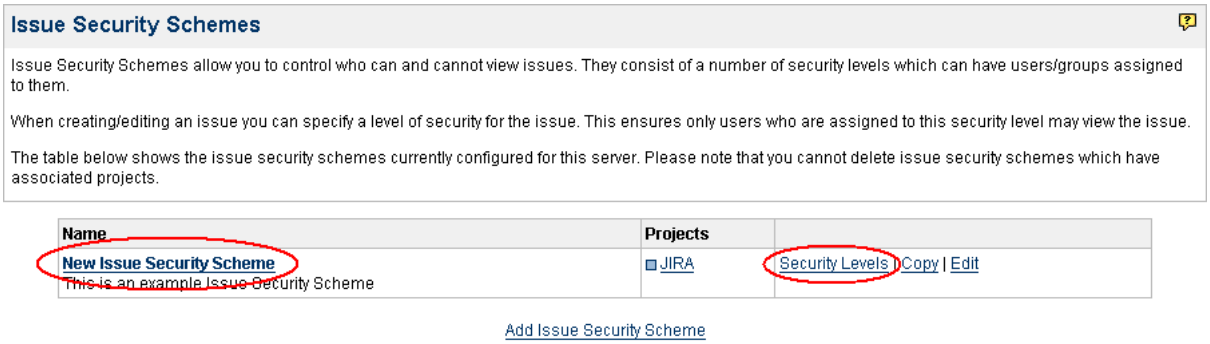


Link to Administration section

3. On the panel on the left, under the title 'Schemes', click on the link labelled 'Issue Security Schemes'.

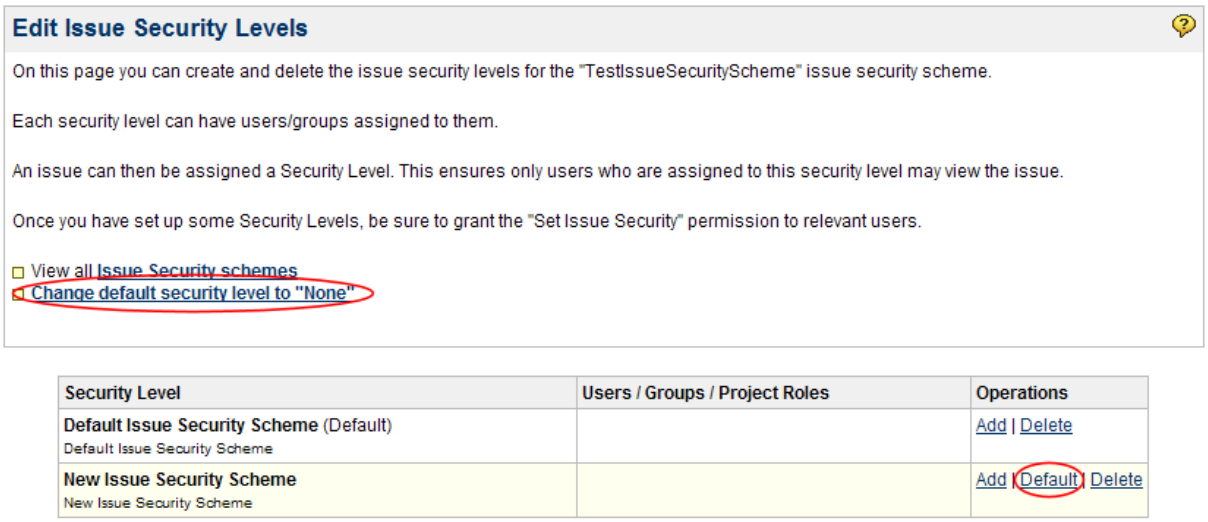
View Issue Security Schemes

4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has.
5. Click on the name of any scheme or the link 'Security Levels' to bring up the 'Edit Issue Security Levels' page.



View Issue Security Levels

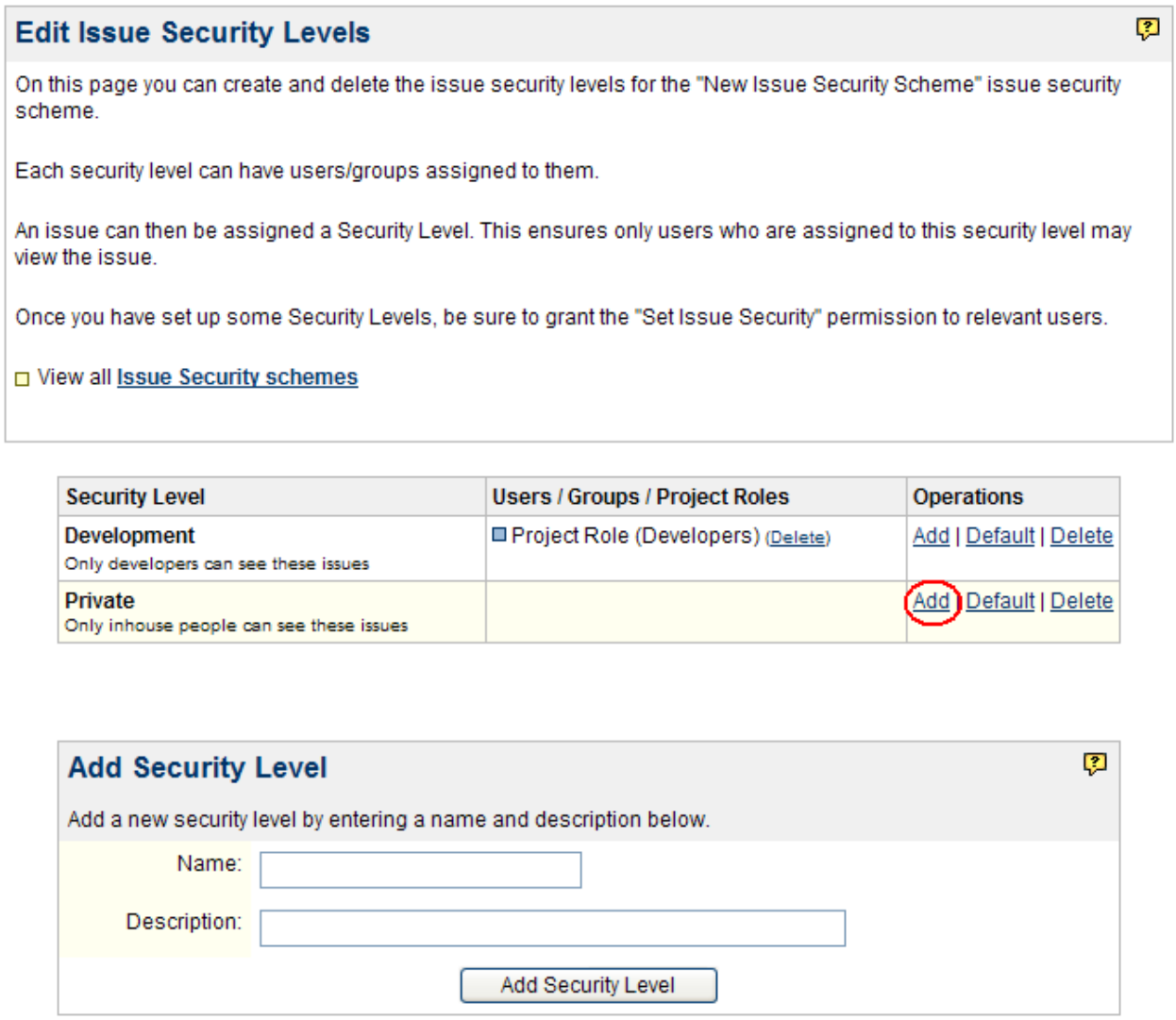
6. The default issue security level for a scheme can be set and removed, as follows:
   - To set the Default Security Level for an issue security scheme, locate the appropriate Security Level and click its 'Default' link.
   - To remove the Default Security Level for an issue security scheme, click the 'Change default security level to "None"' link.



Set the Default Issue Security Level

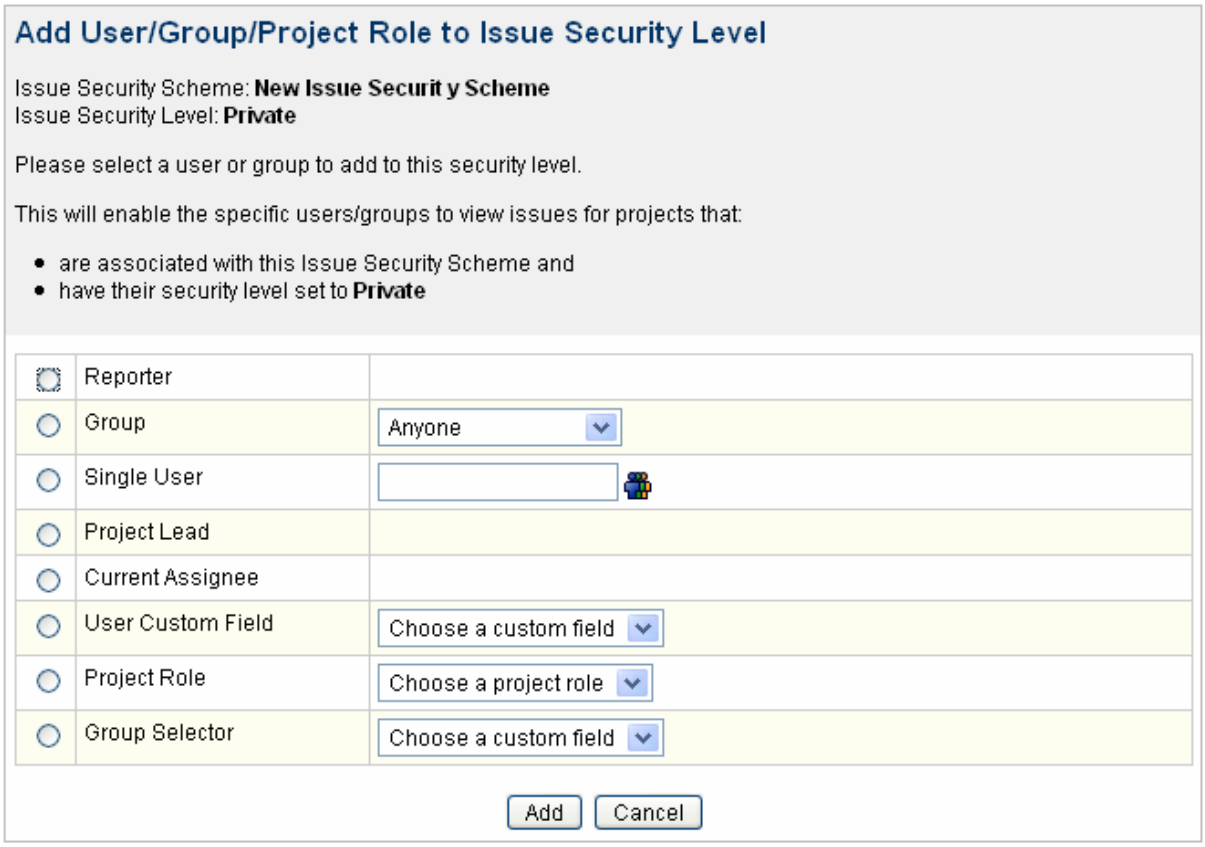
1.6.4.5. Adding Users/Groups/Project Roles to a Security Level

1. Go to the 'Edit Issue Security Levels' page (see above).
2. Locate the appropriate Security Level and click its 'Add' link:



Add Users to Issue Security Levels

3. This will display the "Add Issue Security" page. Select the appropriate user, group or project role, then click the "Add" button.

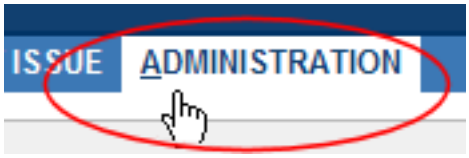


Add Users to Issue Security Levels

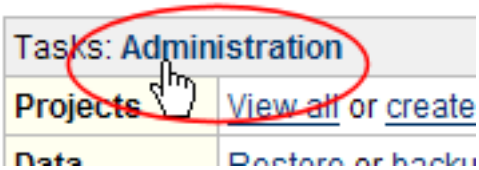
4. Repeat steps 2 and 3 until all appropriate users and/or groups and/or project roles have been added to the security level.

1.6.4.6. Assigning an Issue Security Scheme to a Project

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

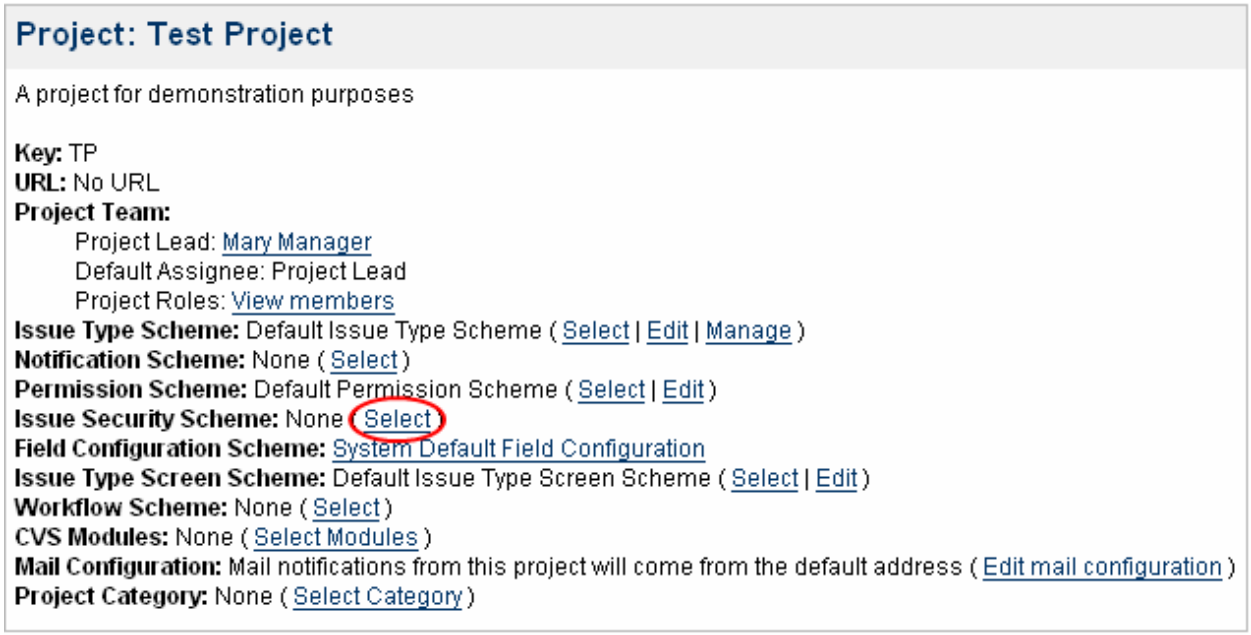


Link to Administration section

3. A list of projects is displayed

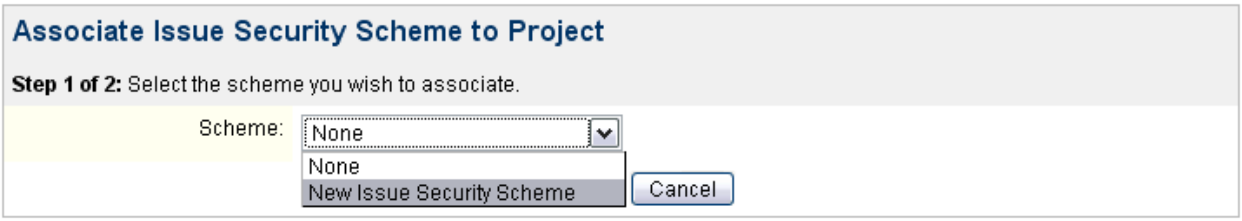
List of Projects

4. Select the project you want by clicking on the project name. This will display the project details
5. Click the 'Select' link beside the 'Issue Security Scheme' caption.



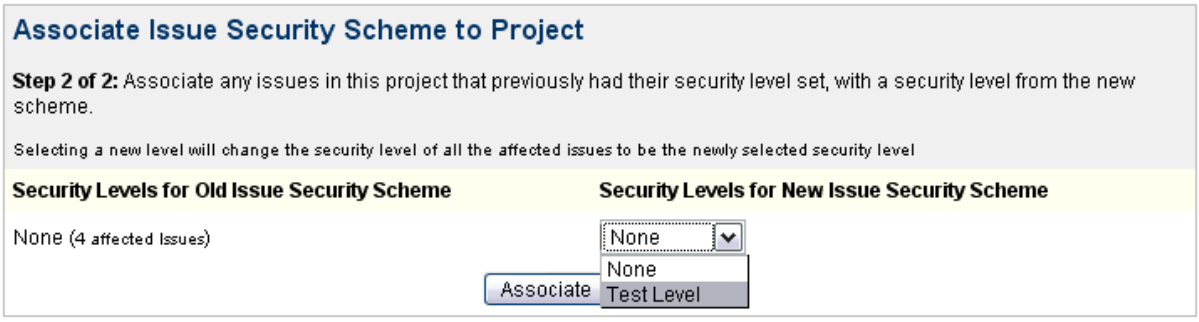
Project Details

6. This will bring up a list of Issue Security Schemes. Select the Issue Security Scheme that you want to associate with this project.



Select Scheme

7. If if there are no previously secured issues (or if the project didn't previously have an issue security scheme), skip the next step.
8. If there are any previously secured issues, select a new security level to replace each old level. All issues with the security level from the old scheme will now have the security level from the new scheme. You can choose 'None' if you want the security to be removed from all previously secured issues.

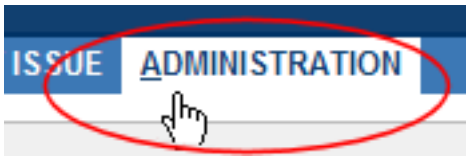


Select New Level

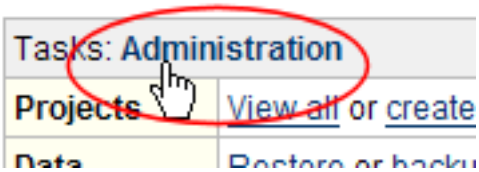
9. Click the 'Associate' button to associate the project with the issue security scheme.

1.6.4.7. Deleting an Issue Security Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

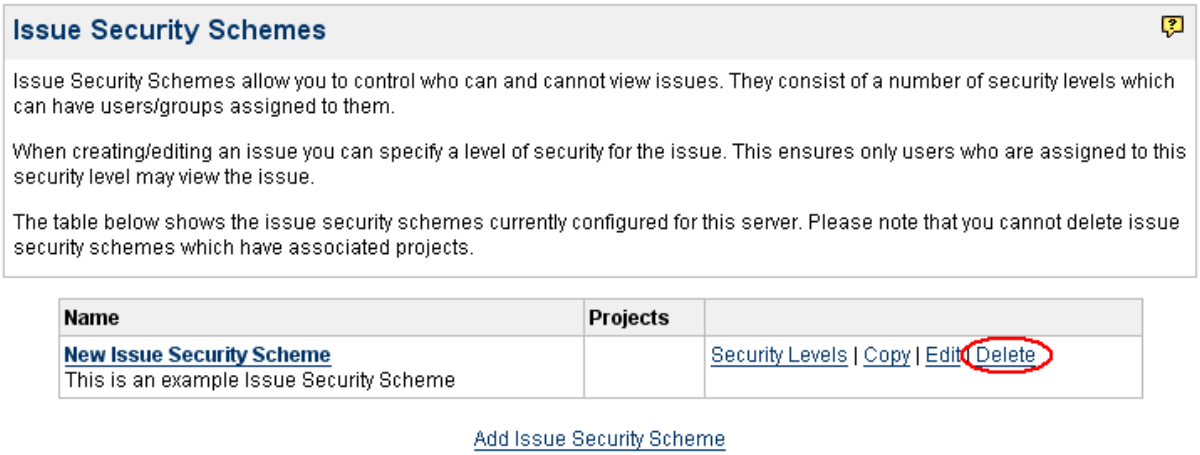
3. On the panel on the left, under the title 'Schemes', click on the link labelled 'Issue Security Schemes'.

View Issue Security Schemes

4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has. Click the 'Delete' link for the scheme that you want to delete.

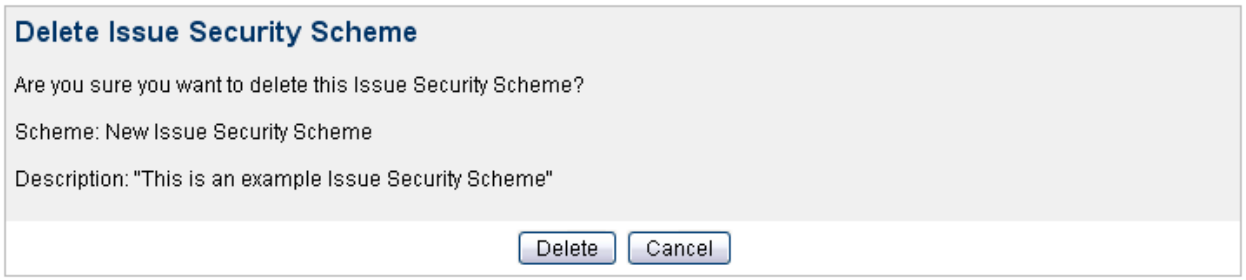
Note:

You cannot delete a Issue Security Scheme if it is associated with a project; you must first unassign the scheme. To unassign a scheme, please refer to Assigning an Issue Security Scheme.



Delete Issue Security Schemes

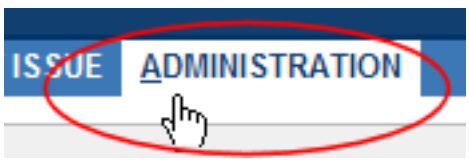
5. A confirmation screen will appear. To delete, click 'Delete'; otherwise click 'Cancel'.



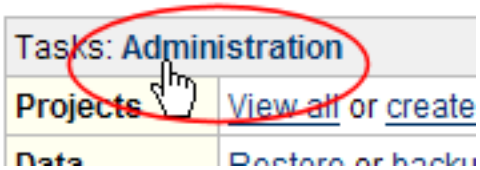
Delete Issue Security Scheme

1.6.4.8. Copying an Issue Security Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

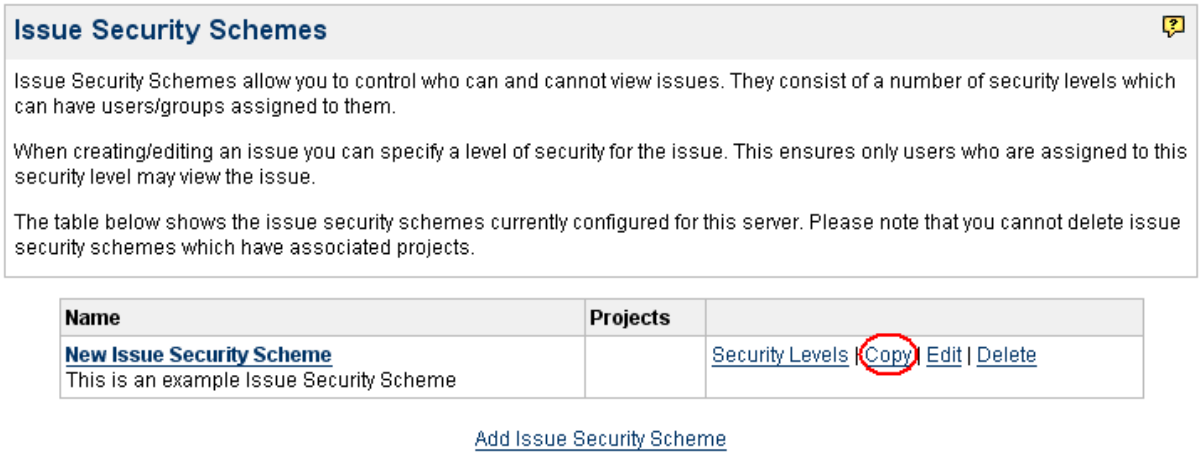


Link to Administration section

3. On the panel on the left, under the title 'Schemes', click on the link labelled 'Issue Security Schemes'.

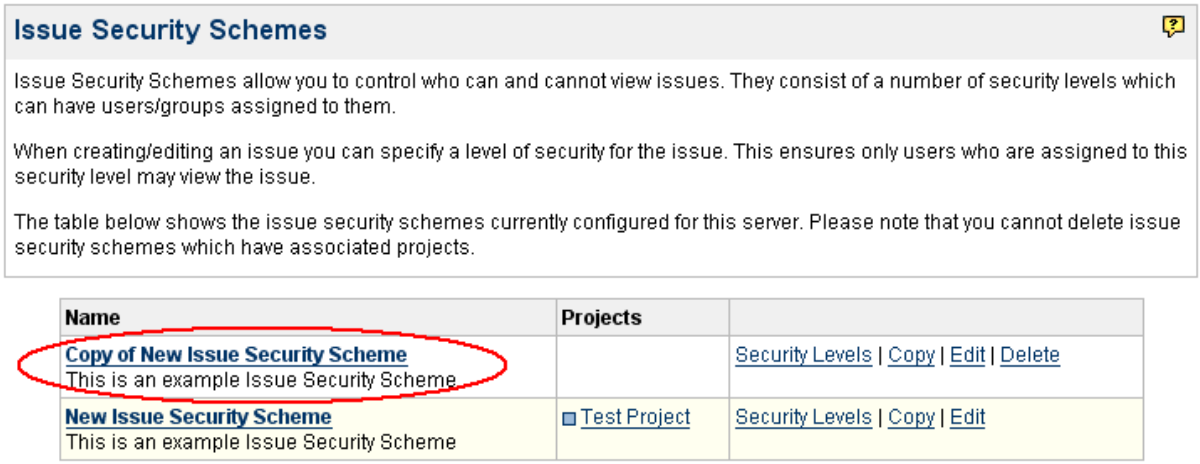
View Issue Security Schemes

4. This will display the 'Issue Security Schemes' page. This page lists all of the Issue Security Schemes that JIRA currently has. Click the 'Copy' link for the scheme that you want to copy.



Copy Issue Security Schemes

5. A new scheme will be created with the same security levels and the same users/groups/project roles assigned to them.



Copy Issue Security Schemes

1.6.4.9. Additional Resources

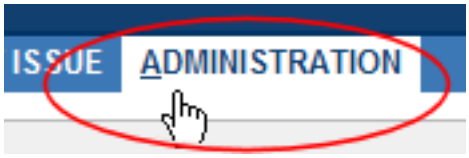
- Issue security scheme overview tutorial video — Watch this short tutorial video to see how to use an issue security scheme to restrict the viewing of issues to specified users. Please note the JIRA version and JIRA edition of the tutorial video before watching.

1.7. Project Management

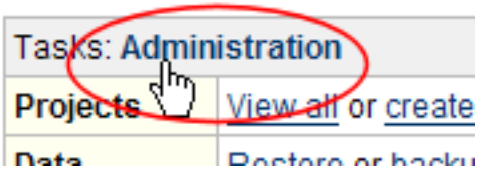
1.7.1. Defining a project

To configure a project in JIRA:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



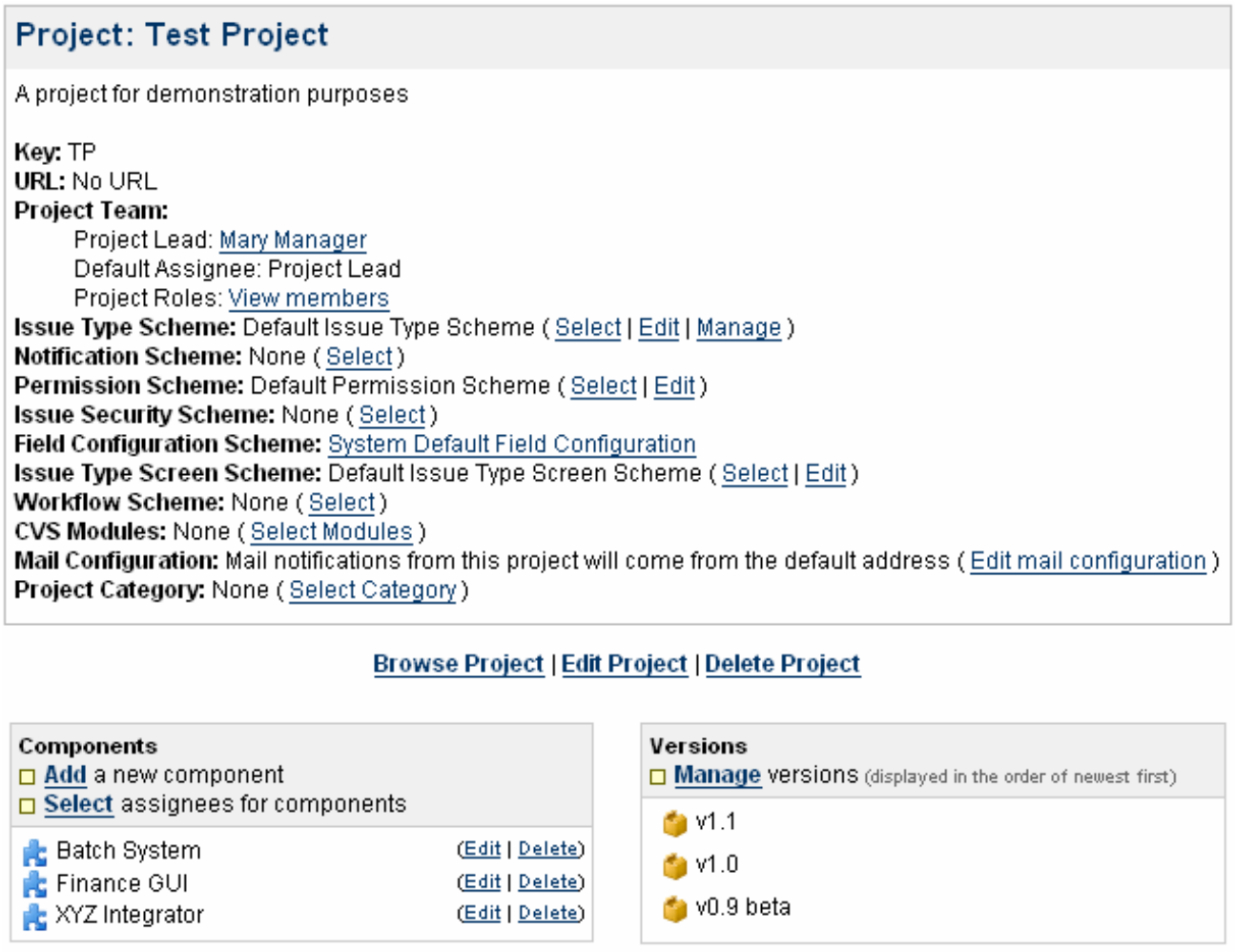
Link to Administration section



Link to Administration section

3. Select an existing project, or click **Add Project** to add a project.

Here is what a project looks like once created:



Project admin page

A project's configuration settings are as follows:

- **Key** - a 'key' unique to this project (eg. 'WEB'), which specifies the first few letters of this project's issue keys (eg. 'WEB-100'). We recommend that you define a key that describes the project and is easy to type (as it prefixes each issue in the project). Please note that the key is shown to users who do not have permission to see the project and cannot be changed once the project exists.
- **URL** - An optional URL associated with this project, eg. pointing to project documentation.
- **Project Team:**
  - **Project Lead** - user fulfilling the role of project leader. Used as the 'Default Assignee' (see below), and potentially elsewhere in JIRA (eg. in permission schemes, notification schemes, issue security schemes and workflows).
  - **Default Assignee** - the user to whom issues in this project are initially assigned when created. Can be either the 'Project Lead' (above), or, if **Allow unassigned issues** is set to 'On' in JIRA's general configuration, 'Unassigned'. Note: in JIRA Enterprise there are also default component assignees.
  - **Project Roles** - members are users/groups who fulfil particular functions for this project. Project roles are used in permission schemes, notification schemes, issue security schemes and workflows.
- **Issue Type Scheme** (Enterprise and Professional only) - the project's issue type scheme determines which issue types apply to this project.
- **Notification Scheme** - the project's notification scheme determines who receives email notifications of changes to issues in this project.
- **Permission Scheme** - the project's permission scheme determines who has permission to view or change issues in this project.
- **Issue Security Scheme** (Enterprise only) - the project's issue security scheme determines what visibility levels issues in this project can have (see issue-level security).
- **Field Configuration Scheme** (Enterprise only) - the project's field configuration scheme determines which field configuration applies to issue types in this project. (A field configuration determines fields' overall visibility, requiredness, formatting (wiki/rich-text or plain) and existence on various screens).
- **Screen Scheme** (Professional only) - the project's screen scheme determines which screens are displayed for different issue operations (view, edit, create); **or:**
- **Issue Type Screen Scheme** (Enterprise only)- the project's issue type screen scheme determines which screens are displayed for different issue operations (view, edit, create), for different issue types.
- **Workflow Scheme** (Enterprise only) - the project's workflow scheme determines which workflows (issue state transitions) apply to issue types in this project.
- **CVS Modules** - configures CVS integration for this project.
- **Mail Configuration** - specifies the 'From' address for emails sent from this project. Only available if an SMTP email server has been configured in JIRA.
- **Project Category** (Enterprise only) - a logical category to group this project into. Useful for managing multiple related projects. New categories can be created via the 'Project Categories' link in the 'Administration' menu.

As well as:

- **Components** - logical groups that this project's issues can belong to. See the component management page for details.
- **Versions** - versions defined in the project. See the version management page for details.

1.7.1.1. A note about Project Administrators

A JIRA project administrator is someone who has the project-specific 'Administer Project'

permission, but not necessarily the global 'JIRA Administrator' permission.

A project administrator can:

- Edit the project name ('Test Project' in the screenshot above)
- Edit the project description ('A project for demonstration purposes' in the screenshot above)
- Edit the URL
- Edit the Project Lead
- Edit project role membership*
- Define project components
- Define project versions

*Enterprise edition only. In Professional and Standard editions, only a global administrator can edit project role membership.

1.7.1.2. Additional Resources

- Adding a Project tutorial video — Watch this short tutorial video to see how a project is added in JIRA. Please note the JIRA version and JIRA edition of the tutorial video before watching.

1.7.2. Managing project role membership

A JIRA project role is a flexible way to associate users and/or groups with a particular project.

Unlike groups, which have the same membership throughout JIRA, project roles have specific members for each project. Users may play different roles in different projects.

Note:

This page contains instructions for managing membership of existing project roles. For information on creating and using project roles, please see Managing project roles.

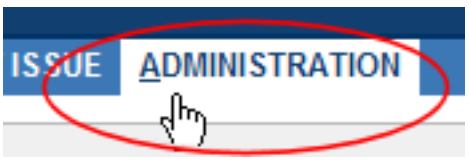
1.7.2.1. Viewing project role members

To see which users and groups belong to each project role for a particular project:

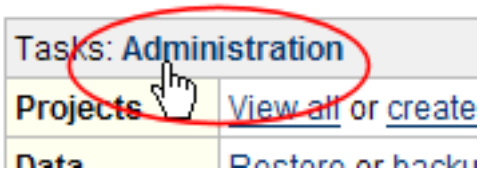
1. The first step depends on which edition of JIRA you are using:
   - *If you are using JIRA Enterprise edition,*
     1. Login to JIRA as a project administrator. (A project administrator is someone who has the project-specific permission '**Administer Project**', but not necessarily the global permission '**JIRA Administrators**'.)
     2. Click the '**Administration**' link on the top bar:

Click Administration link

   - *Or, if you are using JIRA Professional or Standard edition,*
     1. Log in as a user with the '**JIRA Administrators**' global permission.
     2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

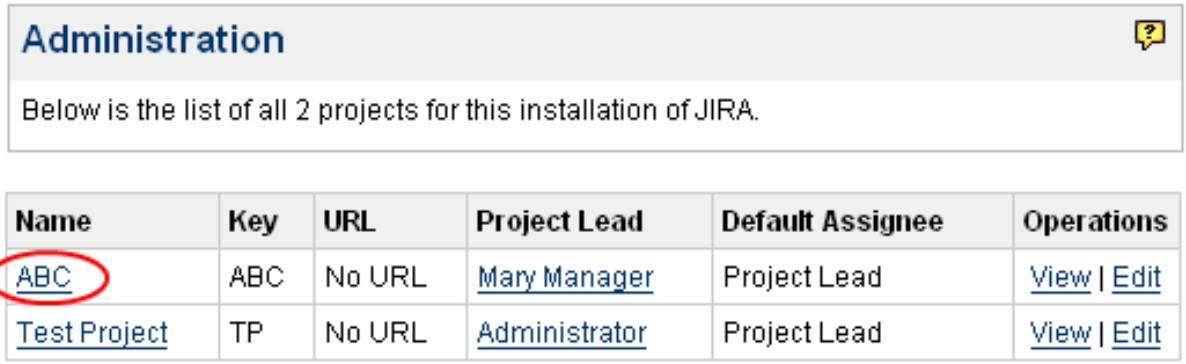


Link to Administration section



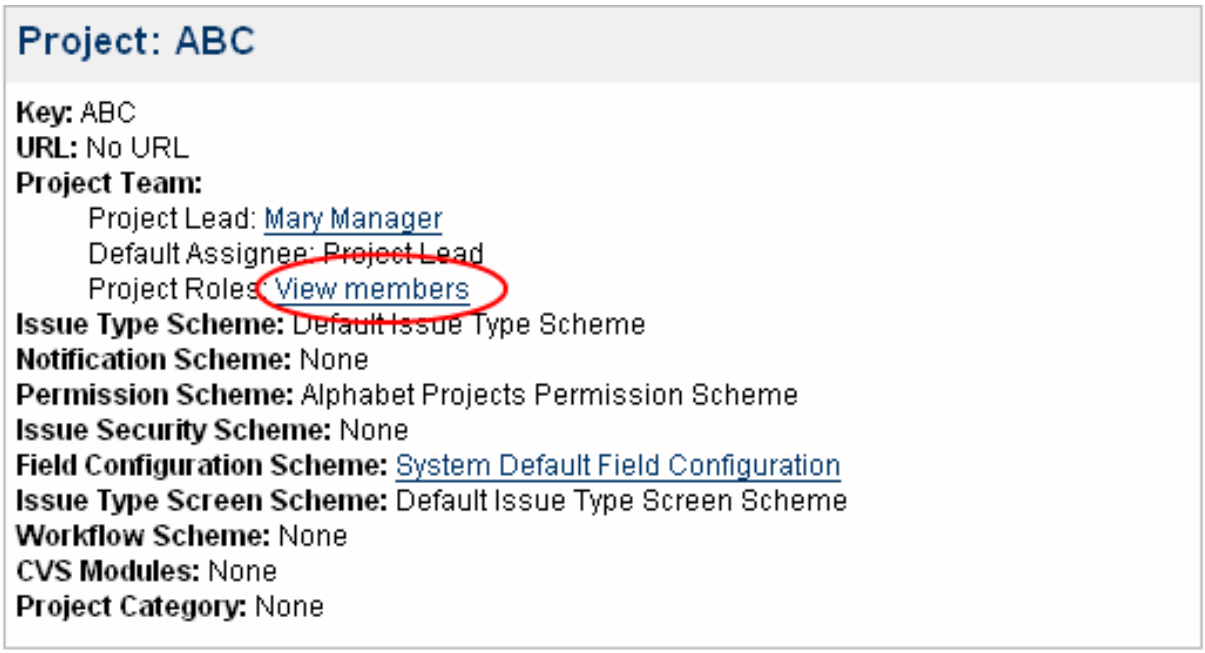
Link to Administration section

2. This will display the **Administration** page, showing a list of projects which you have permission to manage. Click the project of interest.



Select project

3. This will display the **Project Administration** page. Click the 'View members' link:



View project roles

4. This will display the **Manage Project Role Membership** page, showing the project role members for this project:

Manage Project Role Membership for Project: ABC

On this page you can manage project role membership for the ABC project.

Role	Users	Groups
Administrators A project role that represents administrators in a project	Mary Manager Edit	jira-administrators Edit
Developers A project role that represents developers in a project	*None selected.* Edit	jira-developers Edit
Users A project role that represents users in a project	*None selected.* Edit	jira-users Edit

View project roles

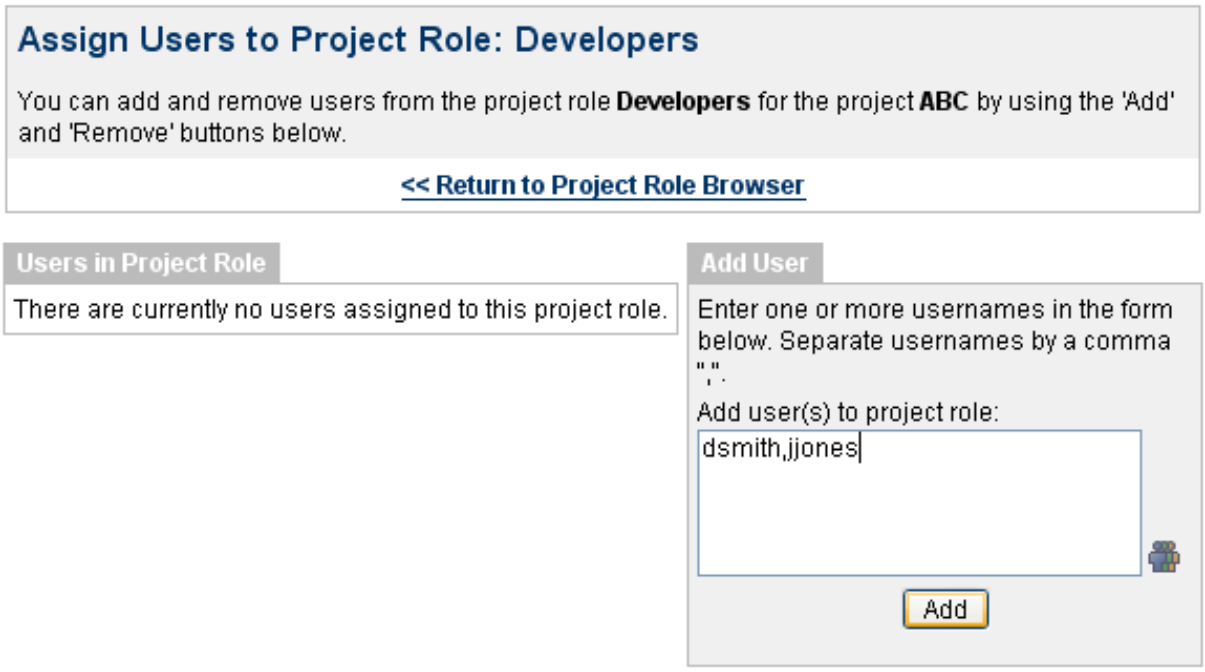
From this page you can assign users/groups to and remove them from project roles, as described below.

Note:

*A project administrator is someone who has the project-specific 'Administer Project' permission, but not necessarily the global 'JIRA Administrator' permission.

1.7.2.2. Assigning a user to a project role

1. Open the **Manage Project Role Membership** page as described in 'Viewing project role members' (above).
2. Click the **Edit** link in the **Users** column for the project role you wish to edit.
3. This will display the **Assign Users to Project Role** page:



Assign users to a project role

The users currently in the project role are listed on the left-hand side of the page.

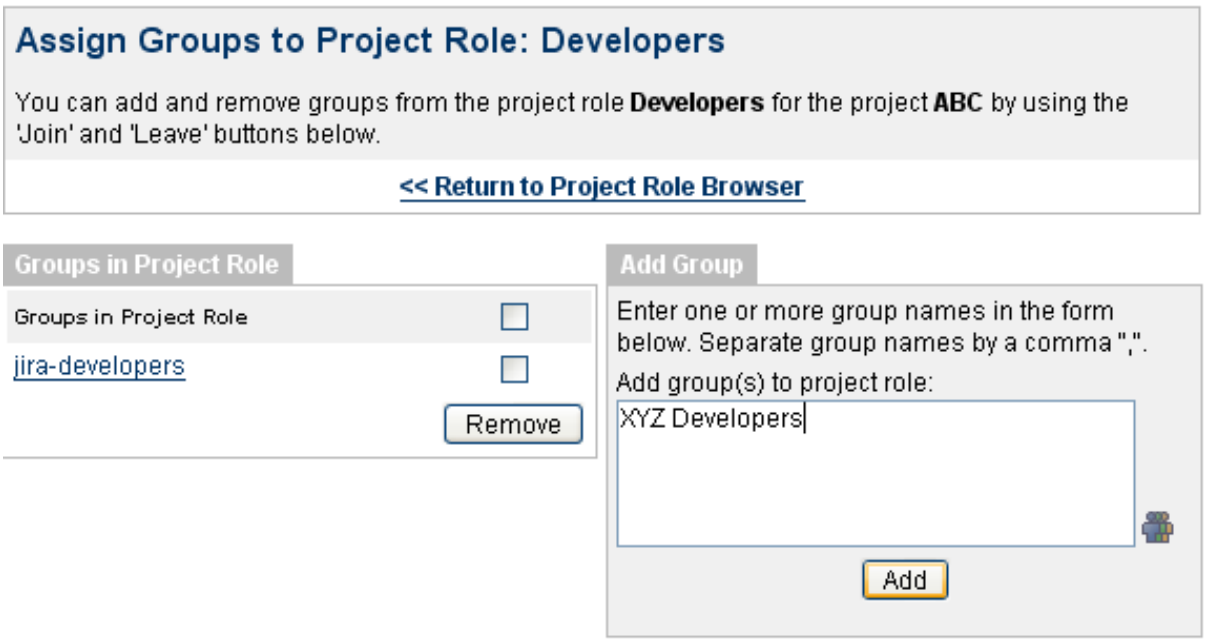
Type the username(s) in the 'Add User' box on the right-hand side of the page, then click the **Add** button.

1.7.2.3. Removing a user from a project role

1. Open the **Manage Project Role Membership** page as described in 'Viewing project role members' (above).
2. Click the **Edit** link in the **Users** column for the project role you wish to edit.
3. This will display the **Assign Users to Project Role** page. The users currently in the project role are listed on the left-hand side of the page. Tick the user(s) you wish to remove from the project role, then click the **Remove** button.

1.7.2.4. Assigning a group to a project role

1. Open the **Manage Project Role Membership** page as described in 'Viewing project role members' (above).
2. Click the **Edit** link in the **Groups** column for the project role you wish to edit.
3. This will display the **Assign Groups to Project Role** page:



Assign groups to a project role

The groups currently in the project role are listed on the left-hand side of the page.

Type the group name(s) in the 'Add Group' box on the right-hand side of the page, then click the **Add** button.

Note:

Group membership can only be viewed/edited by people with the global 'JIRA Administrator' permission. Project administrators may therefore prefer to assign users, rather than groups, to their project roles.

1.7.2.5. Removing a group from a project role

1. Open the **Manage Project Role Membership** page as described in 'Viewing project role members' (above).
2. Click the **Edit** link in the **Groups** column for the project role you wish to edit.
3. This will display the **Assign Groups to Project Role** page. The groups currently in the project role are listed on the left-hand side of the page. Tick the group(s) you wish to remove from the project role, then click the **Remove** button.

Note:

A project role need not have any user or group assigned to it, although project administrators should be careful with this. Depending on how a project role is utilised (e.g. if the project's permission scheme is using project roles), it is possible that not having anyone in a particular project role could make some project activities unavailable.

1.7.3. Component Management

Components are sub-sections of a project. They are used to group issues within a project into smaller parts. The available operations for components are:

- Add - make new components under which issues can be classed.
- Delete - remove a component from a project.
- Edit - Update/change the details of a particular component.
- Select Default Assignee - In **Enterprise** edition you can set the default assignee for issues created in a particular component.

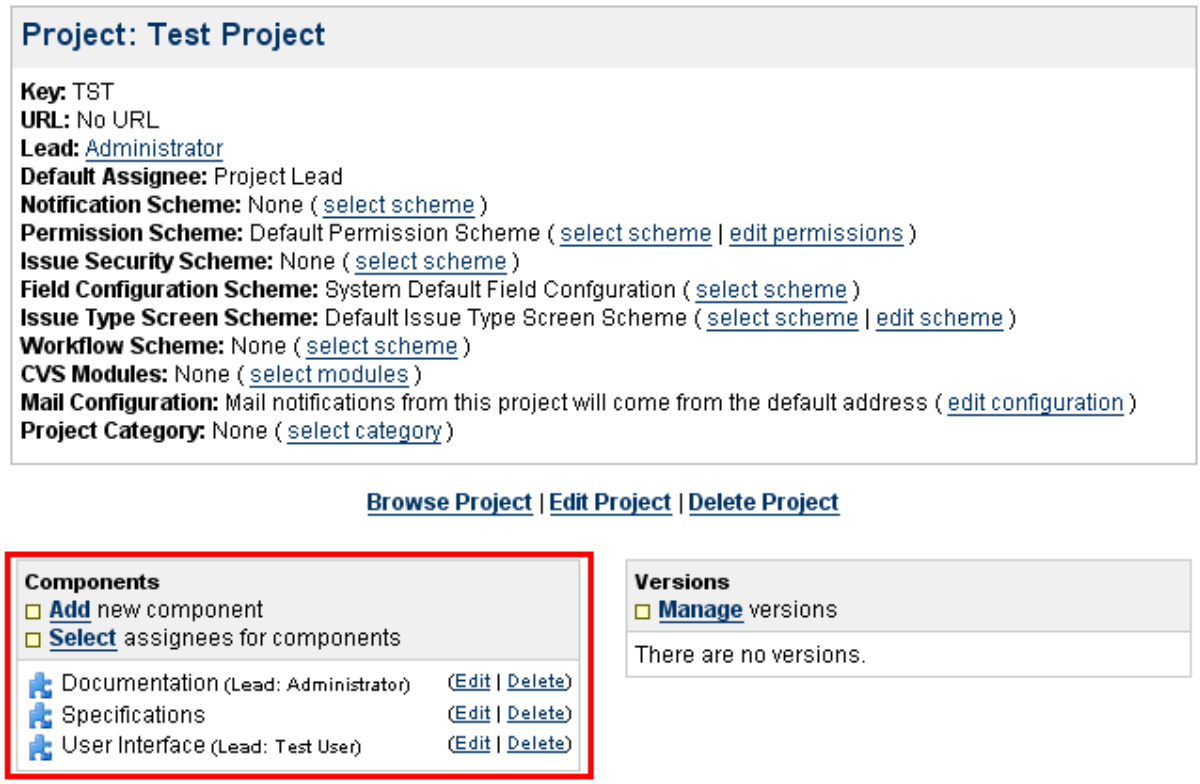
1.7.3.1. Managing project components

All component management operations are available from the Component section of the Project Admin Page.

1. Login to JIRA as a project administrator. (A project administrator is someone who has the project-specific permission '**Administer Project**', but not necessarily the global permission '**JIRA Administrators**'.)
2. Click the '**Administration**' link on the top bar:

Click Administration link

3. This will display the **Administration** page, showing a list of projects which you have permission to manage. Click the project of interest.
4. You will now see a page displaying the project details. On the lower left, the '**Components**' section displays a summary of the project's components along with links to add, edit and remove components (as described below).



Project configuration, with Component Summary highlighted

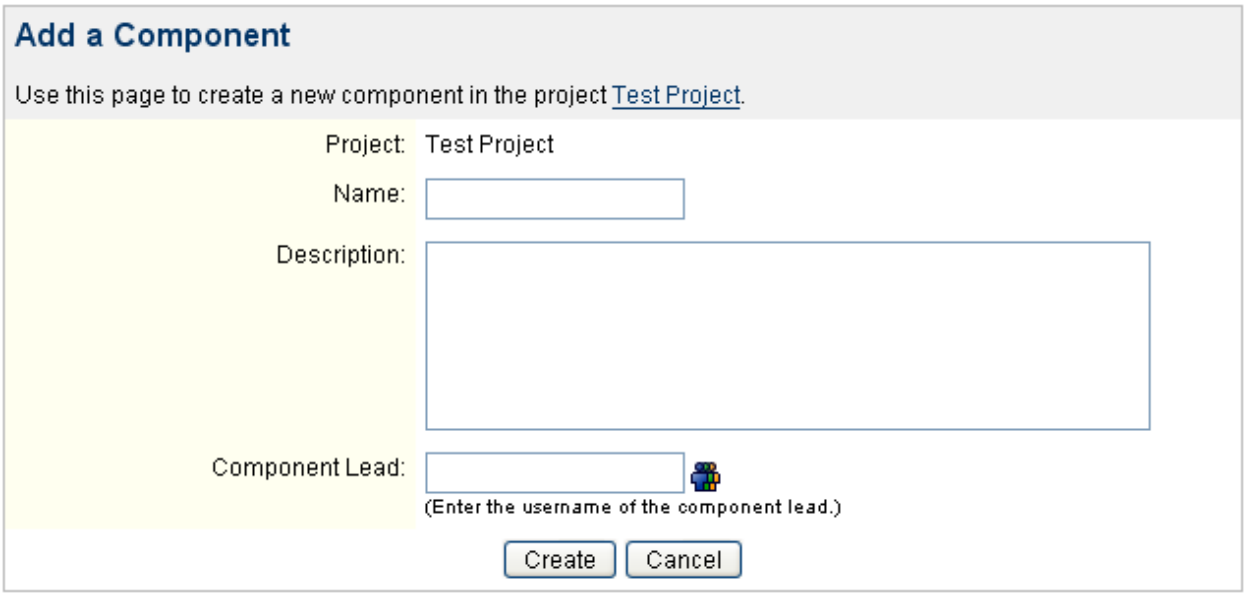
Note:

If you have created a new project and have not assigned a permission scheme with it on creation, then you will not see the above display. Instead, the '**Components**' section will say "There are no components at the moment".

1.7.3.2. Adding a new component

1. In the '**Components**' section (see above), click the 'Add' link.
2. The 'Add a Component' screen will be displayed. Type the name of your new component.

3. You can also optionally enter the component description and/or assign a user to be the component lead.
4. Click the "Add" button. The component summary list is updated immediately and you will be redirected to the project admin page.

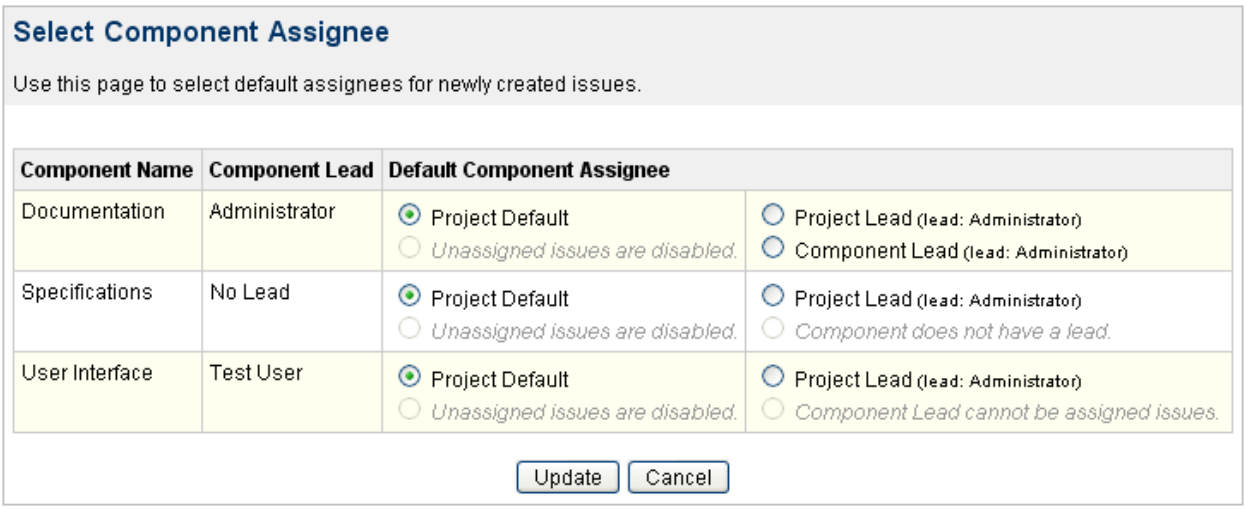


Add Component

1.7.3.3. Selecting a Default Assignee

In **JIRA Enterprise edition** it is possible to extend the default assignee of an issue to be component specific instead of project specific.

1. In the '**Components**' section (see 'Managing project components' above), click the 'Select' link. A 'Select Component Assignee' pane will appear on the Project Admin Page.
2. For each component, select the assignee to whom you want to have new issues in the component assigned by default. See the list of options below.
3. Click on the 'Update' button.



Select Default Assignees

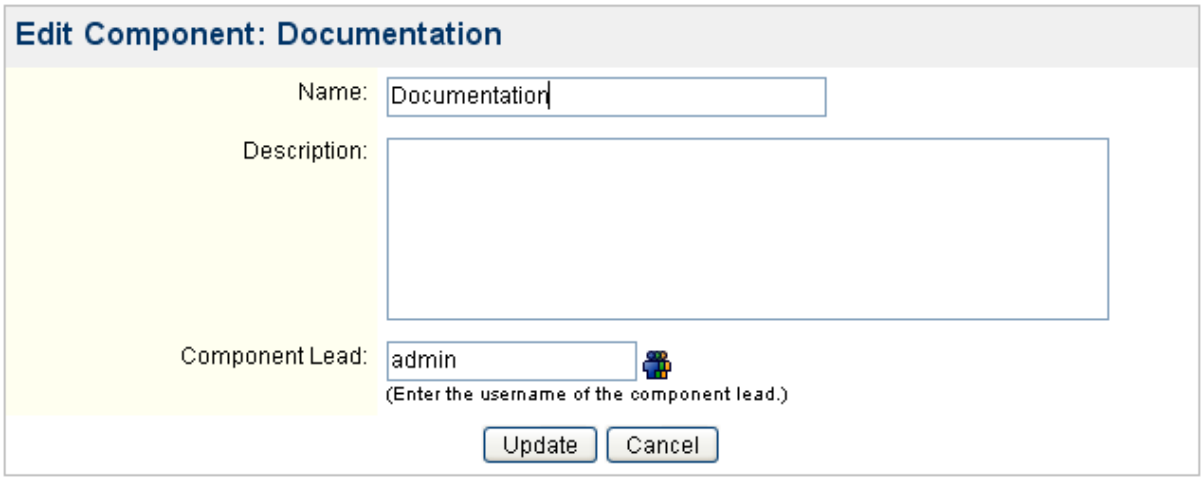
Note:
In the event that the default assignees of components clash, the assignee will be set to the default assignee of the component that is first alphabetically.

Default Assignee Options

Option	Description
Project Default	Issues matching this component will have the assignee set to the same default assignee as the parent project.
Project Lead	The assignee will be set to the project leader. **Condition:** If the project leader is not permitted to be assigned to issues in the permission scheme this option will be disabled and will say "Project Lead is not allowed to be assigned issues.".
Component Lead	The assignee will be set to the component leader. **Condition:** If the project leader is not permitted to be assigned to issues in the permission scheme this option will be disabled and will say "Component Lead is not allowed to be assigned issues". The Component Lead option will also not be available if the component does not have a lead assigned to the component. Instead under this option it will say "Component does not have a lead.".
Unassigned	The assignee of the issue will not be set on the creation of this issue. **Condition:** The unassigned option will only be available if the unassigned is enabled in the General Configuration. Instead under this option it will say "Unassigned issues are disabled.".

1.7.3.4. Editing a component

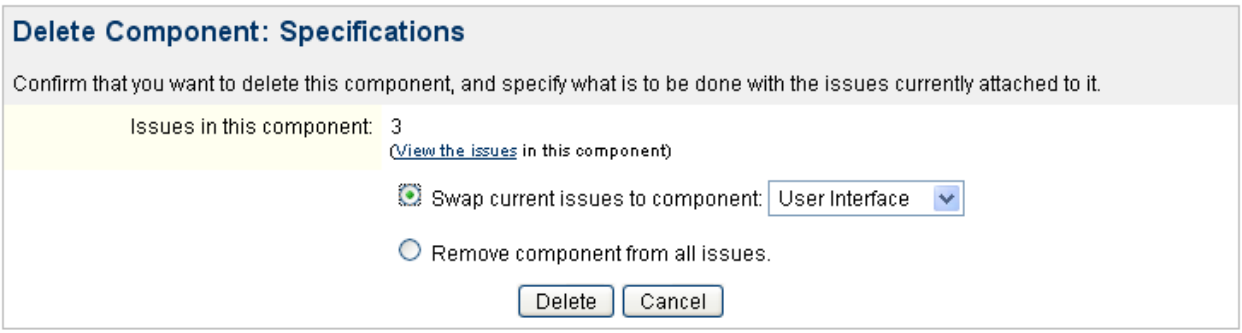
1. In the '**Components**' section (see 'Managing project components' above), click the 'Edit' link at the right of a the component you wish to edit.
2. This will bring you to the "Edit Component" page. Here, it is possible to edit the version name, description and lead.
3. Press the 'Update' button.
4. On completion of the update operation, you are returned to the project admin page - with an updated component list reflecting the changes made.



Edit Component

1.7.3.5. Deleting a component

1. In the '**Components**' section (see 'Managing project components' above), click the 'Delete' link at the right of the component you wish to delete.
2. This will bring you to the "Delete Component" page.
3. On this page you can specify the action to be taken regarding the issues in this component. You can either associate these issues with another active component, or have the references removed.
4. Press the 'Delete' button.
5. On completion of the delete operation, you are returned to the project admin page — with an updated component list reflecting the changes made.



Delete Component

1.7.4. Version Management

Versions are points-in-time for a project. They help you schedule and organise your releases. Once a version is created, the following reports are useful:

- Road Map report - gives you a view of upcoming versions
- Change Log report - gives you a review of released versions

Change Log and Road Map are driven by the 'Fix For Version' field on each issue.

Versions can be:

- Added - create a new version against which issues can be aligned.
- Released - mark a version as released. This changes the Road Map report, Change Log report and some issue fields' drop-downs.
- Rescheduled - re-arrange the order of versions.

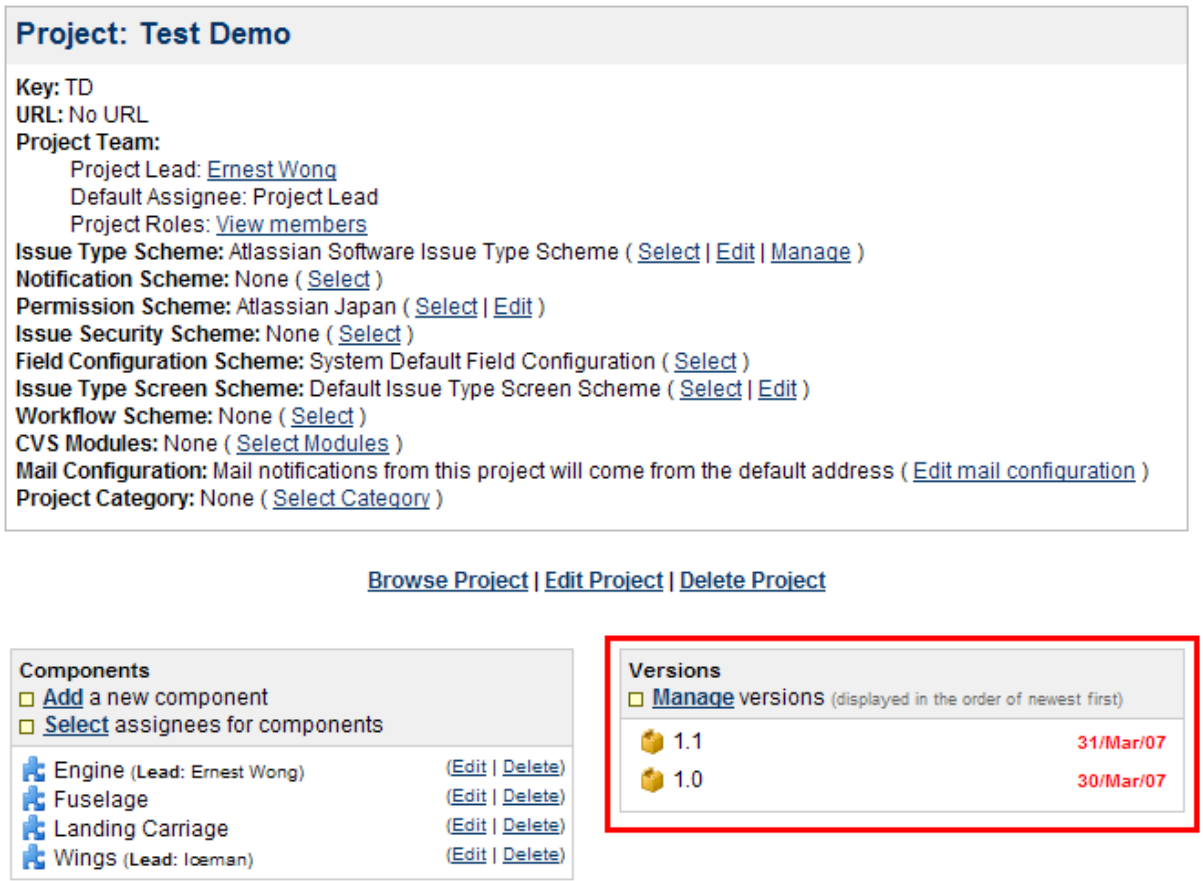
- Archived - hide an old version from the Road Map and Change Log reports, and in the JIRA User Interface.
- Merged - combine multiple versions into one.

1.7.4.1. Managing a project's versions

1. Login to JIRA as a project administrator. (A project administrator is someone who has the project-specific permission '**Administer Project**', but not necessarily the global permission '**JIRA Administrators**'.)
2. Click the '**Administration**' link on the top bar:

Click Administration link

3. This will display the **Administration** page, showing a list of projects which you have permission to manage. Click the project of interest.
4. You will now see a page displaying the project details. On this page all the configurable actions available on the project are easily accessible. On the lower right, a summary of the versions is displayed along with the link to the version management interface. The summary indicates the version status and the scheduled release date for that version.



Project configuration, with Version list highlighted

Note:

If you have created a new project and have not assigned a permission scheme with it on creation, then you will not see the above display. Instead under versions it will say "There are no versions at the moment".

5. Click the '**Manage**' link to display the '**Manage Versions**' screen, which shows a list of

versions and each version's status. From here you can perform the operations described below.

Version status

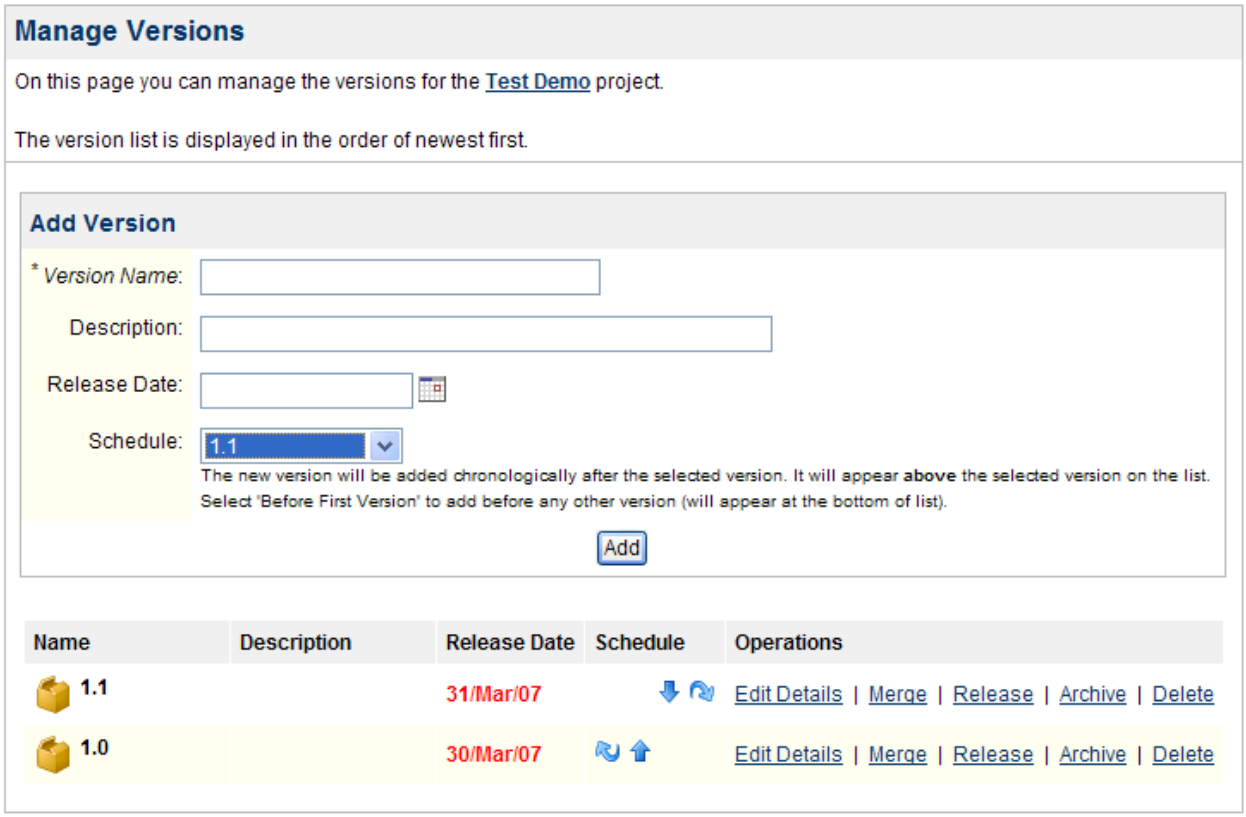
Each version can have any of the following four statuses:

- **Released** - a bundled package
- **Unreleased** - an open package
- **Archived** - a semi-transparent package
- **Overdue** - the release date is highlighted

Note:
The status affects where the version appears in drop-down lists for version-related issue fields ('Fix For Version' and 'Affects Version').

1.7.4.2. Adding a new version

1. The "Add New Version" form is located at the bottom of the '**Manage Versions**' screen.
2. From here, you enter the name for the version. The name is treated as a plain string by JIRA, so it can be simple numeric, e.g. "2.1", it can be complicated numeric, e.g. "2.1.3", or it can be something zany like the project's internal code-name, e.g. "Memphis".
3. Optional details such as the version description and release date can be also be specified.
4. It is also possible to 'schedule' the new version by selecting its position in the version list. The new version is added after the selected version — or it can be placed at the start of the list by selecting 'First'.
5. Click on the "Add" button. The version management list is updated immediately, with the newly created version added in the specified position.



Version Management

Note:
At present, the version release date is only used to indicate the scheduled release date for a version. Currently JIRA only uses the release date for reporting purposes, but in future releases it is intended to add further functionality to take advantage of this data.

1.7.4.3. Releasing a version

1. On the '**Manage Versions**' screen, click the 'Release' link available in the '**Operations**' column for a version you are interested in.
2. If there are any issues set with this version as the 'Fix For' version, JIRA allows you to select if you wish to change the 'Fix For' version. Otherwise, the operation will complete without modifying these issues.
3. This operation immediately updates the specific version as 'released' throughout JIRA.
4. The version list indicates the version 'released' status with the bundled package icon. The 'Unrelease' operation replaces the 'Release' operation in the operations column.
5. To un-release a version, simply click on the 'Unrelease' link in the operations column.

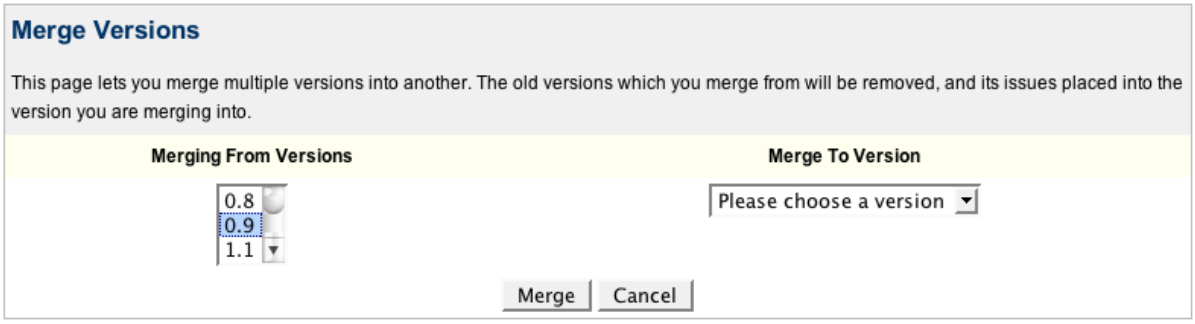
1.7.4.4. Archiving a version

1. On the '**Manage Versions**' screen, click the 'Archive' link available in the '**Operations**' column for a version you are interested in.
2. This operation immediately updates the specific version as 'archived' throughout JIRA.
3. The version list indicates the version 'archived' status with a semi-transparent icon. The list of available operations is replaced with the 'Unarchive' operation. No further changes can be made to this version unless it is un-archived. Also it is not possible to remove any existing archived versions from an issue's affected and fix version fields or add any new archived versions.
4. To un-archive a version, simply click on the 'Unarchive' link in the operations column.

1.7.4.5. Merging multiple versions

1. On the '**Manage Versions**' screen, click the 'Merge' link available in the '**Operations**' column for a version you are interested in.
2. This will take you to the 'Merge Versions' page.

   On this page are two select lists - both listing all un-archived versions. The specified version is highlighted in the 'Merging From Versions' select list on the left. It is possible to select further versions you wish to merge from. Versions selected on this list will be removed from the system. All issues associated with these versions will be updated to reflect the new version selected in the 'Merge To Version' select list on the right. It is only possible to select one version to merge to.



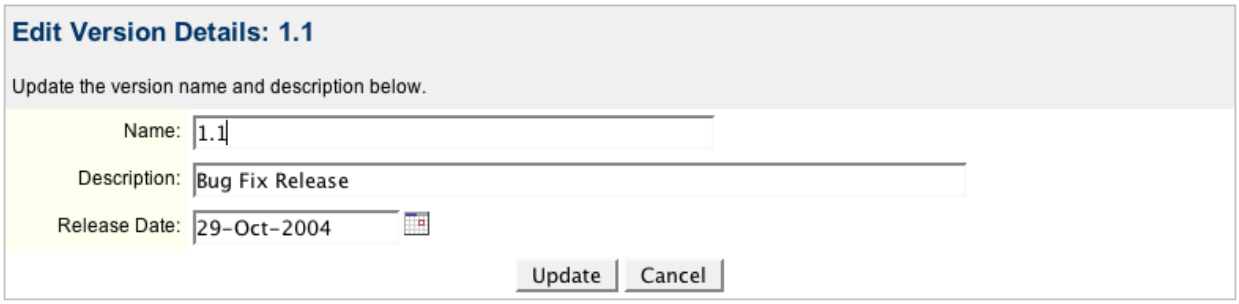
Merging Multiple Versions

3. Click on the 'Merge' button. You will be shown a confirmation Page. Click on 'Merge' to complete the operation.
4. On completion of the merge operation, you are returned to the version management interface. The version list has been updated to reflect the changes that occurred in the merge operation.

1.7.4.6. Editing a version's details

1. On the '**Manage Versions**' screen, click the 'Edit Details' link available in the '**Operations**' column for a version you are interested in.

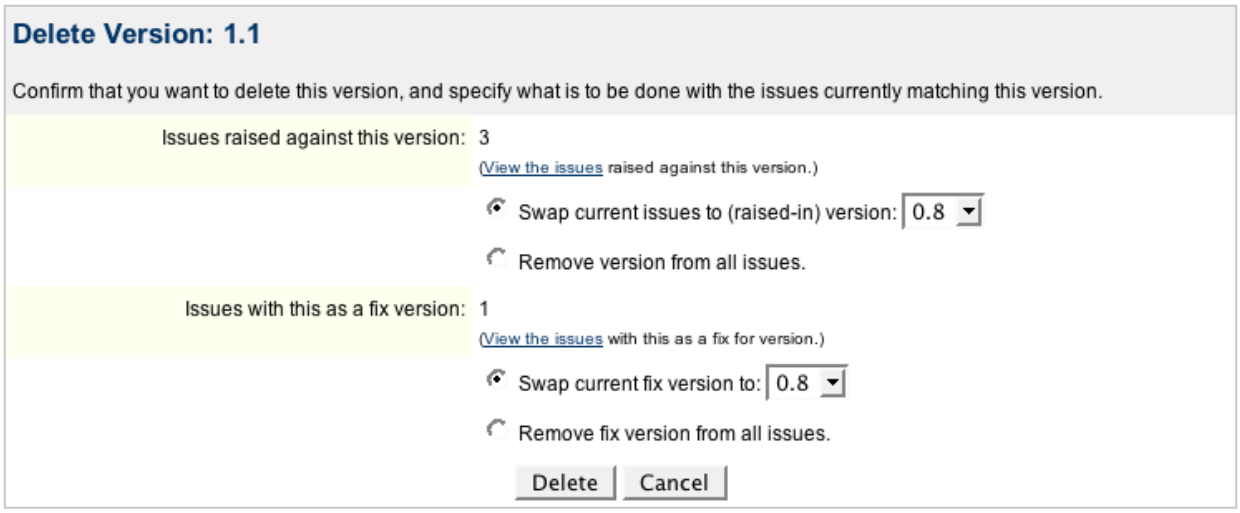
2. This will bring you to "Edit Version: <Version>" page. Here, it is possible to edit the version name, description and release date.
3. Press the 'Update' button.
4. On completion of the update operation, you are returned to the version management screen - with an updated version list reflecting the changes made.



Editing Version Details

1.7.4.7. Deleting a version

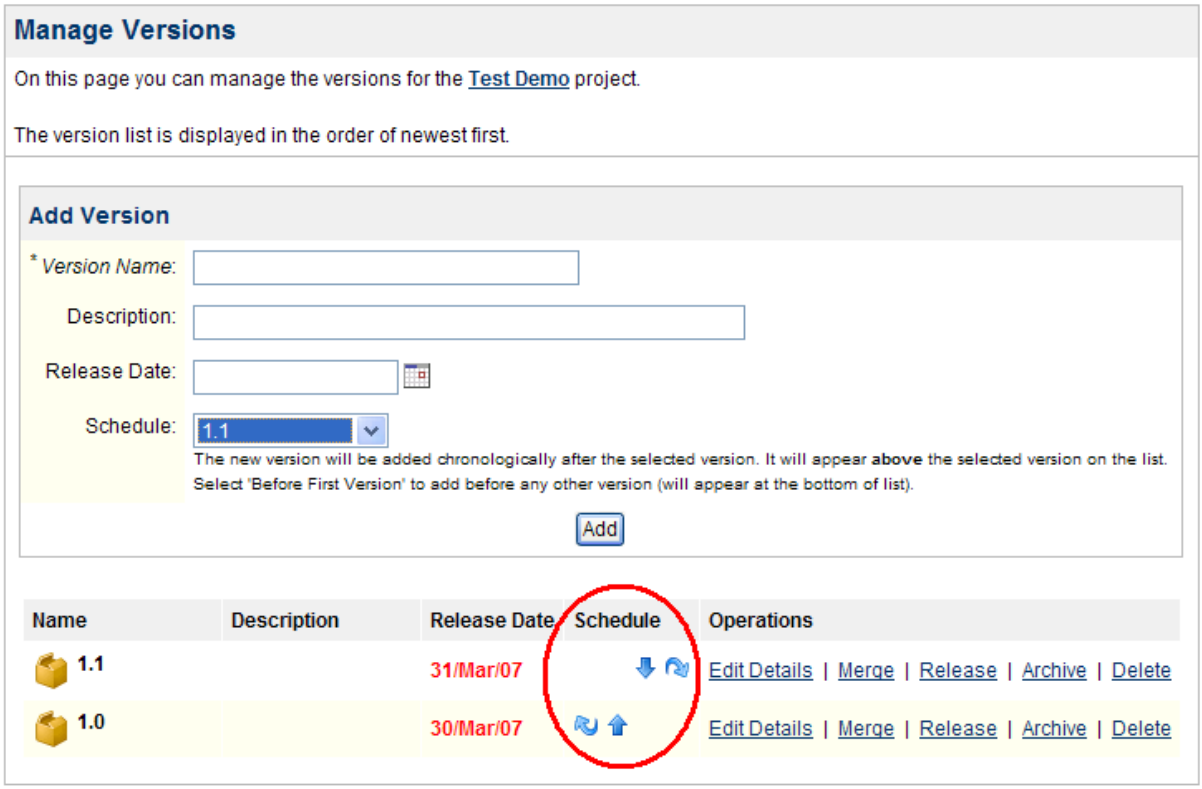
1. On the '**Manage Versions**' screen, click the 'Delete' link in the '**Operations**' column for the version you wish to delete.
2. This will bring you to the "Delete Version: <Version>" confirmation page.
3. From here, you can specify the actions to be taken for issues associated with the version to be deleted. It is possible to associate these issues with another version or to simply remove references to the version to be deleted.
4. Press the "Delete" button.
5. On completion of the deletion, you are returned to the version management screen - with an updated version list reflecting the changes made.



Deleting a Version

1.7.4.8. Rescheduling a version

1. On the '**Manage Versions**' screen, re-scheduling operations are available through the 'Schedule' column.
2. It is possible to move a version up/down a position or to the start/end of the list by clicking on the specific arrow icon associated with the specific version row.
3. The version list is updated immediately with the selected version now occupying the specified position.



Scheduling Versions

1.7.5. Project Release Notes

JIRA provides the functionality to create release notes for a specific version of a project. The release notes contain all issues within the specified project that are marked with a specific "Fix For" version. The release notes can also be generated in a number of formats (e.g. HTML, plain text, etc.) so as they can be included in various documents.

At present, two example format templates are provided - HTML and Text - using Velocity templates. Further format templates can be created and added to the system.

1.7.5.1. Generating Release Notes

1. Select the **Browse Projects** option from the main menu
2. Select the project from which you wish to create release notes.
3. Select **Release Notes**.
4. Select the required format of the release notes - HTML and plain text format templates are provided.
5. Select the required project version for which the release notes will be generated.
6. Selecting the **Create** button will generate the release notes using the specified template in the specified format. The release notes are displayed as HTML and also within a text area in the actual format selected - allowing the contents to be copied for inclusion in another document.

Note:

It is also possible to create a default set of release notes using the HTML format template. By selecting the **Roadmap** option from the **Reports** menu, it is possible to create the HTML release notes for each un-released version by selecting the **Release Notes** link after each version title.

1.7.5.2. Adding New Format Templates

1. Create a Velocity template similar in content to that of the examples provided - `releasenotes-text.vm` and `releasenotes-html.vm`. Consult the JIRA API

documentation and the Jakarta Velocity User Guide.
2. The title within the template should be modified along with the code within the text area. The other sections of the template do not need to be modified.
3. Add the new format template to the list within the `jira-application.properties` file. A corresponding entry must be made in both the `jira.releasenotes.templatenames` and the `jira.releasenotes.templates` lists. It is also necessary that the entries in both lists are in the same order.
4. Restart JIRA.
5. The new format template is available for selection as a release note format template.

Also see the tutorial on How to Create a Custom Release Notes Template Containing Comments.

1.7.6. Configuring Project Keys

JIRA provides the ability to configure the format of project keys within the system. This is achieved by defining a regular expression 'rule' that governs the valid project key format.

1.7.6.1. Project Key Pattern

Through the property `jira.projectkey.pattern`, the administrator can specify a Perl5 regular expression defining the rule for a valid project key. This property can be found in the `jira-application.properties` file. During project creation, the user must specify a project key that conforms to this rule.

JIRA prepends the regular expression specified with '^' and closes it with '$' for an exact matching rule within the system. The project key must only be allowed to contain ASCII characters, as it is used in HTTP GET requests.

1.7.6.2. Editing the key pattern

This can be done by editing jira-application.properties. You will then need to restart JIRA (JIRA Standalone) or rebuild the JIRA webapp and redeploy in your app server.

1.7.6.3. Project Key Details

The `jira-application.properties` file also contains the following properties:

- `jira.projectkey.description` - a configurable description (to match the project key pattern) displayed on project creation
- `jira.projectkey.warning` - a configurable validation warning (to match the project key pattern)

Note:
It is not possible to configure the issue key as JIRA expects this key to conform to specific rules.

Note:
Further information on Perl5 is available here.

1.8. Configuring Fields and Screens

1.8.1. Configuring Fields and Screens: Overview

To help you tailor JIRA to your organisation's needs, JIRA enables you to manipulate the display and behaviour of issue fields ('Summary', 'Description', 'Issue Type', etc).

You can:

- Change a field's description
- Make a field hidden or visible
- Make a field required or optional
- Add your own values for 'Issue Type', 'Priority', 'Resolution' and 'Status'
- Create new 'custom' fields
- Enable a rich text renderer for (some) fields
- Position fields on a screen
- Choose which screen should be displayed for each issue operation (e.g. 'Create Issue', 'Edit Issue') or workflow transition (e.g. 'Resolve Issue', 'Close Issue')

1.8.1.1. Concepts

Some key JIRA concepts include:

- Field Configuration — a set of definitions for all fields, comprising: each field's description; whether each field is hidden or visible; whether each field is required or optional; and what type of renderer to use for each text field.
- Screen — defines which fields are present on a screen, and their order. (Note that a hidden field can be present on a screen, but will still be invisible.)
- Screen Scheme — associates different screens with different issue operations (e.g. 'Create Issue', 'Edit Issue', 'View Issue') and (in *Professional Edition only*) projects.
- *(Enterprise Edition only)* Issue Type Screen Scheme — associates Screen Schemes with issue types and projects.
- *(Enterprise Edition only)* Field Configuration Scheme — associates Field Configurations with issue types and projects.
- *(Professional and Enterprise Editions only)* Issue Type Scheme — associates issue types with projects.

Below is an overview of the functionality available in the three different editions of JIRA in relation to these concepts.

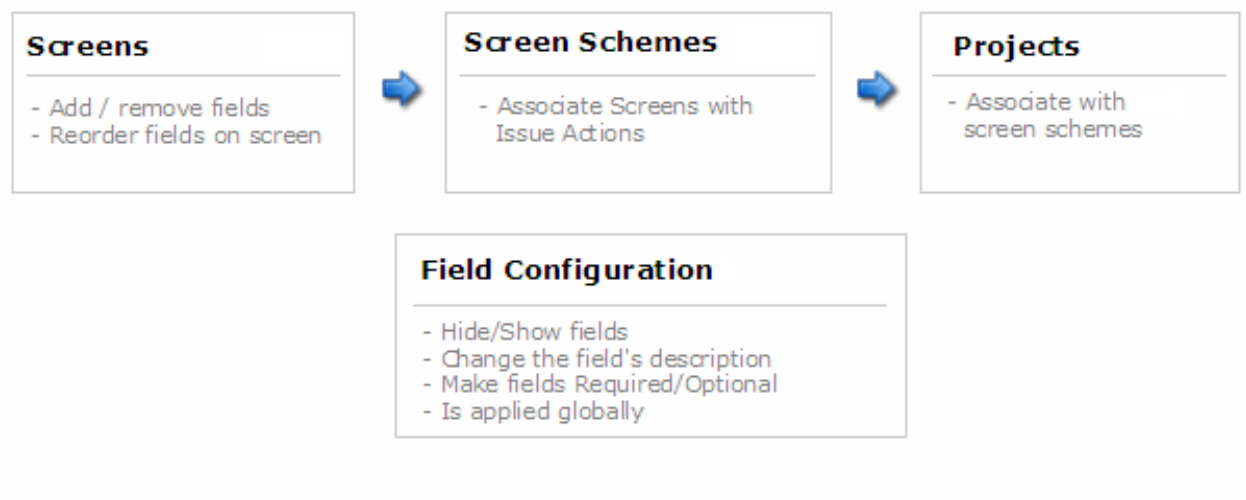
Standard Edition: Basic Configuration

In Standard Edition, you have one global Field Configuration and one global Screen Scheme. Any changes you make to the Field Configuration, Screens or Screen Scheme are applicable across all projects.

Professional Edition: Screens Per Project

Professional Edition provides multiple Screen Schemes. Each project has a Screen Scheme. Each Screen Scheme can have different Screens associated with different issue operations. For example, you can have different 'Edit' screens from one project to another. Once you have set up the Screen Schemes, you can reuse the same scheme for multiple projects.

Note that, in Professional Edition, any changes you make to the Field Configuration are applicable across all projects.



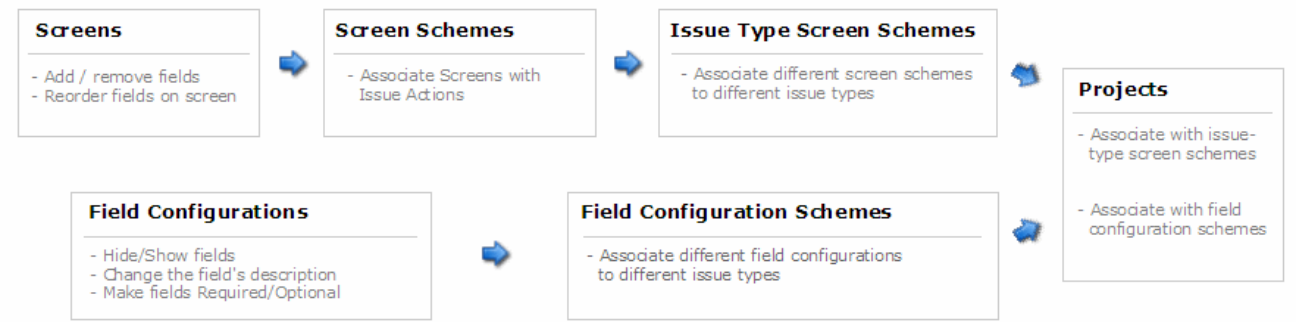
Professional Edition

Enterprise Edition: Screens and Field Configurations Per Project and Issue Type

JIRA adds further flexibility for Enterprise Edition by providing different Screens and Field Configurations per issue type.

Each project has an Issue Type Screen Scheme. Issue Type Screen Schemes allow you to associate Screen Schemes with issue types. This allows you to specify different Screens for the same operation (e.g. 'Create Issue') in the same project for issues of different types. For example, you could use one screen when creating an issue of type 'Bug', and a different screen when creating an issue of type 'Task'.

Additionally, each project has a Field Configuration Scheme. Field Configuration Schemes allow you to associate Field Configurations with issue types. This allows you to specify different field behaviour for the same field in the same project for issues of different types.

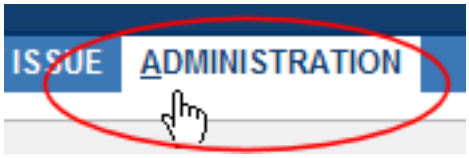


Enterprise Edition

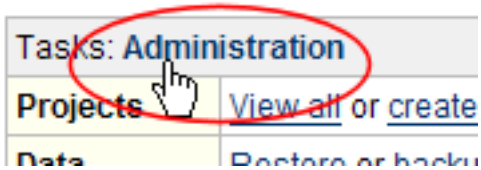
1.8.2. Defining New Resolutions

Resolutions are the ways in which an issue can be closed. JIRA ships with a set of standard resolutions, but you can add your own. To do so, follow the following steps.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

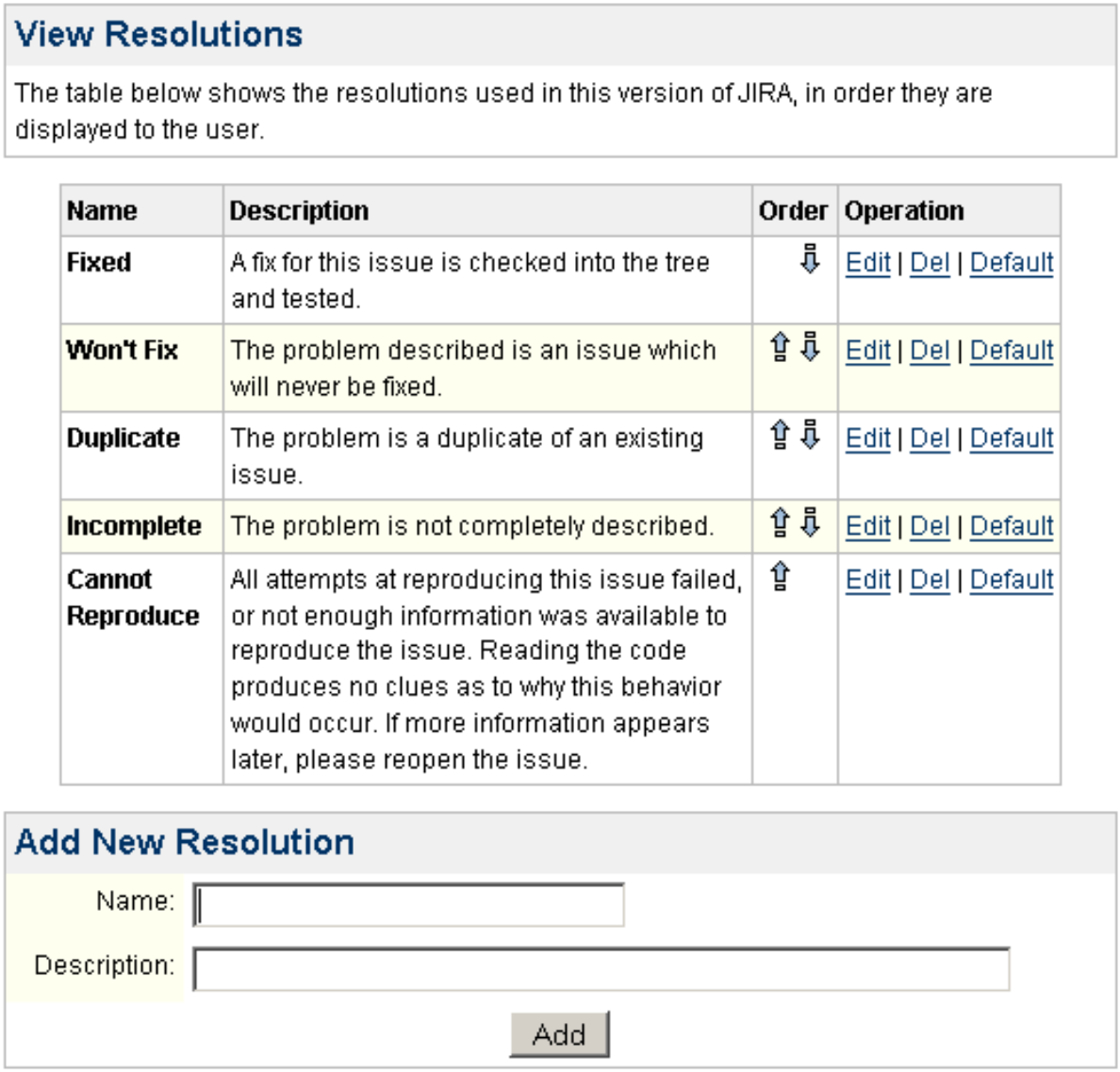


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title "Issue Settings", click on the link labelled "Resolutions".
4. This will bring up the View Resolutions page. The page lists the standard resolutions, along with a form underneath to add new resolutions.



View Resolutions

The table below shows the resolutions used in this version of JIRA, in order they are displayed to the user.

Name	Description	Order	Operation
Fixed	A fix for this issue is checked into the tree and tested.	⇩	Edit | Del | Default
Won't Fix	The problem described is an issue which will never be fixed.	⇧ ⇩	Edit | Del | Default
Duplicate	The problem is a duplicate of an existing issue.	⇧ ⇩	Edit | Del | Default
Incomplete	The problem is not completely described.	⇧ ⇩	Edit | Del | Default
Cannot Reproduce	All attempts at reproducing this issue failed, or not enough information was available to reproduce the issue. Reading the code produces no clues as to why this behavior would occur. If more information appears later, please reopen the issue.	⇧	Edit | Del | Default

Screenshot of page for adding new Resolutions

5. To add a new resolution, fill in the Add New Resolution form. For the name put a short phrase that best describes your new resolution. For the description, put a sentence or two to

describe when this resolution should be used.

6. The View Resolutions table can be used to edit, delete, set as default, and re-order the resolutions as they are displayed to the user who is resolving an issue.

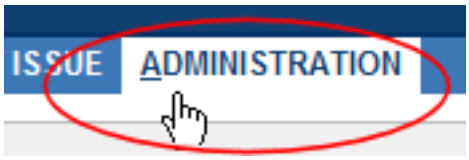
1.8.3. Defining 'Priority' Field Values

An issue's *priority* is its importance in relation to other issues.

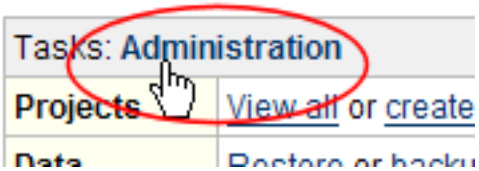
JIRA ships with a set of default priorities. You can modify these or add your own as follows.

1.8.3.1. Defining a new priority

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

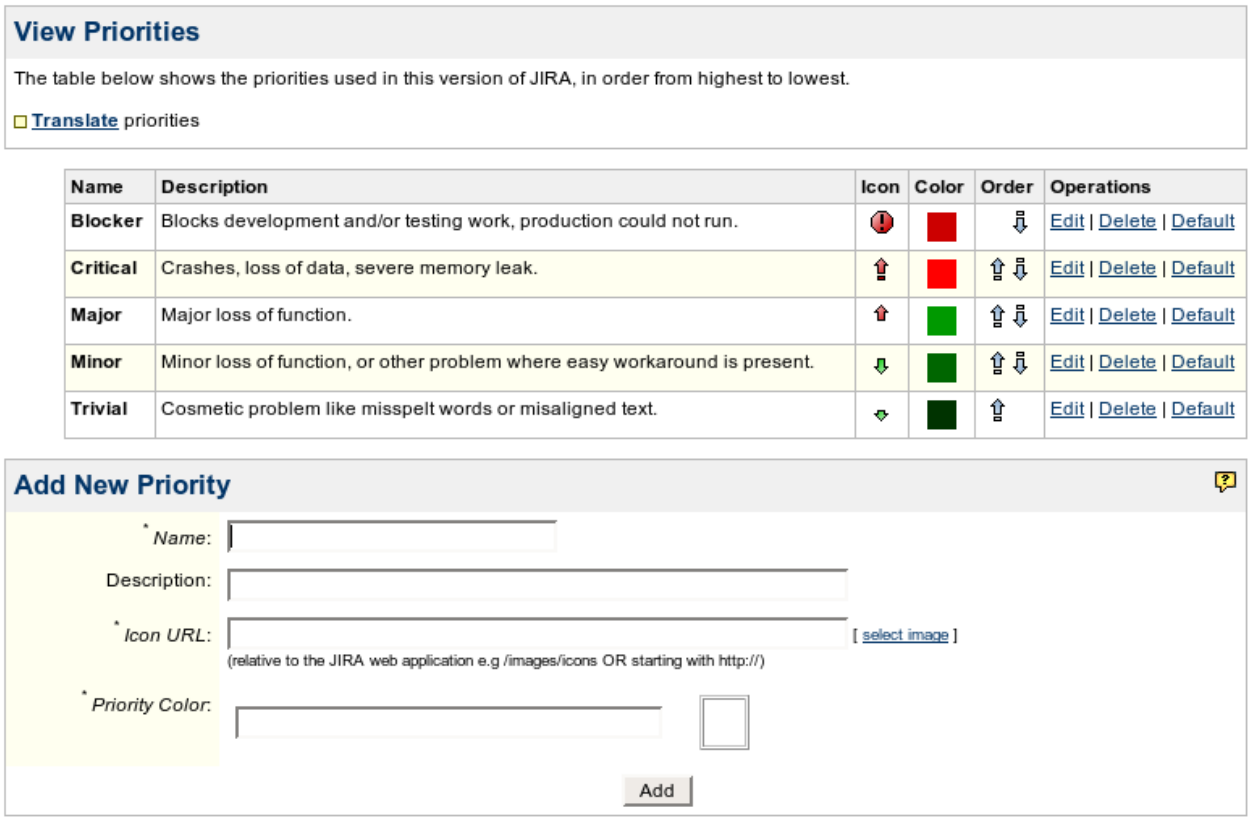


Link to Administration section



Link to Administration section

3. In the left-hand panel, under the title '**Issue Settings**', click the '**Priorities**' link.
4. The '**View Priorities**' page will appear. This page lists the currently-defined Priorities, below which is the '**Add New Priority**' form.



View Priorities

The table below shows the priorities used in this version of JIRA, in order from highest to lowest.

Translate priorities

Name	Description	Icon	Color	Order	Operations
Blocker	Blocks development and/or testing work, production could not run.				Edit | Delete | Default
Critical	Crashes, loss of data, severe memory leak.				Edit | Delete | Default
Major	Major loss of function.				Edit | Delete | Default
Minor	Minor loss of function, or other problem where easy workaround is present.				Edit | Delete | Default
Trivial	Cosmetic problem like misspelt words or misaligned text.				Edit | Delete | Default

Add New Priority

* Name:

Description:

* Icon URL: [select image]

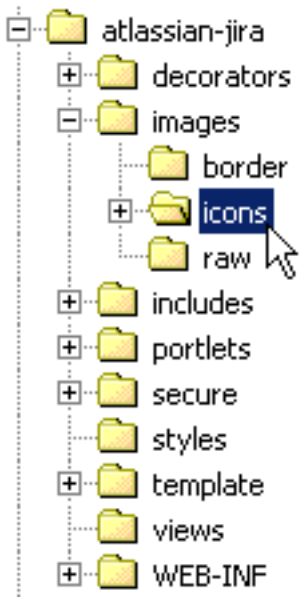
(relative to the JIRA web application e.g /images/icons OR starting with http://)

* Priority Color:

Add

'Add New Priority' screen

5. In the '**Name**' field, type a word or two to describe your new priority. (The Name will appear in the drop-down field when a user creates or edits an issue.)
6. In the '**Description**' field (optional), type a sentence or two to describe when this priority should be used.
7. In the '**Icon URL**' field, specify an image file to represent this priority. The dimensions of the image must be 16-pixels by 16-pixels. You can either type a URL, or click the '**Select image**' link to browse to a file location somewhere inside your JIRA installation directory, usually in /images/icons:



Directory view of icons dir in a JIRA webapp

8. In the '**Priority Color**' field, specify a colour to represent this priority. You can either type the HTML colour code, or click the box at the right of the field to select from a colour chart.
9. Click the '**Add**' button.

1.8.3.2. Editing a priority

1. Go to the '**View Priorities**' page as described in steps 1-4 of '**Adding a priority**' (above).
2. Click the '**Edit**' link corresponding to the priority you wish to edit.
3. Update the fields as described under '**Defining a new priority**' (above), then click the '**Update**' button.

1.8.3.3. Re-ordering priorities

Re-ordering priorities changes the order in which they appear in the drop-down list when a user creates or edits an issue.

1. Go to the '**View Priorities**' page as described in steps 1-4 of '**Adding a priority**' (above).
2. To re-order the priorities, click the arrows in the '**Order**' column:
   - Click the up-arrow to move a priority higher up in the list.
   - Click the down-arrow to move a priority lower down in the list.

1.8.3.4. Translating priorities

To translate your priorities into another language, please see Translating Resolution/Priority/Status/Type.

1.8.3.5. Deleting a priority

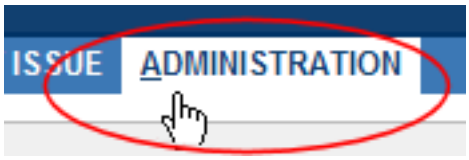
1. Go to the '**View Priorities**' page as described in steps 1-4 of '**Adding a priority**' (above).
2. Click the '**Del**' link corresponding to the priority you wish to delete.

1.8.4. Defining New Statuses

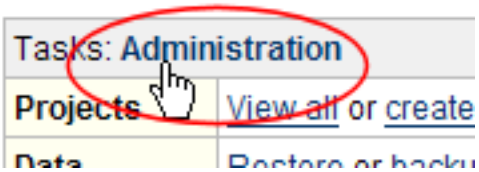
Statuses are used to represent the position of the issue in its workflow. A workflow represents a business process, represented as a set of stages that an issue goes through to reach a final stage (or one of the final stages). Each stage in the workflow (called a *workflow step*) is linked to an *issue status*, and an issue status can be linked to only one workflow step in a given workflow.

JIRA ships with a set of default statuses that are used by the default workflow. In JIRA *Enterprise* and *Professional editions* you can add your own statuses and customise the workflow. In *Standard edition*, workflow is not customisable and you cannot add new Statuses — but in all editions you can change the names, descriptions and icons of existing Statuses.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

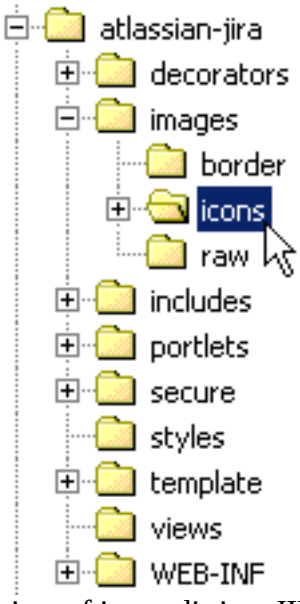
3. Open the left-hand sub-menu titled "Issue Settings" if it is not open already and click on the link labelled "Statuses".
4. This will bring up the *View Statuses* page. The page lists the existing Statuses, along with a form underneath to add a new Status.

Status Details	Mode	Workflows	Operations
Open The issue is open and ready for the assignee to start work on it.	Active	jira Copy of jira	Edit
In Progress This issue is being actively worked on at the moment by the assignee.	Active	jira Copy of jira	Edit
Reopened This issue was once resolved, but the resolution was deemed incorrect. From here issues are either marked assigned or resolved.	Active	jira Copy of jira	Edit
Resolved A resolution has been taken, and it is awaiting verification by reporter. From here issues are either reopened, or are closed.	Active	jira Copy of jira	Edit
Closed The issue is considered finished, the resolution is correct. Issues which are not closed can be reopened.	Active	jira Copy of jira	Edit

'Add New Status' screen

5. To add a new Status, fill in the Add New Status form. For the name put a short phrase that best describes your new Status. For the description, put a sentence or two to describe what stage this Status represents. For the Icon URL you need to supply the path of a 16 by 16 image that has

been placed somewhere inside JIRA's opened .WAR. We suggest you place it in /images/icons:



Directory view of icons dir in a JIRA webapp

JIRA ships with a number of images that can be used as status icons. These images are located in the /images/icons directory inside the JIRA .war and include:

- status_assigned.gif
- status_closed.gif
- status_document.gif
- status_down.gif
- status_email.gif
- status_generic.gif
- status_information.gif
- status_inprogress.gif
- status_invisible.gif
- status_needinfo.gif
- status_open.gif
- status_reopened.gif
- status_resolved.gif
- status_trash.gif
- status_unassigned.gif
- status_up.gif
- status_visible.gif

6. The View Statuses table can be used to edit and delete Statuses. Please note that only statuses that are not used in any workflow can be deleted.

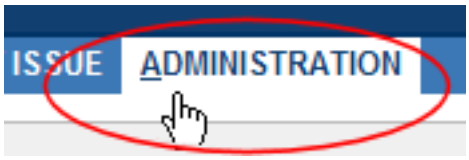
1.8.5. Defining New Issue Types

JIRA ships with a set of default issue types to help you get started with the issue tracker. Everyone's needs are different and so JIRA also allows you to add, edit and delete your own custom issue types.

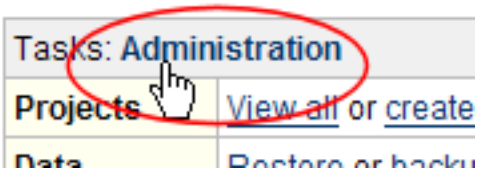
JIRA Professional and *Enterprise* editions also allow you to control the set of available Issue Types for each project — see Associating Issue Types with Projects. Additionally, *JIRA Enterprise* allows you to associate particular Issue Types with particular Fields, Screens and Workflow — for details see 'Associating Field Behaviour with Issue Types', 'Associating Screens with Issue Types' and 'Activating Workflow', respectively.

1.8.5.1. Creating, Editing and Deleting Issue Types

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

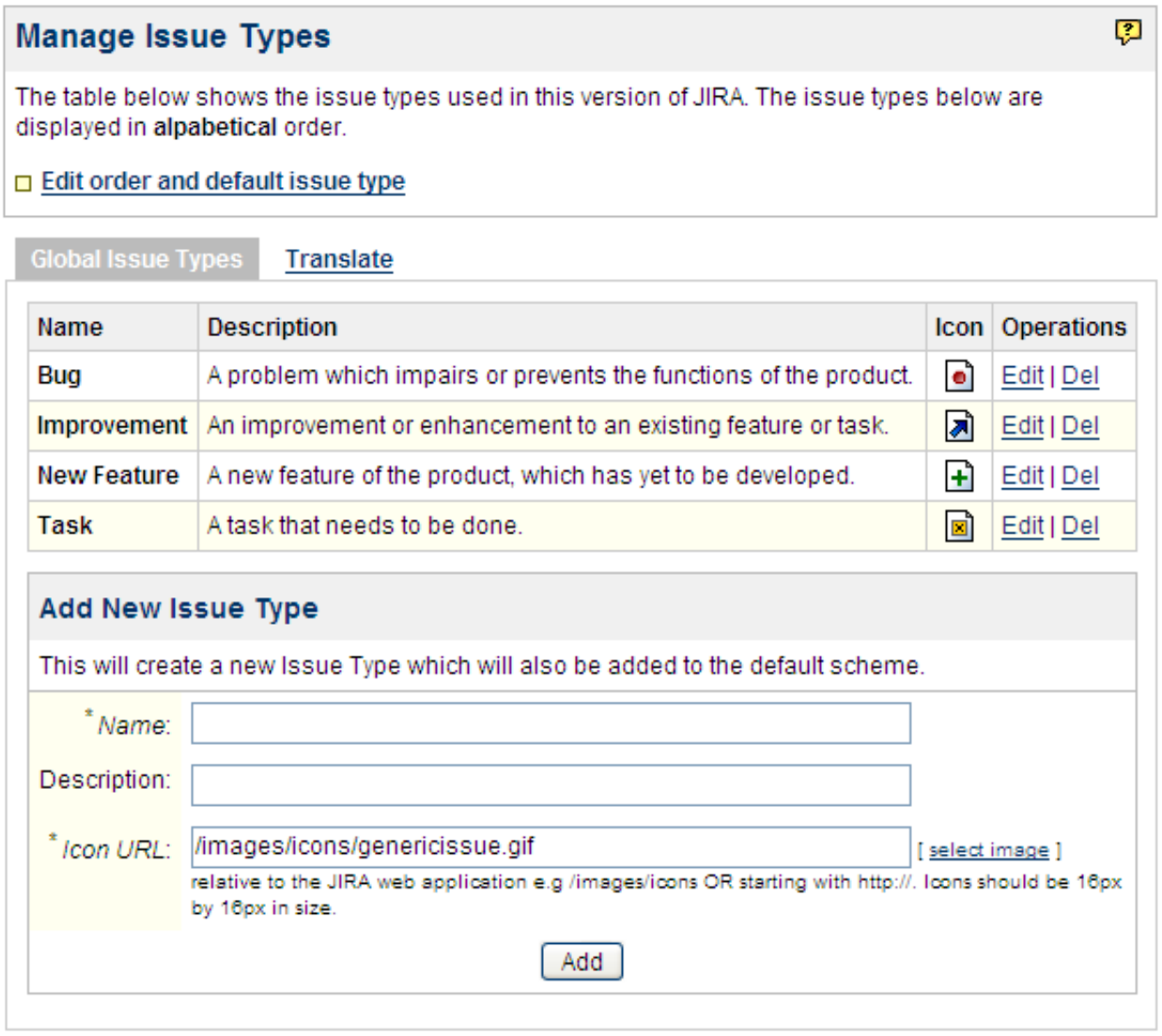


Link to Administration section



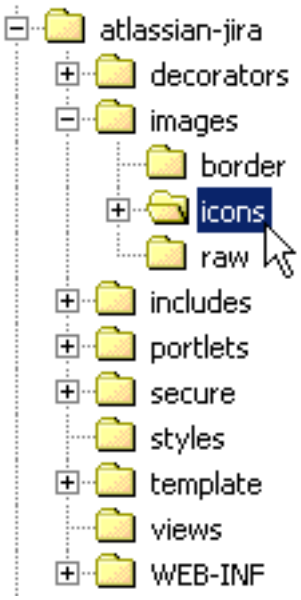
Link to Administration section

3. On the panel on the left, under the title "Issue Settings", click on the link labelled "Issue Types".
4. This will bring up the Manage Issue Types page. The page lists all issue types, along with a form underneath to add new issue types.



Manage Issue Types Screen

5. To add a new Issue Type, fill in the Add New Issue Type form. For the Name, enter a short phrase that best describes your new Issue Type. For the Description, enter a sentence or two to describe when this Issue Type should be used. For the Icon URL you need to supply the path of a 16-by-16-pixel image that has been placed somewhere inside JIRA's opened .war. We suggest you place it in /images/icons:



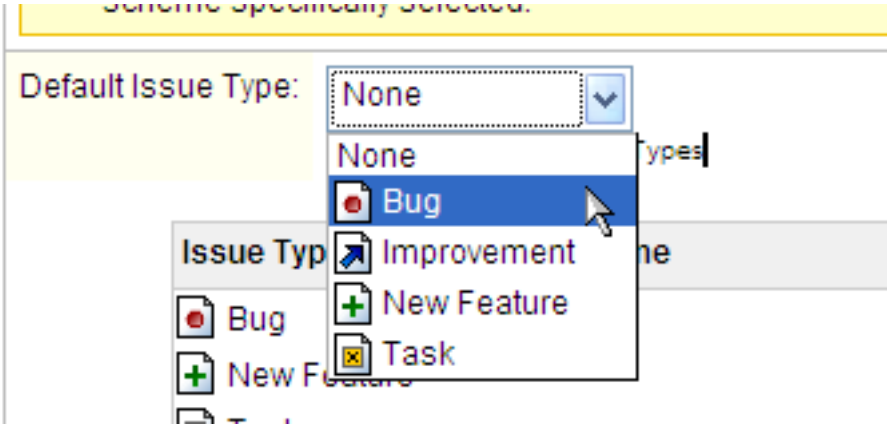
Directory view of icons dir in a JIRA webapp

Once you create your new Issue Type, it will be automatically added to the **Default Issue Type Scheme** (in *Professional* and *Enterprise Editions* of JIRA). For more information, see *Managing Issue Type Schemes*.

You can edit and delete an existing issue type by clicking on the *Edit* and *Del* links on the right.

1.8.5.2. Ordering issue types and setting defaults

1. Reordering issue types changes the order in which they are displayed to the user who is creating an issue; and the default issue type is the one that is displayed in the selection-box (see Creating an Issue).
   - In Standard Edition, you can choose the default issue type, and re-order issue types, by clicking on the link *"Edit order and default issue type"*.
   - In *Professional* and *Enterprise Editions* of JIRA, it is possible to configure different sets of issue types for each project. Each set can have a different order and default issue type. Refer to *Managing Issue Type Schemes* for more information.
2. Clicking on the *"Edit order and default issue type"* link will display the screen below. Here you can select the issue type that will be selected by default. If you don't select a default issue type, the user will see an option *"Please Select..."* when they're creating an issue.



Setting defaults and reordering issue types

3. To re-order the issue types, **drag and drop** the issue types into the correct order. Note that none of the changes are persisted until you click on the *Save* button. You can revert to the last saved changes any time by click on the *Reset* button. Clicking on *Cancel* will not save any changes.

Drag and Drop

1.8.6. Associating Issue Types with Projects

1.8.6.1. What is an 'Issue Type Scheme'?

JIRA Professional and *Enterprise* editions allow you to restrict the set of available issue types for each project. This is achieved through the use of **Issue Type Schemes**. An Issue Type Scheme represents a sub-set of issue types, with its own order and default value. An Issue Type Scheme can be shared across multiple projects, so that a group of similar projects can share the same issue type settings.

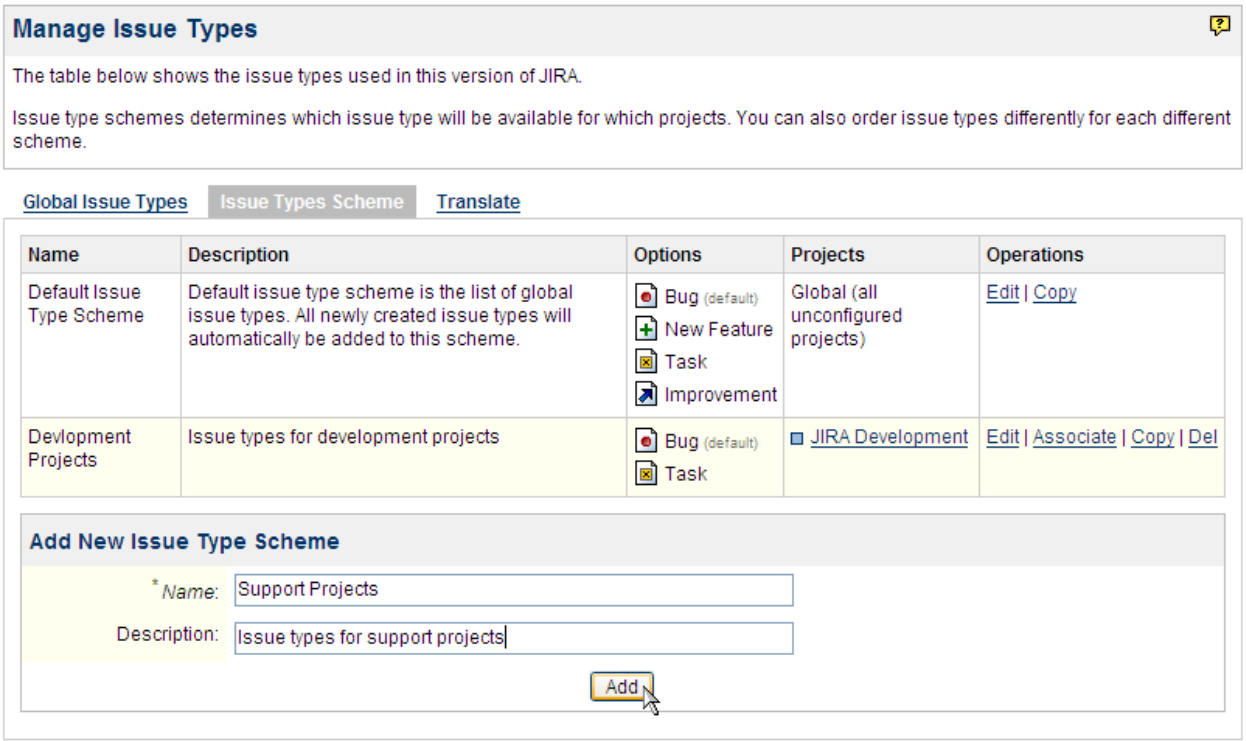
For example, in your company all projects may be one of two types, a development project or a support project. You could then create one scheme called *Task* with issue types *Bug* and *Development* and another called *Support Issue Types*, with *Development Query* and *Support Request*. You can then associate each scheme to the appropriate projects, giving your users a different set of issue types depending on which projects they decide to create issues in. Using schemes also gives you added benefit that any change you make to a scheme is made across all projects that are associated with the scheme. In this example, adding a new issue type to all support projects only requires a simple step of adding the issue type to the *Support Issue Types* scheme.

You can find out how to achieve this and more below.

1.8.6.2. Managing Issue Type Schemes

The Issue Type Schemes can be found on the **Issue Type Schemes** tab on the Issue Type page. Here you can see all existing Issue Type Schemes, their related issue types and the associated projects.

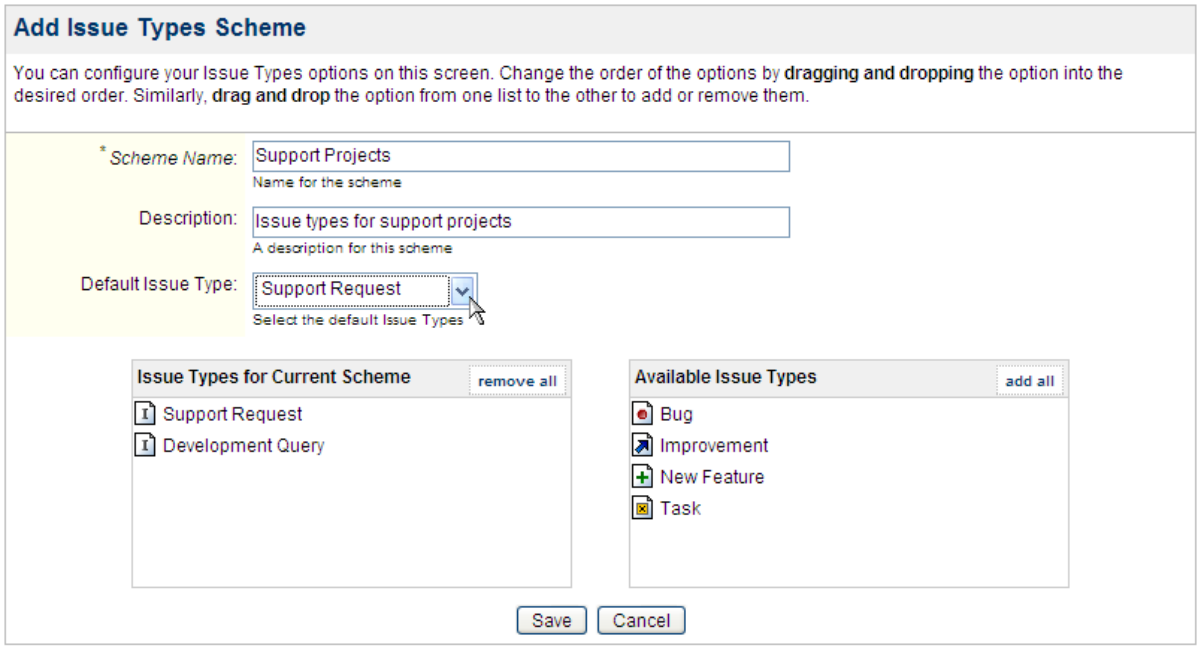
The **Default Issue Type Scheme** contains all the issue types that exist in your JIRA system. This scheme is associated with all newly created projects by default. If some of your issue types are not relevant to all of your projects, create one or more new Issue Type Schemes (e.g. 'Development Projects' in this screenshot) as described below, and associate them with the appropriate projects instead of using the **Default Issue Type Scheme**.



Manage Issue Types Schemes Screen

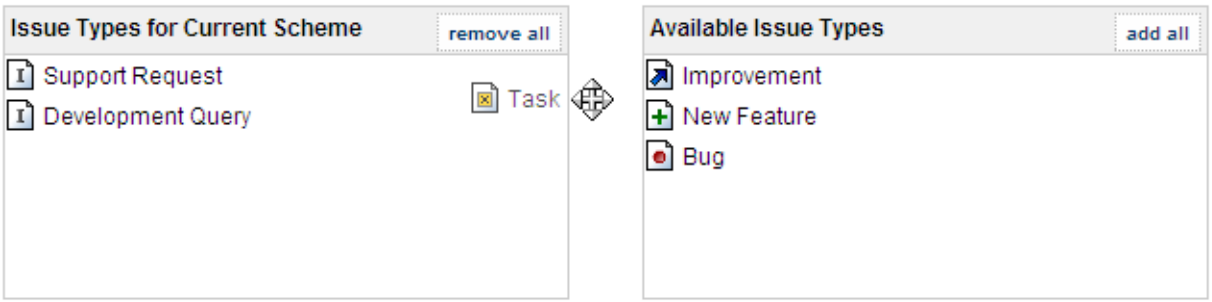
Creating a new scheme

1. To create a new scheme, enter the name and description for the new scheme. Ensure that the name is meaningful as this will be visible to other administrators and will allow them to better reuse the scheme.
2. Click on the *Add* button and the screen below will be displayed.



Manage Issue Types Scheme

3. Set the default for the new scheme from the select list. Add a new issue type to your scheme by dragging and dropping the issue type from the right hand list to the left. You can similarly reorder the issue types in the desired order by dragging and dropping them into the right positions.



Dragging and dropping issue types

4. If you need an issue type that does not currently exist, you can add this easily in using the add issue type form at the bottom of the page. This will add the issue type to the system and also add it to the scheme you're editing.

5. Once you've finished with your scheme. Click on the Save button to persist your changes. Note that unless you click on Save, no scheme will be created.

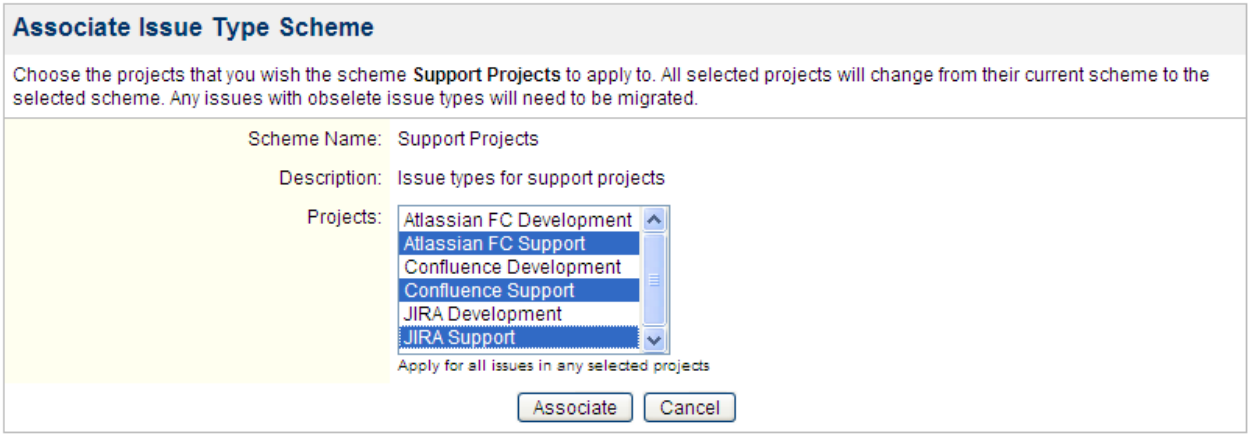
Editing a scheme

The process of editing a scheme is identical to the creation process. You can set defaults, reorder, add and remove issue types as before. However, if you're removing issue types from the scheme and there are issues associated with that issue type, you will be required to use the Issue Type Migration Wizard which will move your issues from the obsolete issue type to a valid one. Note that if you cancel out of this process at any time, your changes will not be committed. See below for more information about the migration wizard.

Associating a scheme to projects

You can restrict the issue types available by associating your Issue Type Scheme to various projects. Click on the *Associate* link and simply choose the projects that you wish your scheme to apply to. All selected projects will change from their current scheme to the selected scheme.

If the new scheme does not have an issue type that was present in the old scheme, you will be asked to use the Issue Type Migration Wizard to migrate the issues.

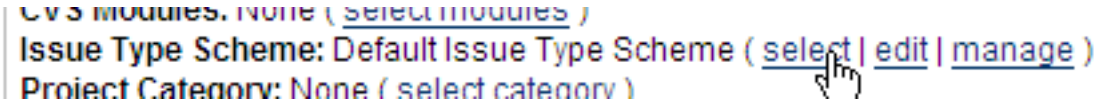


Associated Issue Type Scheme to Project

1.8.6.3. Managing Schemes for a Project

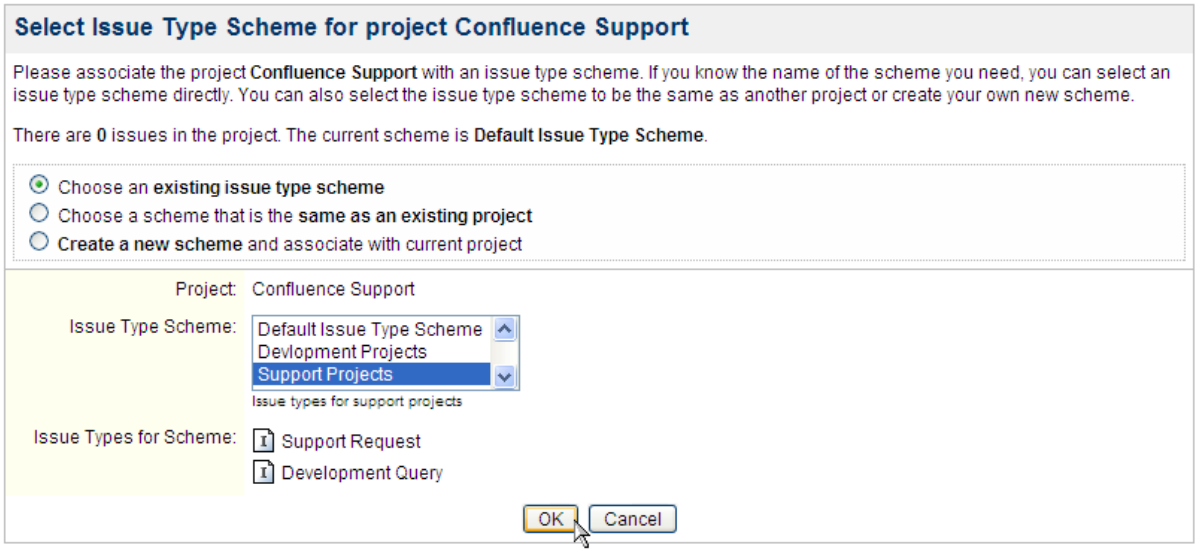
When updating a project you may often want to quickly restrict its issue types. However, the available Issue Type Schemes may not always be applicable, or you do not know which schemes to choose. The *Select Issue Type Scheme* screen makes this process simpler.

1. Click on the *select* link for Issue Type Scheme on the View Project page.



Select an Issue Type Scheme for Project

2. This will bring you the screen below.

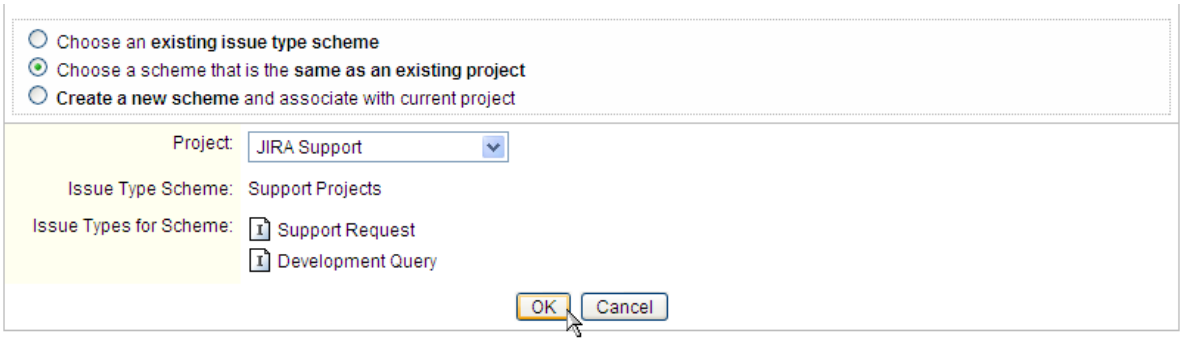


Select an Issue Type Scheme for Project

3. There are three ways you can select your issue type scheme. Select the radio button that is most relevant.
    1. *Choose an existing issue type scheme* - If you know the name of you scheme (e.g. Support

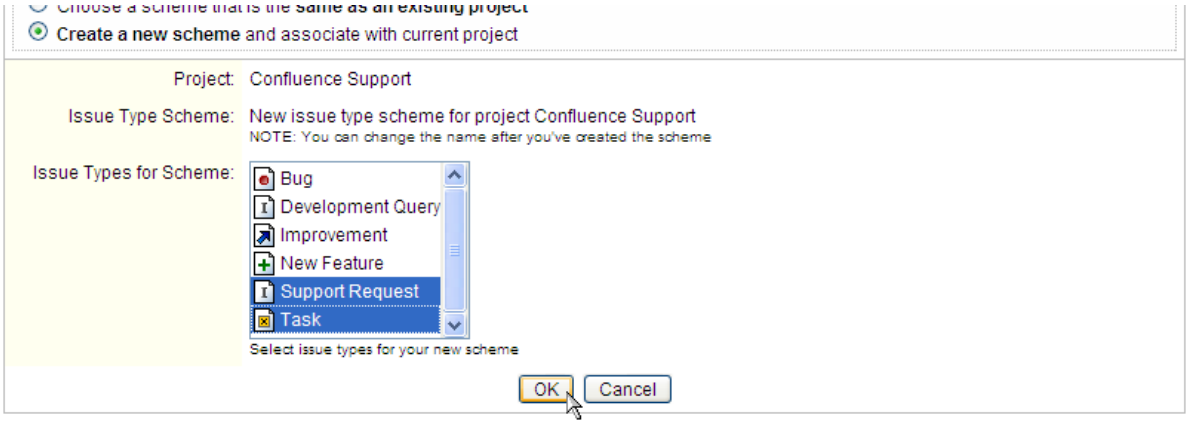
Issue Type Scheme), you can immediately choose it from the list. You will see a preview of issue types that would be available for your project as well as the description of the scheme.

2. *Choose a scheme that is the same as an existing project* - If you do not know the name of the scheme you would like to use, but you do know the name of the project whose set of issue types you wish to use for the project you are editing, please select this option. You will be prompted to select a project and the scheme that is currently associated with the selected project will be used for your project as well.



Select scheme same as an existing project

3. *Create a new scheme and associate with current project* - Select this option if you can't find any existing scheme that fits your needs and would like to quickly create a new scheme. Simply select the relevant issue types for your project and a new scheme will be created with the default name and order. You can edit the name, default value and order of the newly created scheme later.



Quickly add a new scheme

4. If after you make your changes there are any issues in the selected project that will have obsolete issue types, they will have to be migrated with the Issue Type Migration Wizard.

1.8.6.4. Issue Type Migration Wizard

This Issue Type Migration Wizard allows you to migrate issues from an obsolete issue type to a valid issue type. The wizard will be triggered whenever an action (e.g. editing a project's issue type scheme) results in an issue type becoming obsolete (not available in the scheme).

The wizard bears some resemblance to the Bulk Move function except for that you can't change the project of the issues. The major steps are:

1. Overview - provides a summary of the issues that will require migration
2. Choose Issue Type
3. Set new status
4. Set field values
5. Confirmation

Steps 2 to 4 will be repeated for each issue type that requires migration. After you have migrated all the issues you'll see a summary of changes that will occur. If you click on the Confirm button, the wizard will migrate your issues to the new issue types and then complete your action.

Please refer to the Bulk Move documentation for more information on status changes and setting of fields values.

1.8.7. Issue Constant Translations

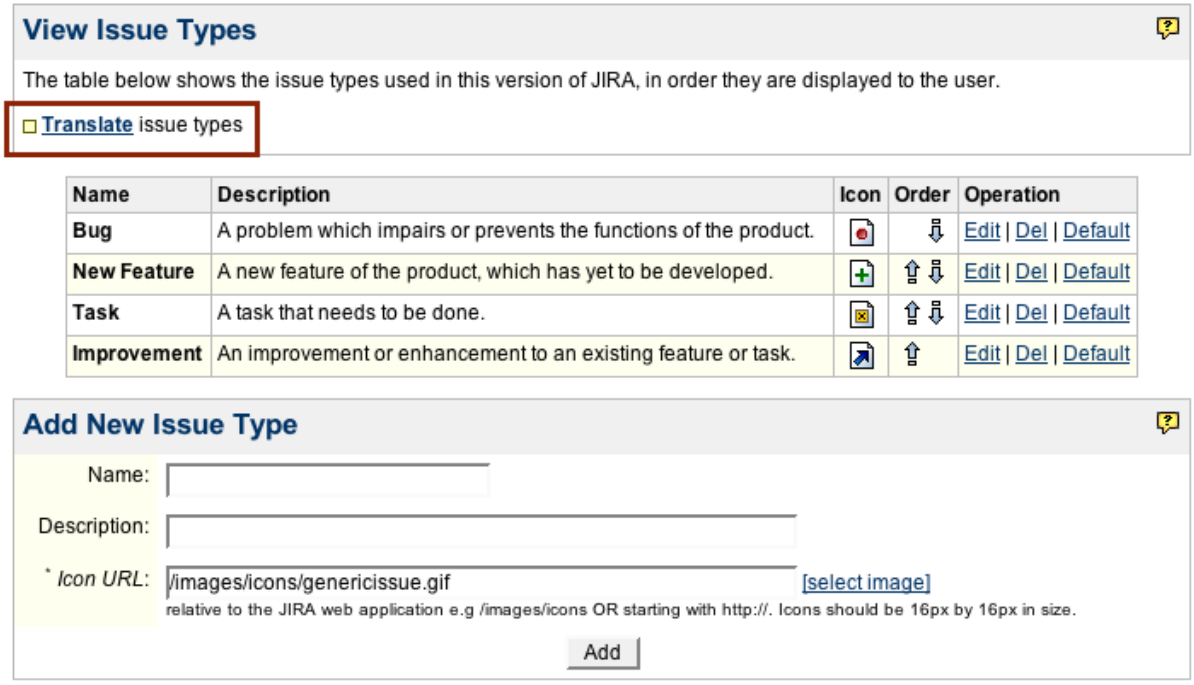
Further extending JIRA as an internationalisable issue manager, it is possible to specify a translated name and description for each issue constant - i.e all Issue Type, Status, Resolution and Priority fields.

This functionality allows the administrator to specify an issue constant translation set for each available language - providing each user with a more complete translation in their own chosen language. The translated issue constant names and descriptions appear throughout JIRA; in reports, portlets and all issue views.

1.8.7.1. Issue Constant Translation

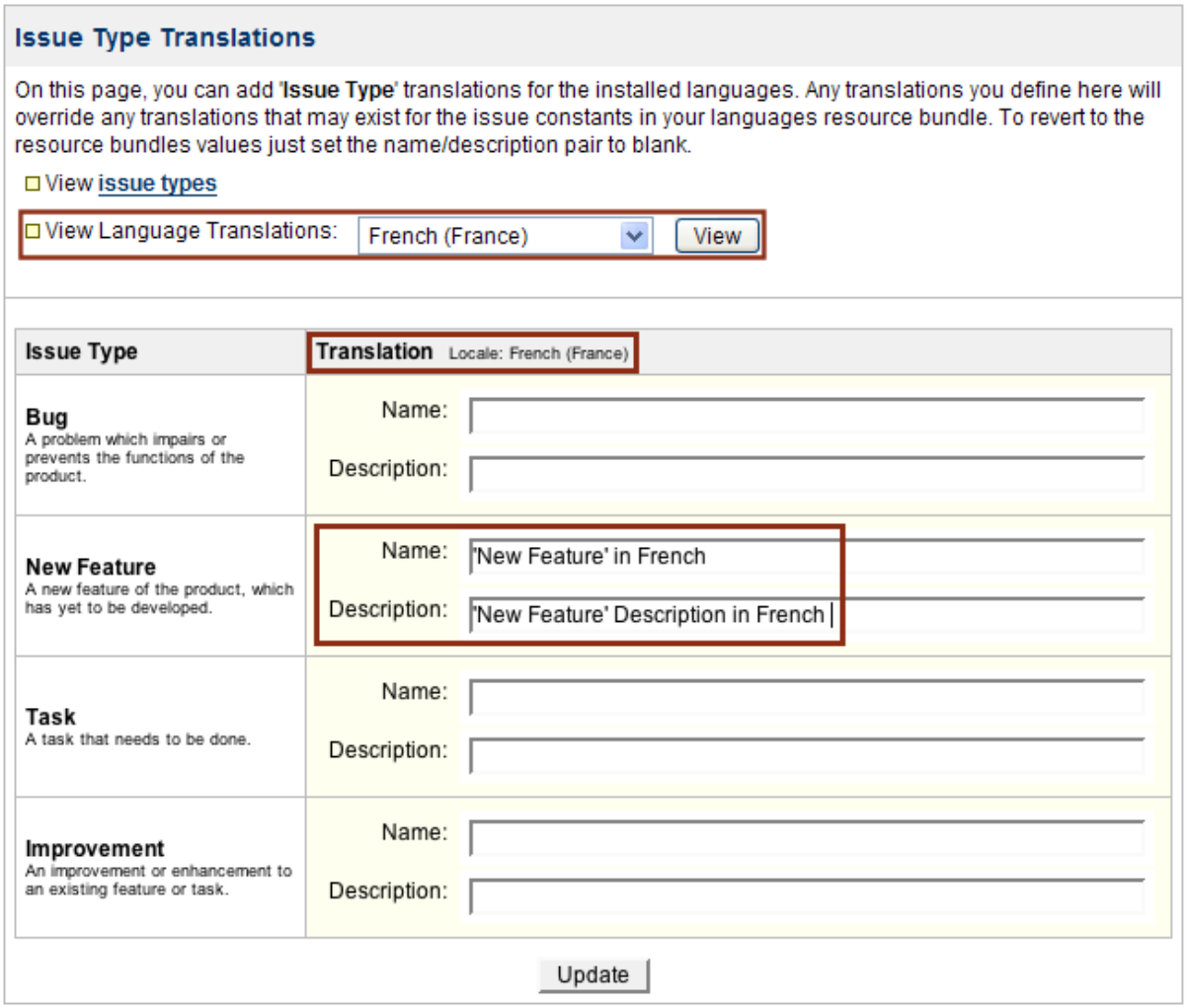
Each issue constant can be configured to have a translation set for each available language. The default issue constant name and description are displayed if no translation has been configured.

1. The issue constant translation operation is accessed through the **Translate** link located on each issue constant management screen.



Issue Constant Translation Link

2. The issue constant translation screen displays the translation set for the currently selected language. In order to view/update a translation set for a specific language, select the required language from the list at the top of the screen and click the **View** button.



Issue Constant Translation Screen

3. The currently selected language is displayed above the translation set table.
4. A translated name and description set can be specified for each type of issue constant. Once all translations have been entered, the translation set can be saved by clicking the **Update** button at the bottom of the translation table.
5. The process can be repeated for the all types of issue constants - i.e. Issue Type, Status, Resolution and Priority fields.
6. The translated issue constant name and description will be displayed throughout JIRA; in reports, portlets and all issue views.

Issue Constant Translation - Create Issue

Note:

The default issue constant name and description are displayed if a translation has not been specified.

1.8.8. Changing the behaviour of fields

1.8.8.1. What is a 'Field Configuration'?

A **Field Configuration** provides the ability to change field behaviour. A Field Configuration specifies:

- the **description** that appears under each field when an issue is edited
- whether each field is **hidden** or **visible**
- whether each field is **required** (i.e. the field will be validated to ensure it has been given a value) or **optional**
- (for text fields only) which **renderer** to use

Please note that in the *Enterprise* edition of JIRA it is possible to create multiple Field Configurations (see below). You can then associate different Field Configurations with different issue types by creating a Field Configuration Scheme. Once created, a Field Configuration Scheme can be associated with one or more projects, allowing you to control field behaviour on a per project, per issue type basis.

In the *Standard* and *Professional* editions of JIRA, only one Field Configuration exists. This Field Configuration applies to all projects and all issue types in the system.

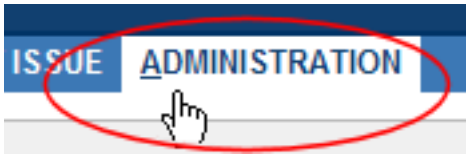
Note:

For information about placing fields on Screens, and using Screen Schemes to associate screens with issue operations, please see the Overview.

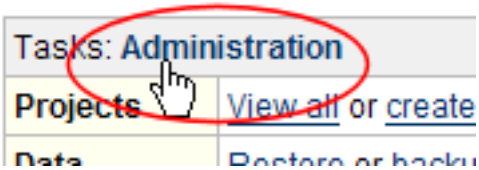
1.8.8.2. Editing a Field Configuration

To change the behaviour of fields, you first need to navigate to the Field Configuration. The way this is done depends on the edition of JIRA:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

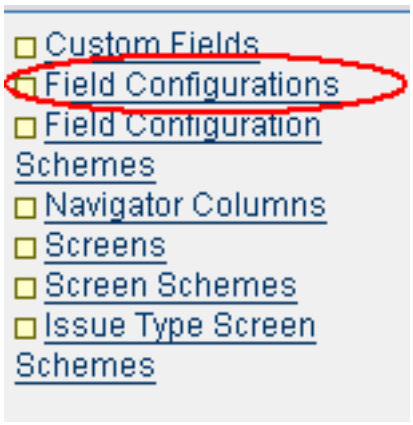


Link to Administration section

3. For *Standard* and *Professional* editions:
   Click the "Issue Fields" sub-menu on the left-hand column, and choose "Issue Fields" from the list.

Standard and Professional Edition Sub Menu for Issue Fields

For *Enterprise* edition:
Click the "Issue Fields" sub-menu on the left-hand column, and choose "Field Configurations" from the list. Select the Field Configuration of interest.



Enterprise Edition Sub Menu for Issue Fields

4. This will bring you to the "View Field Configuration" page.

The "View Field Configuration" page lists all system and custom fields in a table as shown below. The "Operations" column lists all the operations that are available for each field. These operations could be:

- Edit — change the field's description
- Hide/Show — hide the field from view or show it
- Require/Optional — set a field to be required (so that whenever a field is edited it must be given a value) or optional.
- Renderers — change a field's renderer (see Configuring Renderers for more information).

View Field Configuration

The table below shows all fields configured in JIRA and their properties for **Default Field Configuration**.

You can use this page to make fields required, hide/show fields and specify their description. You can also change the screens the field appears on by using the "Screens" link next to each field.

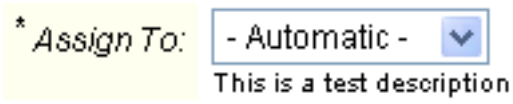
- View all field configurations
- Restore Defaults

Name	Screens	Operations
Affects Version/s	Default Screen	Edit | Hide | Required | Screens
Assign To	Assign Issue Screen Default Screen Resolve Issue Screen	Edit | Hide | Screens
Attachment	Default Screen	Edit | Show
Comment [Default Text Renderer]	This field can not be placed on screens by users.	Edit | Renderers
Component/s	Default Screen	Edit | Hide | Required | Screens
Custom text area [Wiki Style Renderer]	Default Screen	Edit | Hide | Required | Screens | Renderers
Custom text field [Wiki Style Renderer]	Default Screen	Edit | Hide | Required | Screens | Renderers
Description [Wiki Style Renderer]	Default Screen	Edit | Hide | Required | Screens | Renderers
Due Date	Default Screen	Edit | Hide | Required | Screens
Environment [Default Text Renderer] For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).	Default Screen	Edit | Hide | Required | Screens | Renderers
Fix Version/s	Default Screen Resolve Issue Screen	Edit | Hide | Required | Screens
Issue Type Required	Default Screen	Edit | Screens
Priority	Default Screen	Edit | Hide | Required | Screens
Reporter	Default Screen	Edit | Hide | Required | Screens

Field Configuration: View Issue Fields

Editing a Field's Description

Fields can be given descriptions to better identify the meaning of the field. These descriptions are usually displayed under the field on the creation of an issue and whenever it is edited. An example of this, is shown below.



Field Description Example

To edit the description of a field:

1. On the "View Field Configuration" page, click on the "Edit" link next to the field you want to change. This will bring you to the "Edit Field Description" page.

2. On this page you can edit the field's description
3. Press the "Update" button.

Edit Issue Field Description

Hiding/Showing a field

If your organisation or project has no use for a particular field, you have the option to hide it. Hiding a field will ensure that the the field does not appear on any Screens (i.e. issue operation Screens, workflow transition Screens) where the Field Configuration applies.

Hiding a field in the Field Configuration is distinct from not adding a field to a Screen. Fields hidden through the Field Configuration will be hidden in *all* applicable Screens, regardless of whether or not they have been added to the Screen.

Note:

For fields that have a default value:
If the field is hidden in the Field Configuration, then it will not receive a value when an issue is created, regardless of whether the field is present on the 'Create Issue' screen(s). (The following fields can have a default value: 'Resolution'; 'Status'; 'Priority'; 'Issue Type'; custom fields.)

To hide a field:

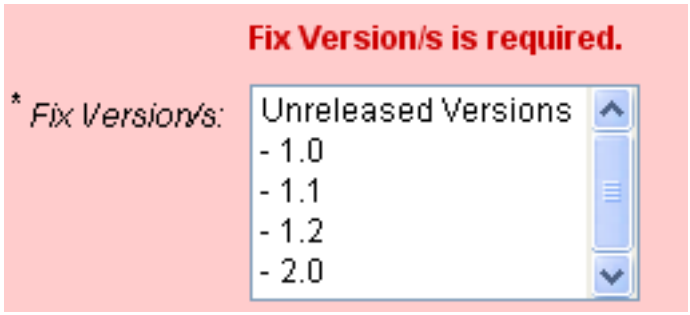
1. On the "View Field Configuration" page, click the "Hide" link next to a field you no longer want. The field will then fade to grey to signify that it has been hidden.
2. As per the image above, the "Environment" field has been hidden. You can make this field visible again anytime by pressing on the "Show" link.

Note:

The fields "Summary" and "Issue Type" cannot be hidden and as such there is no "Hide" option available for these fields.

Required/Optional fields

Certain fields within your organisation may be compulsory for issues. In this case you can set a field to be required, so that JIRA validates that the field has been given a value whenever an issue is edited. If a required field has not been given a value, JIRA will return an error informing the user that the field should be filled, e.g.:



Field required error

Note:
If you set a field to "required", ensure that the field is present on your 'Create Issue' screen(s). Note that in *JIRA Enterprise edition* you can have different field configurations for different projects and issue types (see 'Associating field behaviour with Issue Types'); so you need to ensure that all "required" fields are present on the 'Create Issue' screens for all associated projects and issue types (see 'Associating screens with Projects and Issue Types').

To make a field required:

1. On the "View Field Configuration" page, click the "Required" link next to the appropriate field. The text "Required" will appear next to the field.
2. As per the image above, the "Fix Versions" field has been made required. You can make this field optional again anytime by pressing on the "Optional" link.

Note:
Fields that are hidden cannot be set to required. Making a hidden field required will make it "shown" as well.

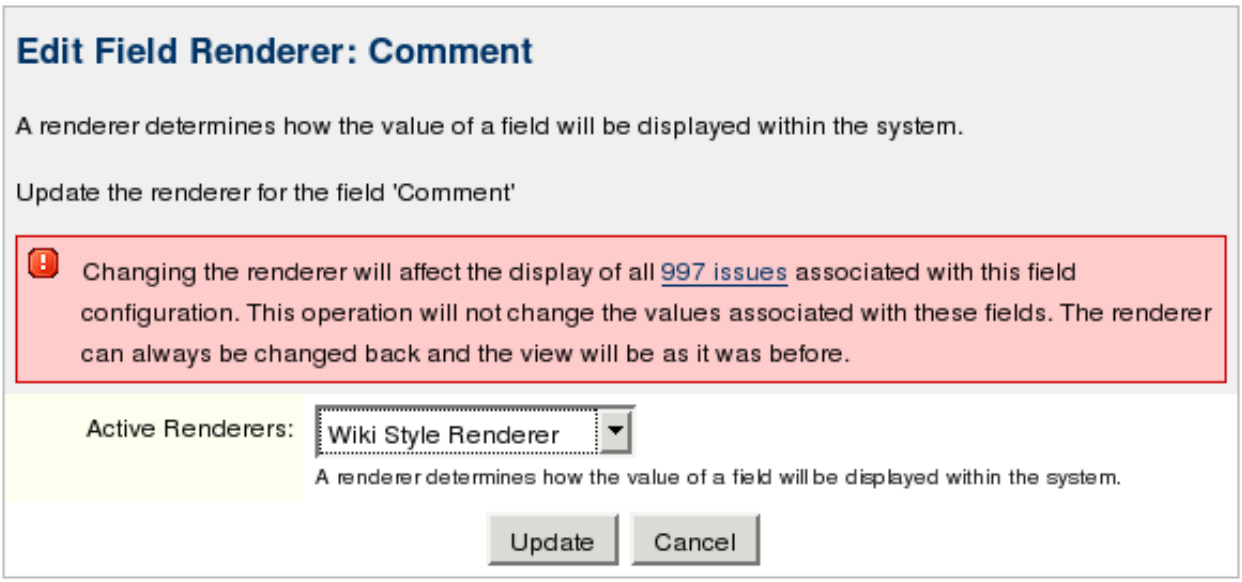
Renderers

Note:
Before you begin, please read 'Configuring rich-text renderers', paying particular attention to the section 'Implications for JIRA operations'.

The "View Field Configuration" page indicates which renderers are currently enabled for all renderable fields.

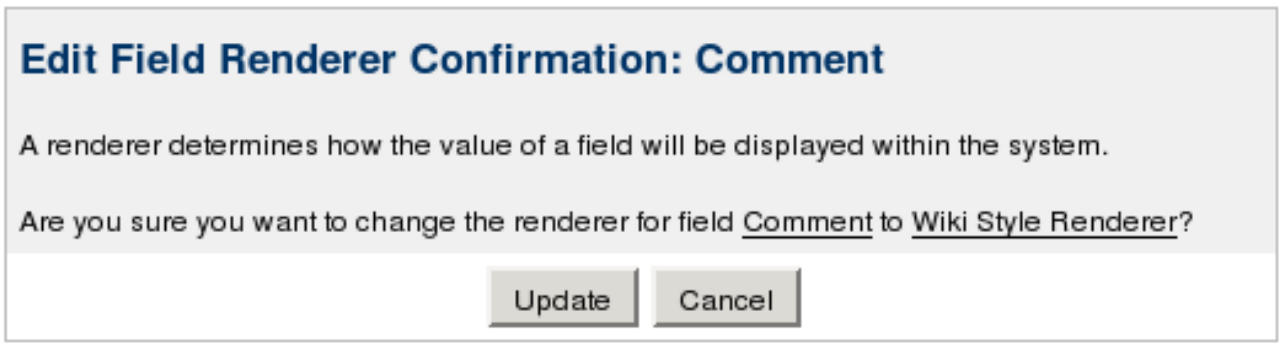
In the above screenshot you will notice the grey text '[Wiki Style Renderer]' under the Description field's name. This indicates that the field is currently configured to use the Atlassian Wiki Renderer. The grey text '[Default Text Renderer]' under the Comment field's name indicates that the field is currently configured to use the Default Text Renderer.

To change the renderer type for a specific field, click on the 'Renderers' link in the 'Operations' column of the screen for the field you want to change. This will take you to a page where you will have the option to select a renderer from all configured and available renderers.



Renderer configuration screen

As shown above, this page will warn you if there are issues that will be affected by the change. If no issues will be affected then the warning does not show. From this page, choose the renderer you wish to use and click 'Update'. You are then presented with a confirmation page, shown below.



Renderer configuration confirmation screen

Click the "Update" button to finish setting the new renderer on the field.

Note:

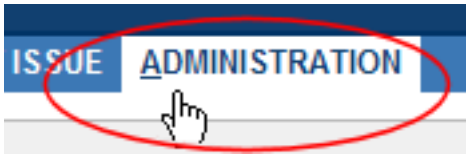
Changing the renderer only affects the *display* of the issue data that exists in the system. You can therefore toggle back and forth between renderer types safely.

1.8.8.3. Managing Multiple Field Configurations

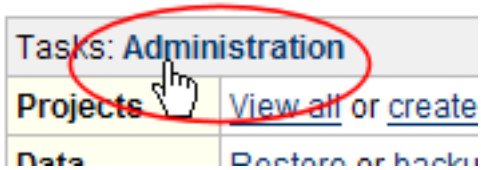
In *Enterprise* edition it is possible to create multiple field configurations for use on separate projects and issue types. Multiple field configurations are organised into Field Configuration Schemes based on issue type. A scheme can be associated with one or more projects, allowing administrators to control fields on per project per issue type basis.

Therefore, in the *Enterprise* edition it is possible to create, edit, delete and copy field configurations. All these operations are available from the "View Field Configuration" page. To navigate to this page:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

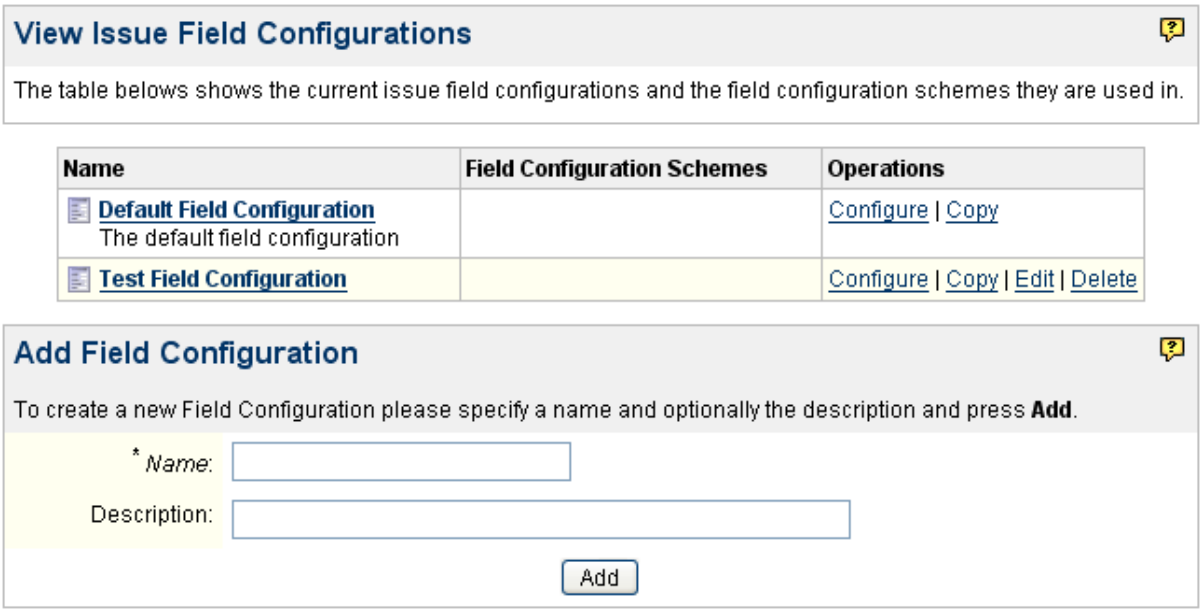


Link to Administration section



Link to Administration section

3. Click the "Issue Fields" sub-menu on the left-hand column, and choose "Field Configurations" from the list.
4. You will now see a list of all your current field configurations.



View Issue Field Configurations

Default Field Configuration

When JIRA Enterprise is installed only the Default Field Configuration is created. All new projects are associated with this configuration. This configuration is also used for projects that are not associated with a Field Configuration Scheme.

Note:
It is not possible to delete the Default Field Configuration.

Adding a Field Configuration

1. The "Add" new field configuration form is located at the bottom of the "View Field Configuration" page
2. Enter the name of the new configuration on this form.
3. You can optionally add a description to this configuration for better identification.
4. Click on the "Add" button to submit the form. The page will be automatically updated with your new field configuration added.

Note:
A newly created Field Configuration will not take effect until it has been activated (see below).

Editing a Field Configuration

1. On the "View Field Configuration" page, click the "Edit" link next to the desired field configuration.
2. You will now see the "Edit Field Configuration" page. Here it is possible to edit the configuration name and description.
3. Click on the "Update" button.
4. On completion you will be returned to the page where you can view your changes.

Deleting a Field Configuration

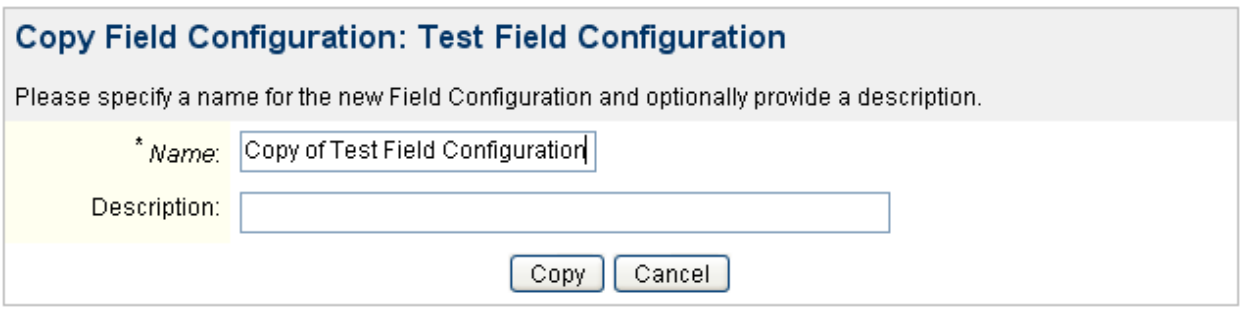
1. On the "View Field Configuration" page, click the "Delete" link next to the desired field configuration.
2. Click the "Delete" button to confirm this operation.
3. The "View Field Configuration" page will now show the updated list of the field configurations.

Note:
The Default Field Configuration cannot be deleted.

Note:
In *JIRA Enterprise edition*, you can only delete a field configuration that is not associated with a Field Configuration Scheme.

Copying a Field Configuration

1. On the "View Field Configuration" page, click the "Copy" link next to the field configuration you wish to copy
2. This will bring you to the "Copy Field Configuration" page.
3. Enter the name and description of the new field configuration.
4. Click on the "Copy" button.
5. You will now be directed back the "View Field Configuration" page, with your new field configuration added to the list. The field settings on the original and the new field configurations will be identical.



Copy Field Configuration

Note:
A newly created Field Configuration will not take effect until it has been activated (see below).

1.8.8.4. Activating a Field Configuration

To activate a Field Configuration in the *Enterprise* edition of JIRA: configure a Field Configuration Scheme to associate the Field Configuration with appropriate issue types; then associate the Field Configuration Scheme with a project. For details of both procedures, see 'Associating field behaviour with Issue Types'.

In the *Standard* and *Professional* editions of JIRA, only one Field Configuration exists. This Field Configuration applies to all projects and all issue types in the system, and is always active — no activation steps are required.

1.8.9. Configuring Rich-Text Renderers

1.8.9.1. Overview

JIRA *renderers* affect the display of a field's content. Renderers were introduced in JIRA 3.4 to allow a greater range of expression within text-based fields such as the **Description** and **Comment** fields — see 'Renderable Fields' (below) for a full list.

JIRA currently ships with two renderers: the Default Text Renderer, which displays plain text; and the Atlassian Wiki Renderer (utilising the Confluence wiki engine), which displays rich text (HTML). See 'Renderer Types' (below) for a full list.

Renderers are configured on a per field level, allowing a flexible combination of plain text and rich text fields. To configure a renderer for a particular field, see 'Configuring field behaviour'. Note that *JIRA Enterprise Edition* also gives you the flexibility to configure fields differently per project and per issue type.

Renderers are implemented as JIRA plugins, meaning that any renderer can be easily added to or removed from use within JIRA. This includes any custom renderers that may be developed. For details see 'Configuring Renderers' (below).

Please read 'Implications for JIRA operations' (below) before configuring renderers

Note:

Renderers affect the rendering (view) of a field's value. This means that you can migrate to a different renderer without affecting your issue data; only the view will be changed. It also means that if you do not like the way your issues look using the new renderer you can simply switch back with no impact on your issue data.

1.8.9.2. Renderable Fields

Potentially any field within JIRA can be a renderable field, but this only really makes sense in the case of text-based fields (since a date field would look nonsensical in wiki-markup). The following table shows the JIRA fields that are renderable out-of-the-box:

Field Type	Description
Description	The description field of an Issue can have a renderer applied.
Comment	The comments field of an Issue can have a renderer applied.
Environment	The environment field of an Issue can have a renderer applied.
Custom Field - Free Text Field (unlimited text)	Any instance of this type of custom field can have a renderer applied.
Custom Field - Text Field	Any instance of this type of custom field can have a renderer applied.

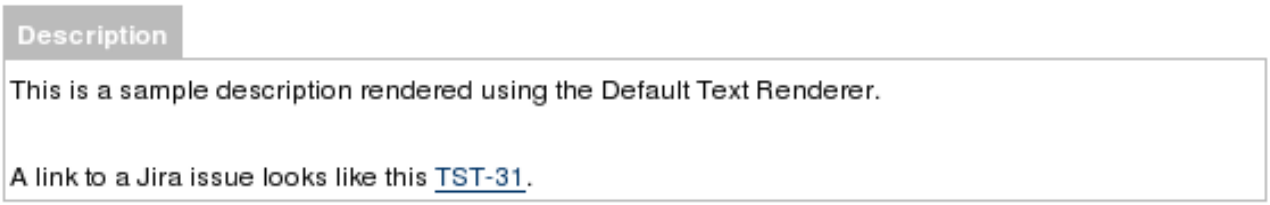
1.8.9.3. Renderer Types

JIRA version 3.4 and later ships with two renderers, the Atlassian Wiki Renderer and the Default

Text Renderer.

Default Text Renderer

The Default Text Renderer, as the name implies, is the default renderer for JIRA. Out of the box, JIRA is configured to use the text renderer for all renderable fields. The text renderer renders a field's content as plain text, with the following additional auto-linking feature: if the text contains text that resolves to a JIRA issue key then an HTML link will be generated that points to that issue. Below is a sample of how some description text looks when rendered through the Default Text Renderer.



Sample of description rendered with text renderer

Note:
It is not possible to disable the Default Text Renderer plugin as it is required for the system to function properly. If a field is setup to use a renderer that is later disabled, the field will revert to using the Default Text Renderer.

Atlassian Wiki Renderer

The Atlassian Wiki Renderer allows a user to enter wiki markup to produce html content, as described in 'Editing Rich-Text Fields' in the *JIRA User's Guide*.

This renderer uses the Confluence wiki renderer engine and therefore uses the Confluence wiki notation. The Confluence notation is easy to learn and allows for:

- Italic, bold and underlined text.
- Multiple levels of headings to organise your document.
- Bullets, numbering, tables and quotations.
- Images, screenshots, and emoticons.
- Powerful mini-applications using macros.

A full notation guide can be found here.

Note:
The Atlassian Wiki Renderer can only be used with JDK 1.4 and up. The renderer will not run on JDK 1.3.

Atlassian Wiki Renderer Macro Support

The Atlassian Wiki Renderer supports pluggable macros in the same way that Confluence does. Macros provide an easy and powerful extension point to the wiki markup language. JIRA ships with a number of macros as described in the *JIRA User's Guide*.

Note:
JIRA and Confluence can share macros, but keep in mind that many Confluence macros are very specific to the Confluence application and will therefore not run within JIRA. For example, the Children macro in Confluence shows links to all of a Page's child pages. JIRA has no concept of 'page' and therefore this macro will not function in JIRA.

1.8.9.4. Implications for Jira operations

The fact that JIRA allows you to configure different renderers across different projects/issue types for the same field has implications for bulk operations. Also, since the Atlassian Wiki Renderer inherently creates HTML as its end product, there are implications as to how this will behave when issue data is viewed outside JIRA's web front-end.

Bulk Move

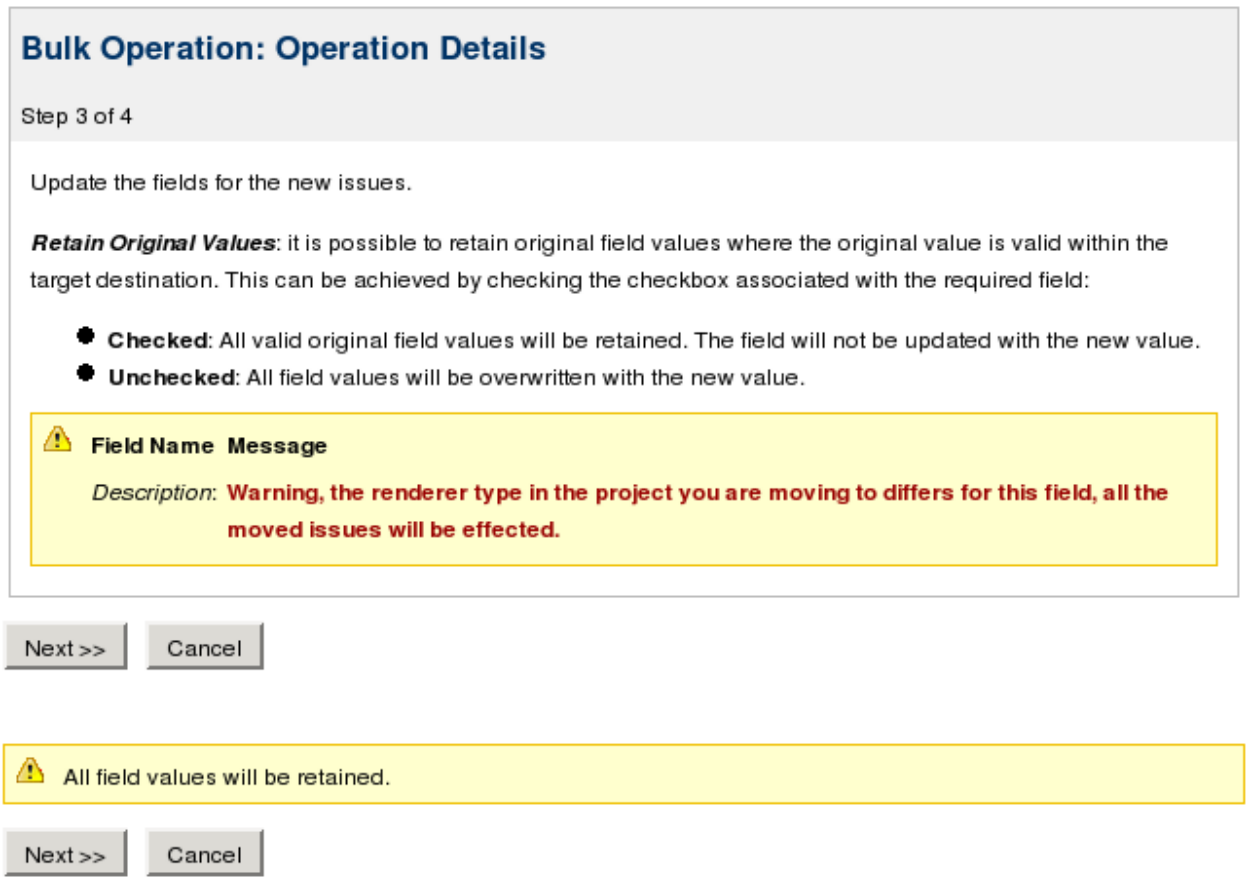
When performing a bulk move operation you can either move issues to an environment (project/issue type) where the renderer types for the fields are the same or where they will be different. If the renderer types for where you are moving to are the same then you will not notice any changes to the way the issues data is displayed once the move has occurred and the move operation will not prompt the user with any warnings.

When bulk moving issues to an environment (project/issue type) that has a different renderer type defined for one of the fields being affected by the move, if any of the issues have a non empty value associated with the field, the move operation will present the user with a warning so that you can be aware of the change. The warning does not affect the move operation in any way but it is there to alert you to the fact that the moved issues' affected fields may look different in their new project/issue type.

This is best illustrated with an example. Let's say you have project 'A' which is configured to use the Atlassian Wiki Renderer for the Description field. Let's say you also have a project 'B' which is configured to use the Default Text Renderer for the Description field. You have three issues that exist in project 'A' and you want to perform a bulk move of the three issues to project 'B'. If none of the issues in project 'A' have a value set for the Description field they will be moved and you will not notice any changes since there is no value to render. If one of the issues has the following value in its Description:

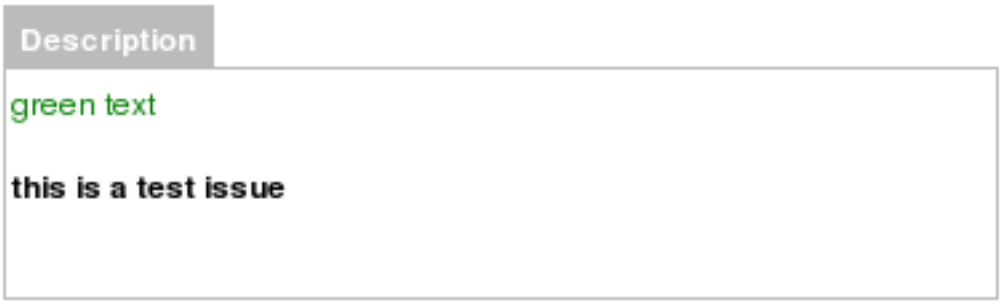
```
{color:green}green text{color}
*this is a test issue*
```

You would be presented with this screen in the bulk move to alert you that you are changing renderers as a result of the move:



Example of the bulk move renderer warning

The move operation does nothing to affect the data itself so after the move the wiki markup will display through the Default Text Renderer. In our example the before and after look like this:



Before

After

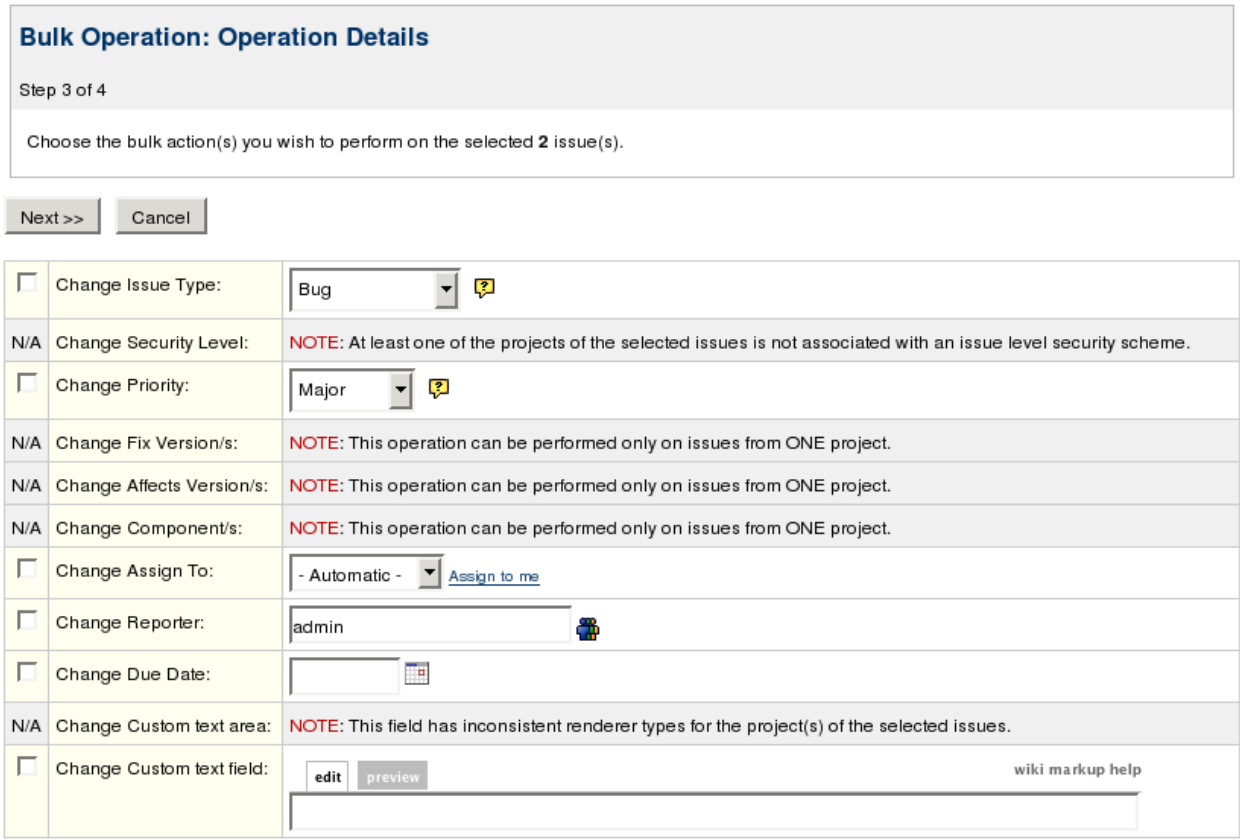
Bulk Edit

When performing a bulk edit operation the only renderable fields you may be able to bulk edit are instances of the Text Field, and Free Text Field (unlimited text) custom fields. The bulk edit

operation does not allow you to bulk edit the description, environment, or comment fields.

You will only be allowed to bulk edit a renderable field if all the issues selected for edit use the same renderer type. If the renderer type differs for any of the selected issues you will be presented with an error message.

This is best illustrated with an example. Let's say you have two global custom fields, 'Custom text area' and 'Custom text field', whose types are as their names imply. Let's say you have project 'A' which is configured to use the Atlassian Wiki Renderer for both of the fields. Lets say you also have a project 'B' which is configured to use the Default Text Renderer for the 'Custom text area' field and the Atlassian Wiki Renderer for the 'Custom text field'. Let's also say that you have one issue in each project. If you were to perform a bulk edit operation on the two issues in these projects you will be presented with the screenshot below:



Sample bulk edit screen

You will notice that for the 'Custom text area' field you are presented with an warning that the field has inconsistent renderer types and that it is not available to be selected for bulk edit. This is because the fields do not share the same renderer in the two issues. You will also notice that for the 'Custom text field' field you are presented with an editable input that allows for wiki preview. This is because the field shares the same renderer in the two issues.

Email Notifications

JIRA allows for extensive configuration in relation to email notifications. JIRA can be send out two types of emails, HTML and text (see 'Email Formatting').

HTML Emails

When using the Atlassian Wiki Renderer, the rendered content (i.e. exactly what you see on the 'View Issue' page) will be sent out in the emails. This will create emails which are as rich as the

content makes it. If using the Atlassian Wiki Renderer this is the preferred type of email since it is a real representation of the wiki markup.

Text Emails

When using the Atlassian Wiki Renderer, the actual wiki markup (unrendered) will be displayed in text emails for fields that use the Atlassian Wiki Renderer. This is obviously less readable than the rendered version of the markup, but because the markup's syntax is quite simple the text does remain easy to read.

Excel View

JIRA allows the Issue Navigator view to be exported to an Excel spreadsheet. If any of the fields being exported to Excel are using the Atlassian Wiki Renderer, the value exported to the cell in Excel will be the original wiki markup. Attempting to display complex HTML within a cell in Excel adds rows and columns that make using the data for formulas very difficult.

Note:
The unrendered wiki markup will be shown in Excel cells for fields that use the wiki renderer.

RSS/XML View

JIRA allows the Issue Navigator view to be exported to RSS/XML. If a field is using the Default Text Renderer its values will be exported in a CDATA section within the generated XML. If a field is using the Atlassian Wiki Renderer, its rendered value will be XML escaped and included in the generated XML. If the XML view is being used as an RSS feed, most RSS readers will render the generated HTML so you will see the rich content within your RSS reader.

If you would like to have this view feed out the raw values (unrendered) then you can send an additional request parameter 'rssMode=raw'. If the original link looks like this:

```
http://localhost:8080/browse/AAA-1?decorator=none&view=rss
```

Then the URL to have the raw values placed inside a CDATA should look like this:

```
http://localhost:8080/browse/AAA-1?decorator=none&view=rss&rssMode
```

Other

This section describes other issues to be aware of in relation to the renderers.

- When editing a renderable custom field's default value, even if it is only ever configured to use the Atlassian Wiki Renderer you will not be presented with the 'Edit' and 'Preview' tabs. Unfortunately it is not possible, in that context, to tell which renderer should be used for editing. This said, if you enter a default value using wiki markup then this will render correctly in environments (project/issue type) where the field has been configured to use the Atlassian Wiki Renderer.

1.8.9.5. Configuring Renderers

Applying a Renderer to a Field

To enable a renderer for a particular field, edit the Field Configuration and choose the appropriate renderer for the field. For details, see 'Configuring field behaviour.

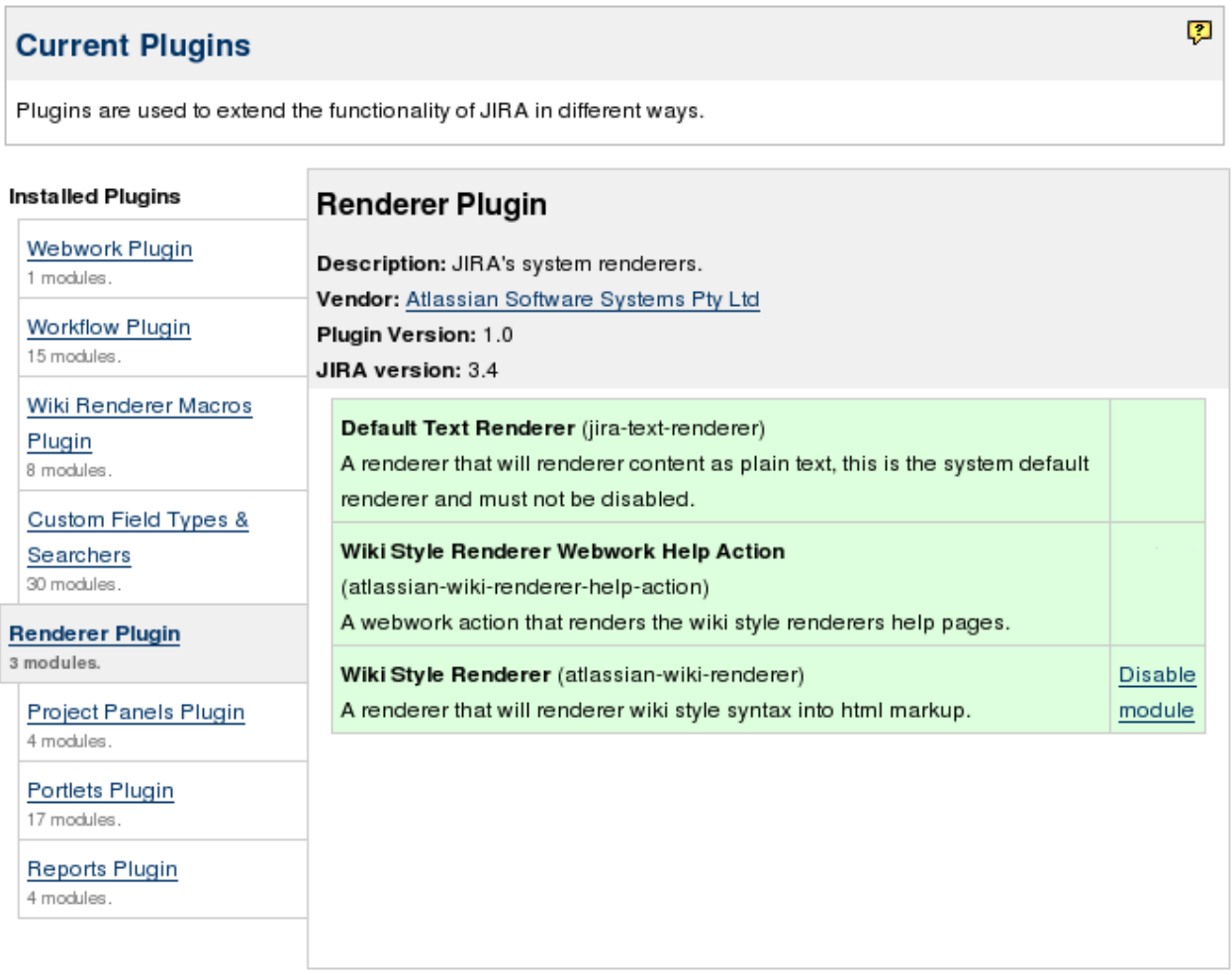
Enabling a Renderer Plugin

Renderers within JIRA are implemented as JIRA plugins. The macros that the Atlassian wiki renderer uses are also implemented as JIRA plugins. For general information on plugins please see this guide.

> **Note:**
> Plugins are configured at a site-wide level — it is not possible to configure plugins at a project/issue type level.

Renderer Plugins Configuration

Renderers and their dependant components, except for the default text renderer, can be enabled/disabled via the plugin administration menus. If you navigate, as an administrator, to 'Administration' > 'Plugins' and then click on the option 'Renderer Plugin' you will see the following screen.

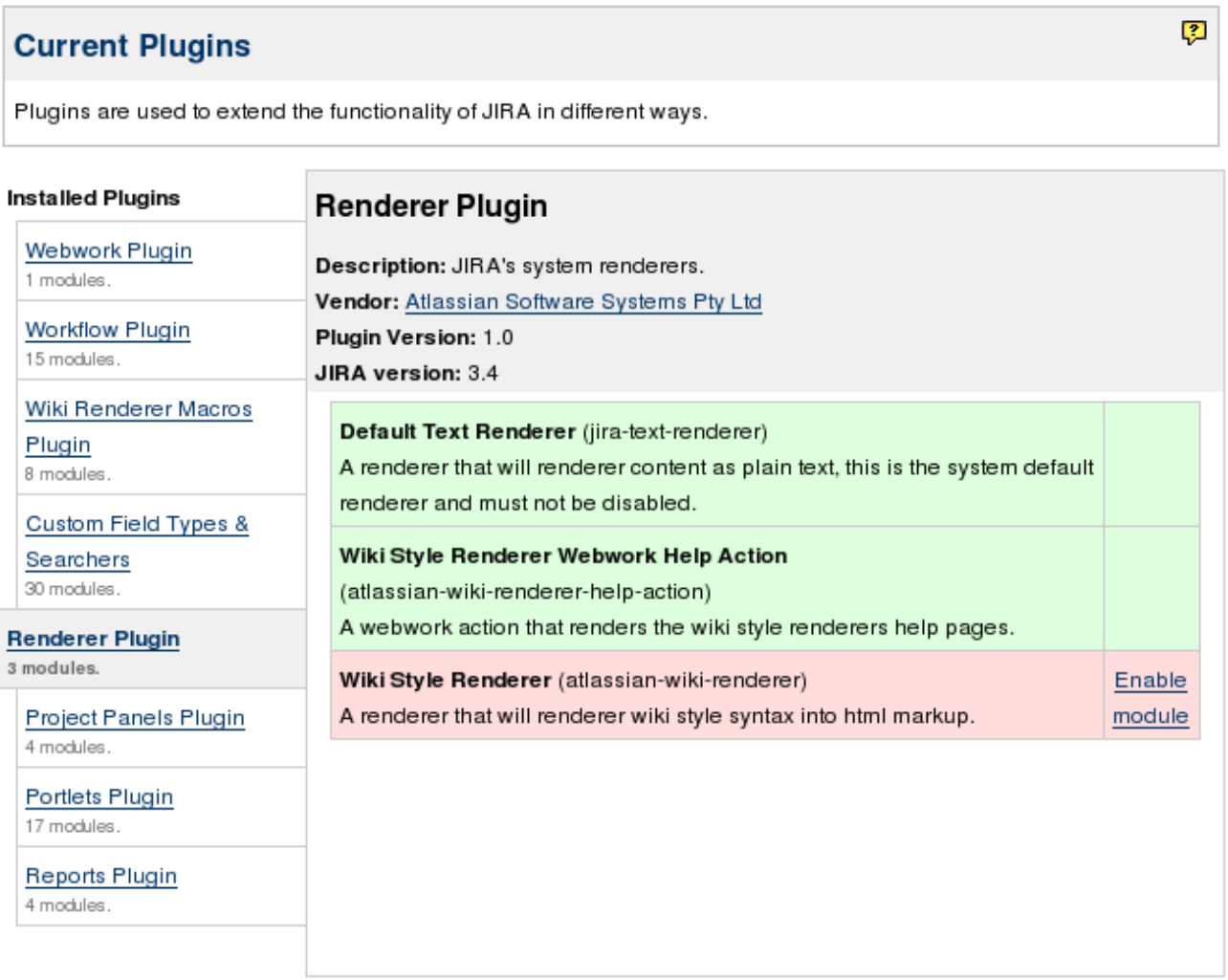


Renderer plugin configuration screen

> **Note:**
> The plugin titled 'Wiki Style Renderer Webwork Help Action' is a front-end helper for showing the Atlassian wiki renderer notation guide and it can not be disabled.

From this screen you will see all the configured Renderers within JIRA. At the moment only two renderers exist but if more are created you will see there configuration here. If you click on the 'Disable Module' link for the 'Wiki Style Renderer' this will deactivate the renderer for the entire

instance of JIRA.



Renderer plugin disabled

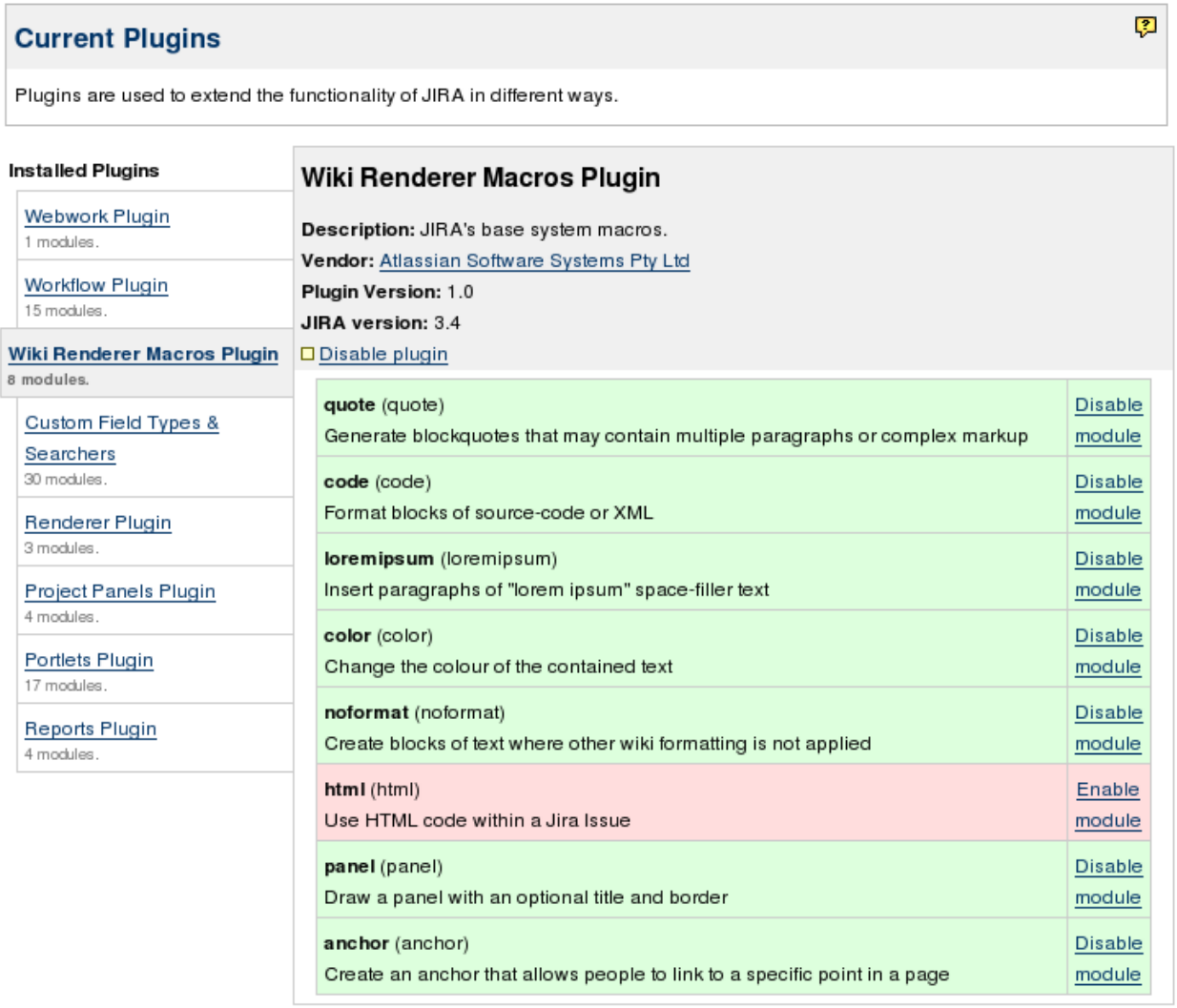
Any fields that are still setup to use the disabled renderer will fall back to the default text renderer and when you attempt to edit the field a warning message will alert you to the fact that you are configured to use a renderer that is not available.

No such renderer available

When a renderer is disabled it will not be available for selection when changing a fields renderer. To enable the renderer just click the 'Enable Module' link. Enabling/Disabling a renderer has no effect on the renderer settings in the field configurations so it is possible to disable and then re-enable a renderer without effecting any data.

Macro Plugins Configuration — Atlassian Wiki Renderer

The macros used by the Atlassian wiki renderer can be enabled/disabled via the plugin administration menus. If you navigate, as an administrator, to 'Administration' > 'Plugins' and then click on the option 'Wiki Renderer Macros Plugin' you will see the following screen.



Atlassian wiki renderer macros plugin configuration screen

From this screen you will see all the configured macros within JIRA. If a macro is disabled then it will not be available to the wiki renderer, likewise a macro must be enabled for it to be available to the wiki renderer. If you deploy any additional macros that you wish to use, they must be enabled here to be available to the wiki renderer. For more information on writing plugins please see this guide.

1.8.10. Associating Field Behaviour with Issue Types

1.8.10.1. What is a 'Field Configuration Scheme' ?

In the *Enterprise* edition of JIRA it is possible to define Field Configurations for a particular issue type of a given project using Field Configuration Schemes. Field Configuration Schemes map field configurations to issue types. Once created, a Field Configuration Scheme can be associated with one or more projects. This means that, for example, it is possible to have a separate field configuration for the 'Bug' issue type and the 'Improvement' issue type for the 'Test' Project.

Please note that field configuration schemes can be associated with more than one project, allowing administrators to reuse the same field configuration for issue type mappings across projects.

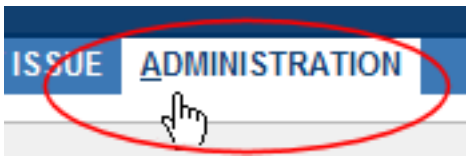
Note:

For information about placing fields on Screens, and using Screen Schemes to associate screens with issue operations, please see the overview.

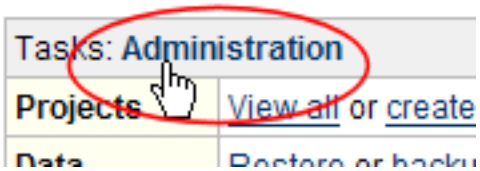
1.8.10.2. Field Configuration Schemes

Scheme operations are available from the View Field Configuration Scheme page. To reach this page follow these steps:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

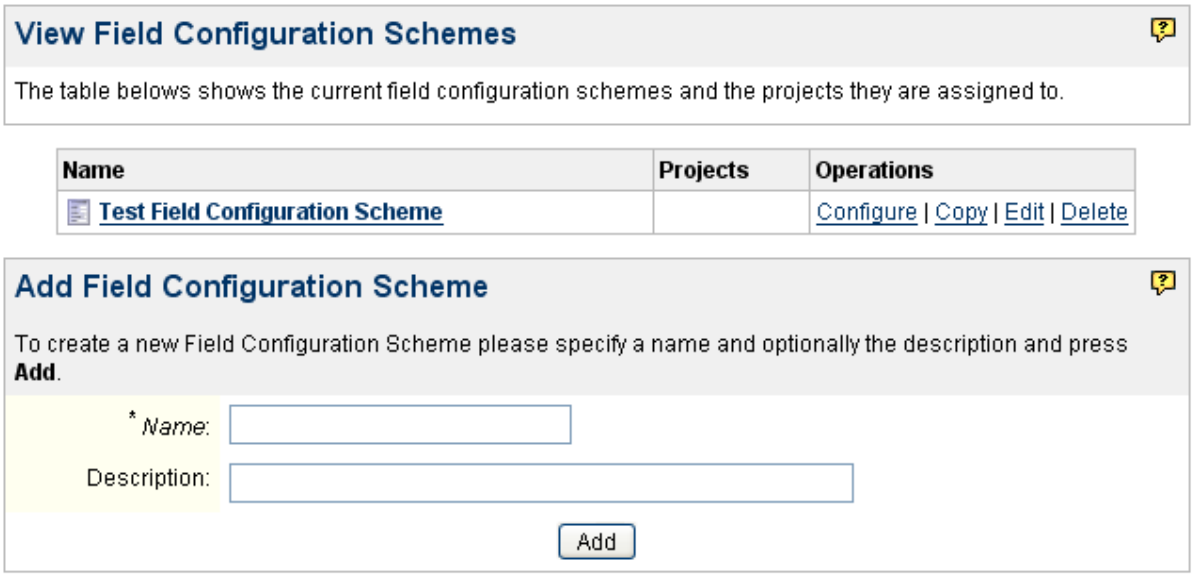
3. Click the "Issue Fields" sub-menu on the left hand side if it is not open already, and choose "Field Configuration Schemes" from the list.
4. You will be directed to the page "View Field Configuration Schemes" with a list of all the Field Configuration Schemes currently configured. From this page you can:
   - Add - create a new Field Configuration Scheme
   - Configure - add or remove associations between issue types and Field Configurations*
   - Edit - edit the name and description of the Field Configuration Scheme.
   - Delete - remove a Field Configuration Scheme
   - Copy - create a new Field Configuration Scheme with the same details as an existing one

Note:

*To create and edit Field Configurations, please see 'Configuring field behaviour'. To create and edit Issue Types, please see 'Defining 'Issue Type' field values'.

1.8.10.3. Adding a Field Configuration Scheme

1. The "Add New Field Configuration Scheme" form is located at the bottom of the "View Field Configuration Scheme" page
2. On this form enter the name of the new scheme.
3. You can optionally add a description to the scheme for better identification.
4. Click on the "Add" button. The Field Configuration Scheme list will be updated automatically with the new scheme.



View Field Configuration Scheme

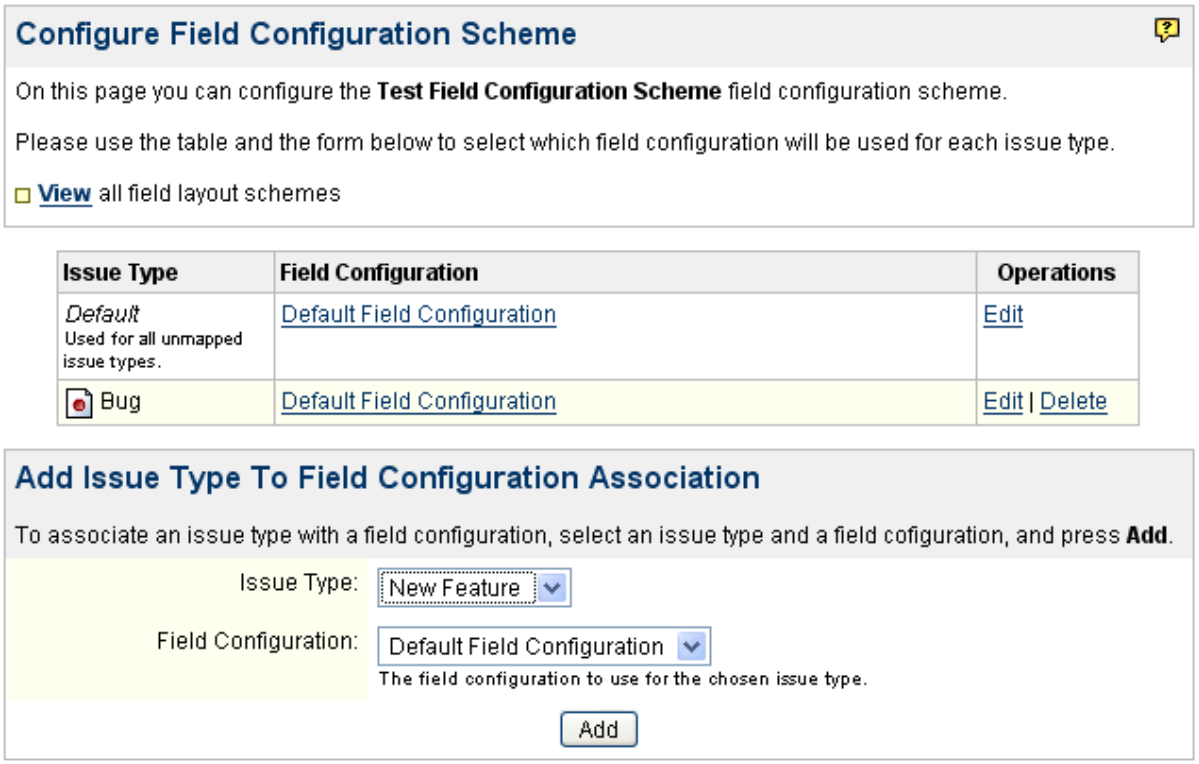
1.8.10.4. Configure a Field Configuration Scheme

With this operation you can associate, un-associate or change an association between a Field Configuration and a particular issue type. To do this please follow these steps:

1. Click on the "Configure" link in the same row as the Field Configuration Scheme you wish to configure.
2. The "Configure Field Configuration Scheme" page will appear, showing the scheme's current mappings of Field Configurations to issue types.
3. The operations available when configuring a field configuration scheme are:
   - Associate an issue type to a field configuration
   - Remove an association between an issue type and a field configuration
   - Edit an association between an issue type and a field configuration

Note:

If you have not added any field configurations you will only have the Default Field Configuration to work with.



Configure Field Configuration Scheme

Associating a field configuration with an issue type

To associate an issue type with a field configuration:

1. Select the issue type you wish to associate.
2. Select the field configuration you wish to associate with this issue type
3. Click on the "Add" button. The table will above will be automatically updated with the the issue type in the left most column and the associated field configuration in the central column.

Note:
An issue type can only have one association within a given configuration scheme.

Note:
If an issue type does not have an association in the scheme, the field configuration associated with the *Default* entry in the scheme will be used for issues of that type.

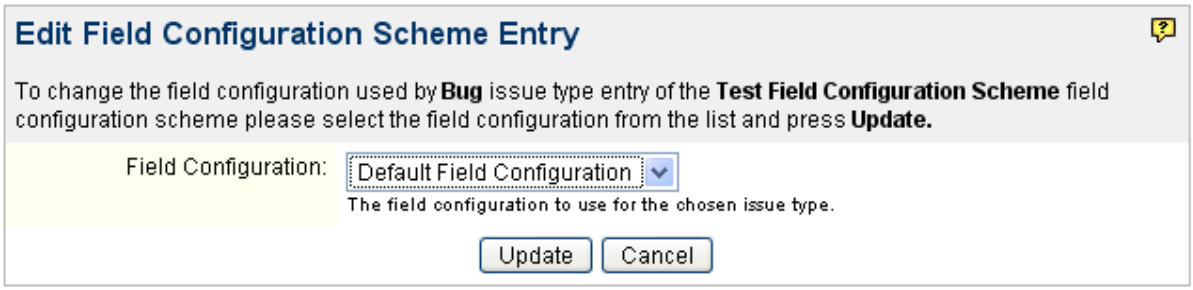
Removing an association between a field configuration and an issue type

1. Click on the "Remove" link next to the issue.
2. The issue type association will automatically be removed from the field configuration scheme.

Note:
The *Default* entry cannot be removed the scheme.

Editing an association between a field configuration and an issue type

1. Click on the "Edit" link next to the issue
2. Select the new field configuration you would like to associate with this issue type
3. Click on the "Update" button
4. The Issue Type will now be associated with the new field configuration.



Edit association between field configuration and issue type.

1.8.10.5. Editing a Field Configuration Scheme

To change the name or description of a Field Configuration Scheme:

1. Click on the "Edit" link next to the desired field configuration scheme. You will now see the "Edit Field Configuration Scheme" page.
2. Change the name and/ord description as necessary.
3. Click on the "Update" button.

1.8.10.6. Deleting a Field Configuration Scheme

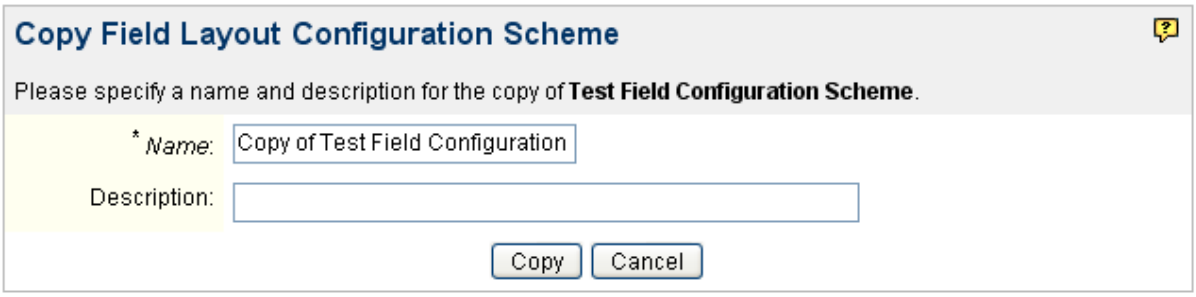
To delete a Field Configuration Scheme:

1. Click on the "Delete" link next to the desired field configuration scheme. You will now see the "Delete Field Configuration Scheme" page.
2. Confirm that you would like to delete the scheme by clicking the "Delete" button.

1.8.10.7. Copying a Field Configuration Scheme

To copy a Field Configuration Scheme:

1. Click on the "Copy" link next to the field configuration scheme you wish to copy. This will bring you to the "Copy Field Configuration Scheme" page.
2. Enter the name and description of the new field configuration scheme.
3. Click on the "Copy" button.
4. You will now be directed back the View Field Configuration Scheme page with your new scheme added. The new scheme will have the same configuration as the copied scheme.



Copy Field Configuration Scheme

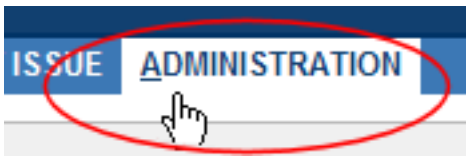
1.8.10.8. Associating a Field Configuration Scheme with a Project

To activate a Field Configuration Scheme, you need to associate it with a project. An association means that the Field Configuration Scheme will now be applied to the chosen project. The issues in that project will use the Field Configuration that is mapped to their issue type by the scheme. Note also that you can use Issue Type Schemes to associate issue types with a project.

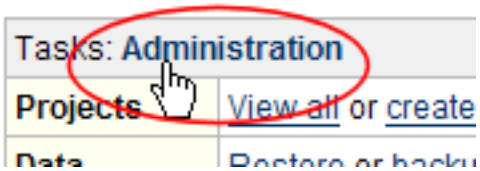
To associate a Field Configuration Scheme with a project:

1. Log in as a user with the '**JIRA Administrators**' global permission.

2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

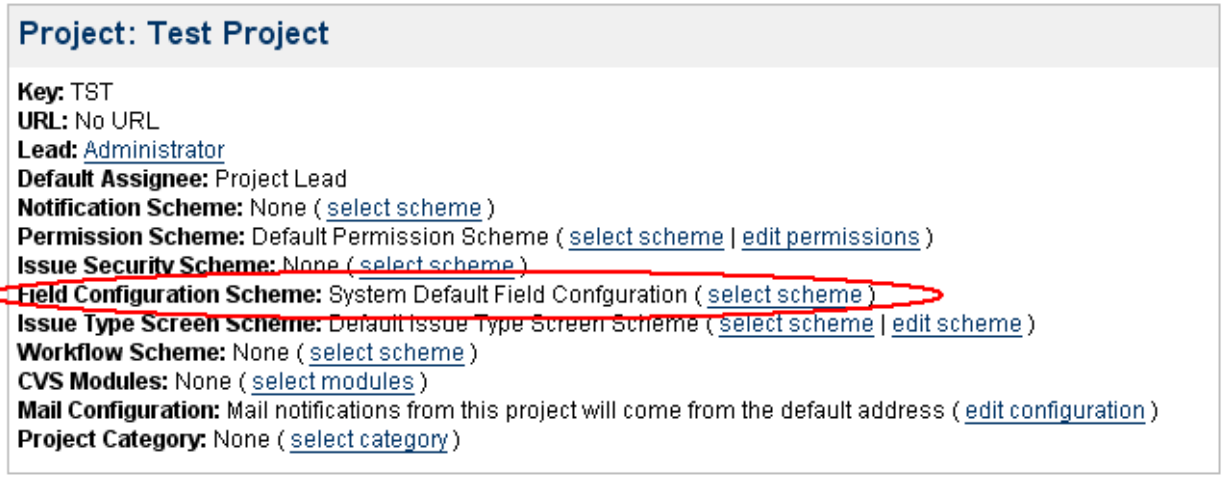


Link to Administration section



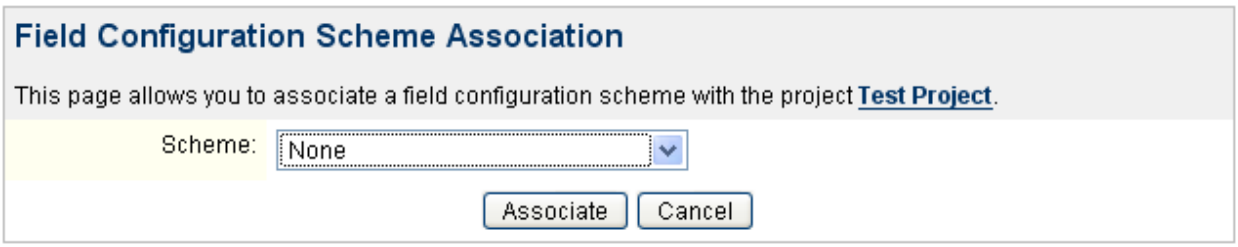
Link to Administration section

3. On the administration page is a list of projects which this user is allowed to manage. Select the project of interest.
4. Click on the "select scheme" link beside the Field Configuration Scheme caption.



Project Admin Page

5. This will bring up a list of all existing field configuration schemes. Select the scheme you want to associate with this project.



Associate Field Configuration Scheme to Project

Note:

Selecting *None* will make all the issues in the project use the Default Field Configuration.

6. Click on the "Associate" button. You will be returned to the project administration page, with the project now associated with the selected Field Configuration Scheme.

Note:

As mentioned above, new projects are not associated with any Field Configuration Schemes and hence use the Default Field Configuration for all issues.

1.8.11. Placing fields on screens

1.8.11.1. What is a 'Screen'?

Screens group multiple issue fields. Using Screens, administrators can control which fields are displayed, and the fields' vertical display order, during issue operations (e.g. 'Create Issue' and 'Edit Issue') or workflow transitions (e.g. 'Resolve Issue') — for details see 'Activating Screens' (below).

In the *Enterprise* edition of JIRA it is also possible to split issue fields on a Screen into multiple tabs.

Screens overlap slightly with Field Configurations in regards to field visibility. Note that when a Screen is displayed to a user, for example, during issue creation, the user will see only the issue fields that:

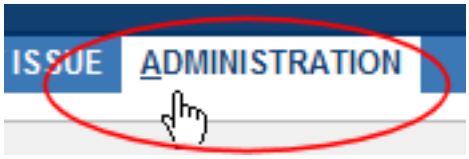
1. the user has permissions to edit.
2. are present on the Screen that is associated with the 'Create Issue' operation for this issue.
3. are not hidden in the Field Configuration applicable to the issue.

Note:

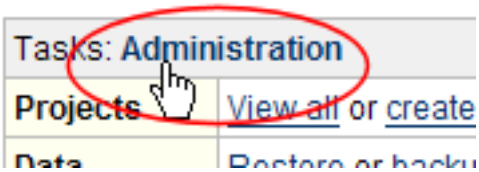
A field may be present on a Screen, but if it is hidden in an appropriate Field Configuration, it will not be visible to the user when the Screen is displayed. Note also that, if a particular field needs to be hidden at all times, it is simpler to hide the field in an applicable Field Configuration rather than remove it from all Screens. For more information please see the Overview.

1.8.11.2. Configuring a Screen's Fields

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

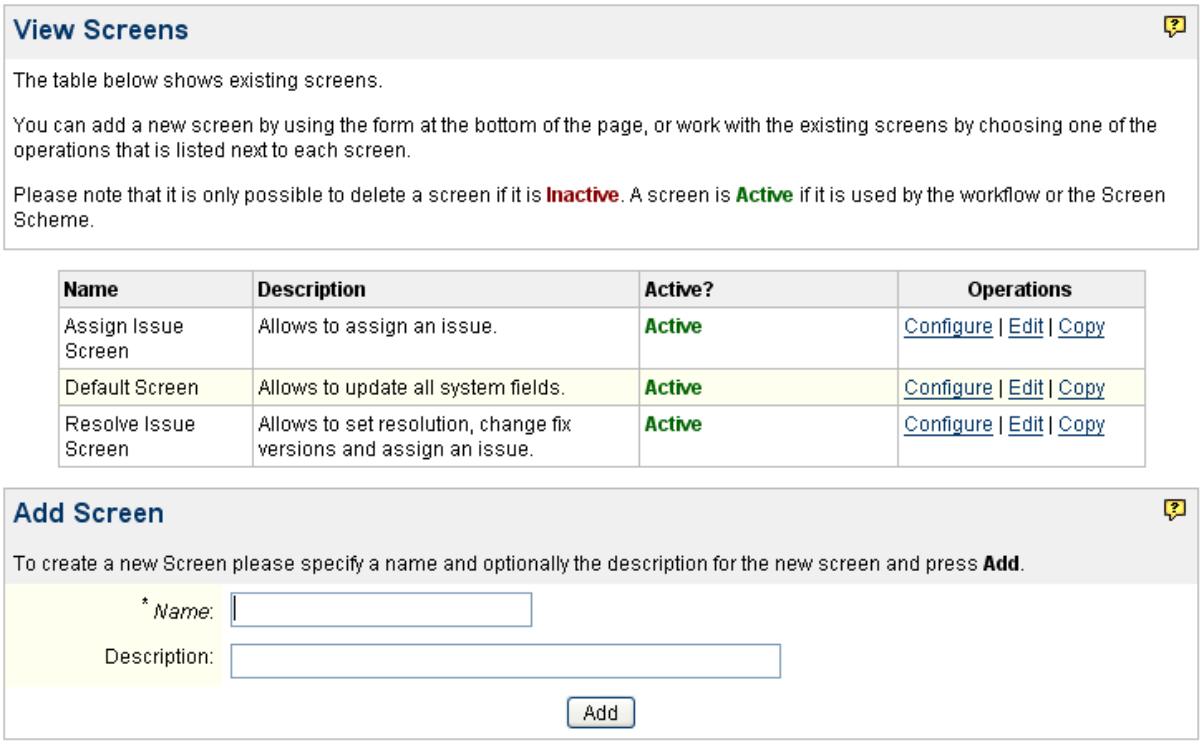


Link to Administration section



Link to Administration section

3. Click the "Issue Fields" sub-menu in the left-hand side column, and choose "Screens" from the list.
4. You will then be directed to the "View Screens" page. (This screenshot is for *Standard* edition. *Professional* and *Enterprise* editions will have a different layout but will includes the same functionality.)

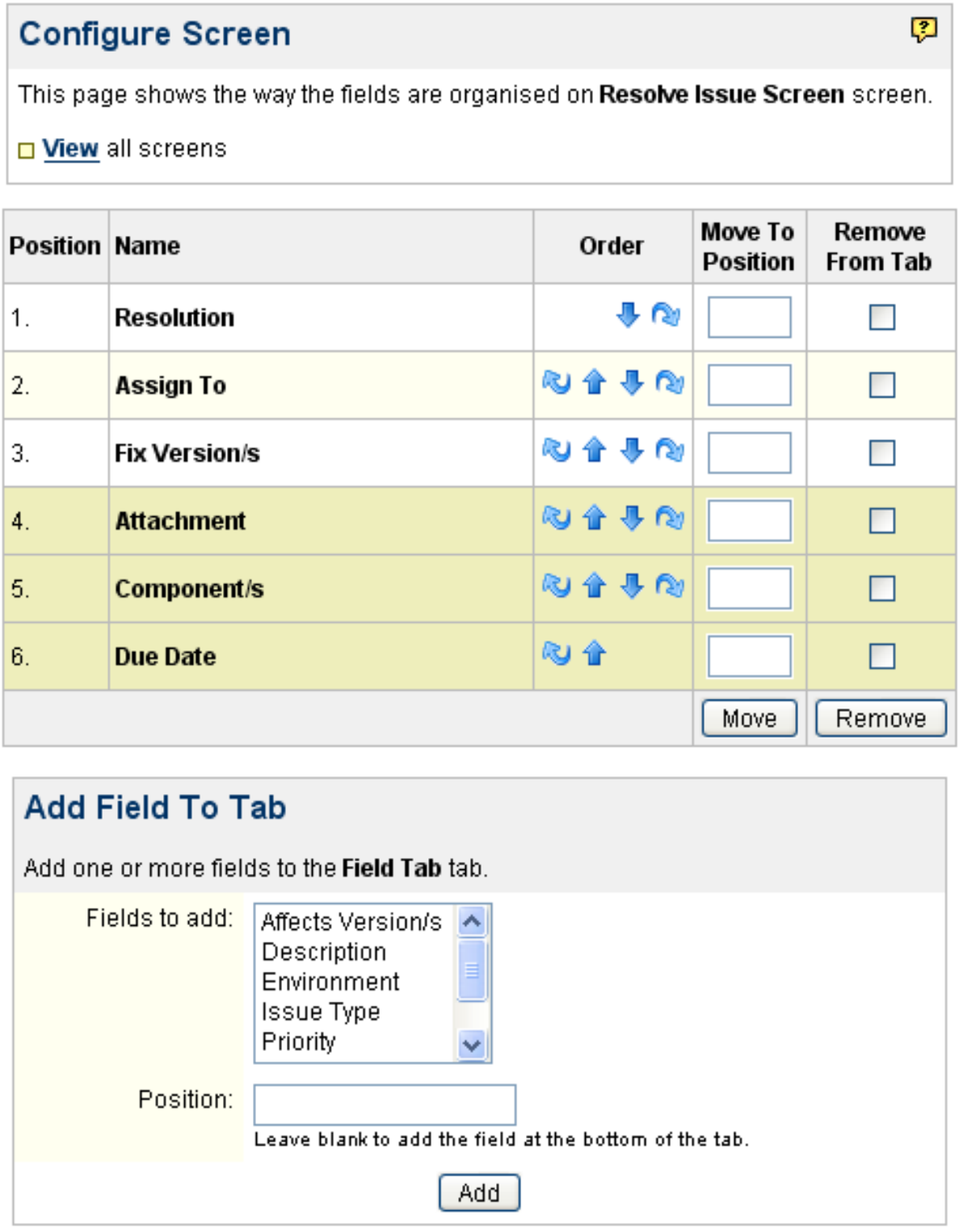


View Screens page

From this page you can:

- Add a new screen.
- Configure a screen's fields
- Edit a screen's name and/or description
- Delete a screen.
- Copy a screen.

1. Click on the "Configure" link next to the Screen of interest.
2. You will now see the "Configure Screen" page. (Note: the screenshot below is common between *Standard* and *Professional* editions; the *Enterprise* edition page includes the tabs functionality — see below).



Configure Screen Page

Adding a Field to Screen

1. The "Add Field" form is located at the bottom of the "Configure Screen" page.
2. Select the field/s that you wish to add to the screen.
3. You can also specify the position in which the field will be placed.

Note:

If you have selected multiple fields and specified a position, the topmost field selected will be placed in the corresponding position and the other fields directly below it.

4. Click on the "Add" button"

Removing a Field from a Screen

1. From the "Configure Screen" page, select the checkboxes next the the fields you wish to remove.
2. Click on the "Remove" button located at the bottom of the table.
3. The fields will be removed from the Screen and will become available in the "Add Field" form at the bottom of the screen.

Reordering Fields on a Screen

To change the vertical display order of fields:

1. In the text box in the "Move to Position" column next to the desired field, specify the position you wish to move the field to.
2. You can repeat this for multiple fields specifying a different position for each field.
3. Click on the "Move" button located at the bottom of the table in the "Move to Position" column.
4. All the fields will be placed in the specified positions.

> **Note:**
> Alternatively, you can click on the arrows next to the desired field to move the field up, down, to the first position or to the last position.

1.8.11.3. Adding a Screen

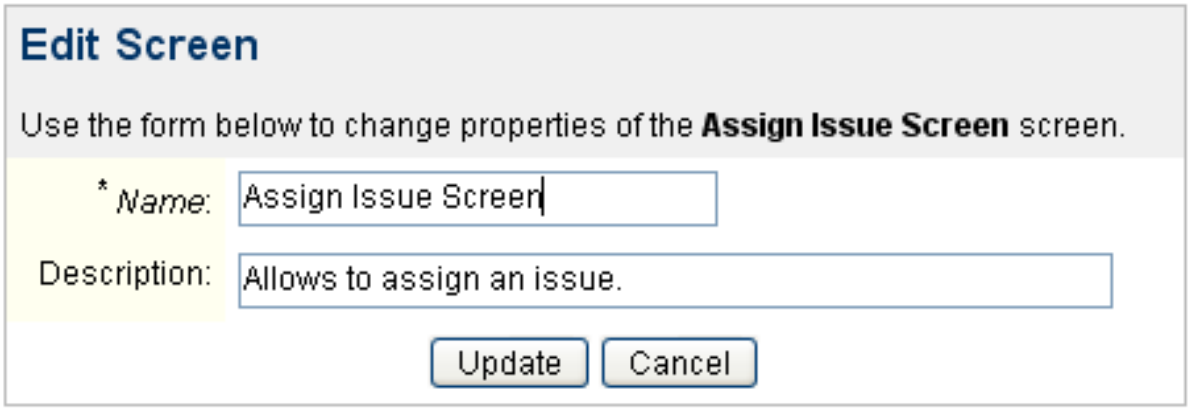
1. The "Add Screen" form is located at the bottom of the "View Screens" page (see 'Configuring a Screen's Fields', above).
2. Enter the name of the new Screen.
3. You can optionally add a Description.
4. Click on the "Add" button. The page will automatically update the Screen list with the new Screen.

> **Note:**
> A newly created Screen is not usable until it has been associated with either an issue operation (via a Screen Scheme) or a workflow transition. See 'Activating Screens' (below).

1.8.11.4. Editing a Screen's Details

To change Screen's name and/or description you can edit a Screen by following these steps:

1. On the "View Screens" page (see 'Configuring a Screen's Fields', above), click on the "Edit" link next to the appropriate screen.
2. You will now be directed to the "Edit Screen" page where you can edit the name and/or description of the Screen.
3. Click on the "Update" button. You will be brought back to the "View Screens" page with your updates now applied to the Screen.

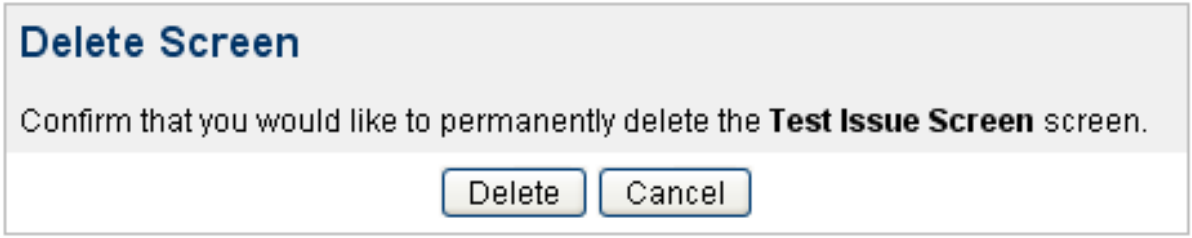


Edit Screen Page

1.8.11.5. Deleting a Screen

To entirely remove a Screen from the system:

1. On the "View Screens" page (see 'Configuring a Screen's Fields', above), click on the "Delete" link next to the Screen you wish to delete.
2. Click on the "Delete" button to confirm this action. You will be brought back to the "View Screens" page with the Screen removed from the list of Screens.



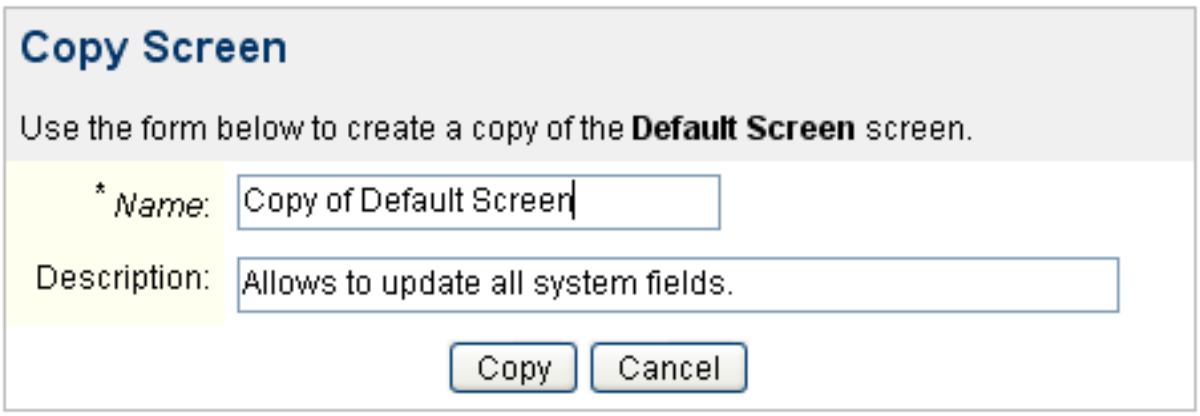
Delete Screen Page

Note:

Field screens that are associated with at least one Screen Scheme or at least one workflow transition cannot be deleted.

1.8.11.6. Copying a Screen

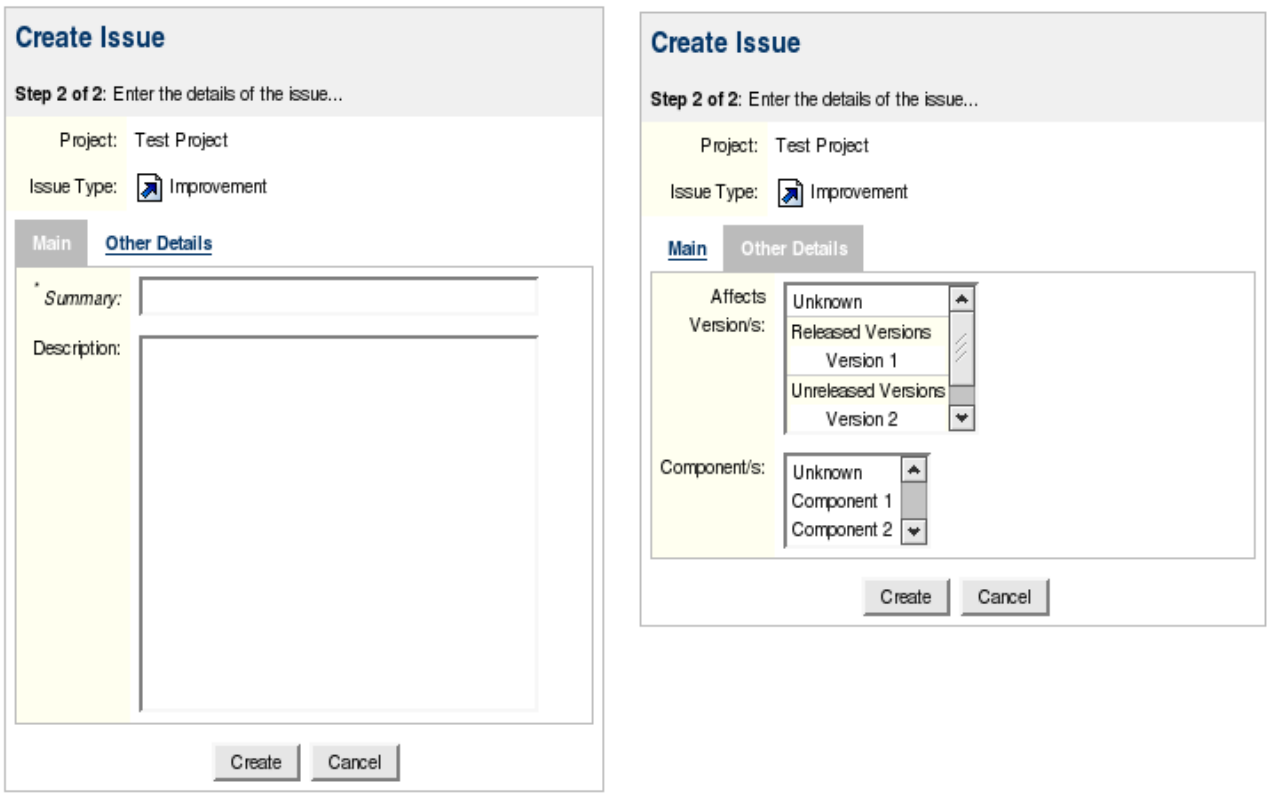
1. On the "View Screens" page (see 'Configuring a Screen's Fields', above), click on the "Copy" link next to the Screen you wish to copy. You will now be directed to the "Copy Screen" page.
2. Enter a name and a description for the new Screen.
3. Click on the "Copy" button. You will be brought back to the "View Screens" page, and the newly added Screen will have the same issue fields and field positions as the original field screen.



Copy Screen Page

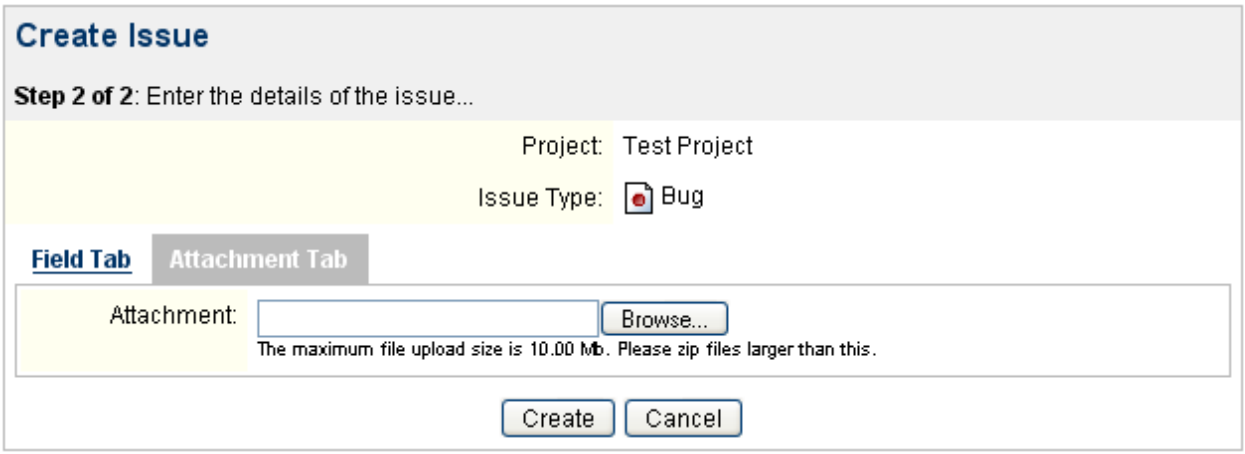
1.8.11.7. Configuring Tabs

The *Enterprise* edition allows you to split Screens into multiple tabs. Tabs can help to group related fields together on a Screen. For example, the following screenshot shows a simple Screen that only shows the issue 'Summary' and 'Description' on the first tab ('Main'), and 'Affected Versions' and 'Components' on the second tab ('Other Details'):



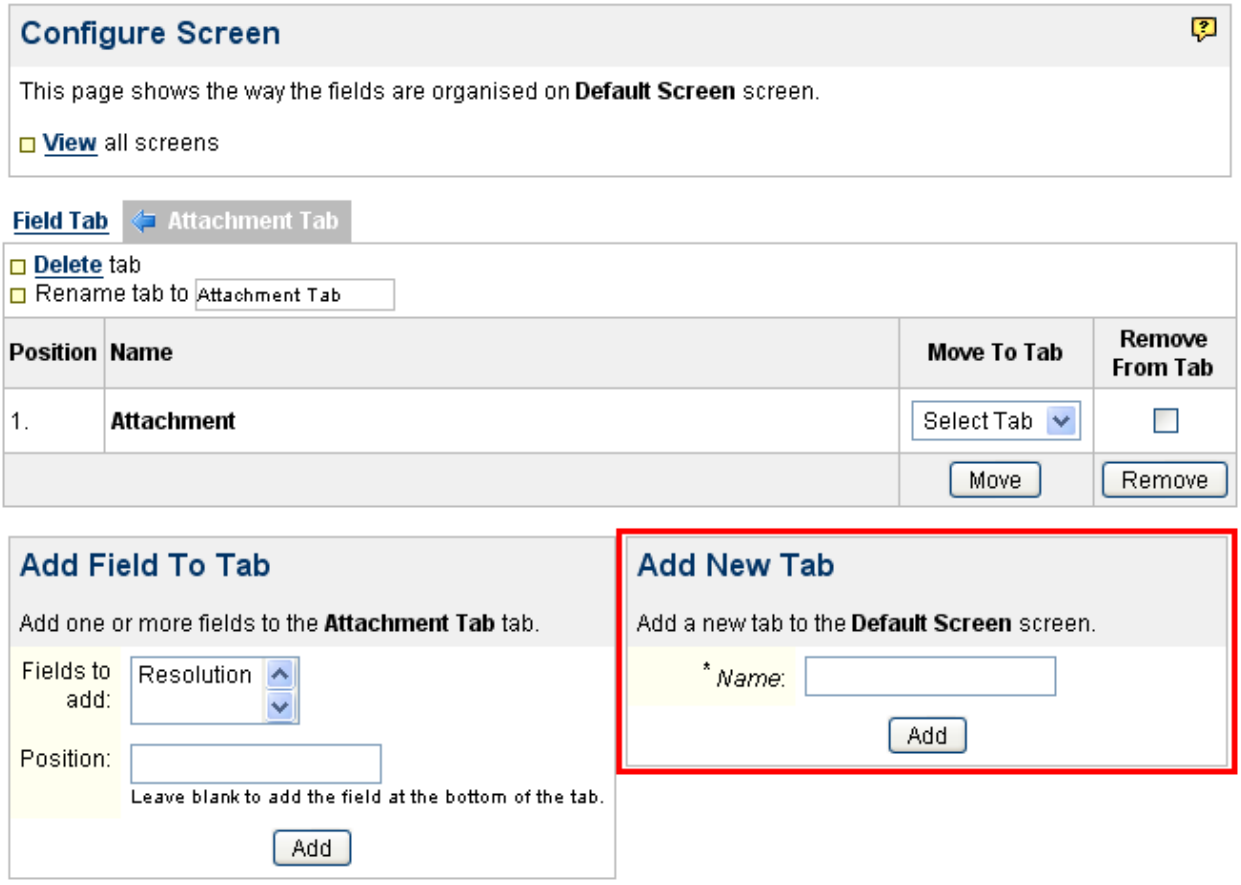
Sample Screen

This functionality is very useful for organising complex Screens, as you can place less used fields, for example, 'Attachment' and 'Environment', onto separate tabs:



Example of a field tab

Screen tabs are available from the "Configure Screens" page (see 'Configuring a Screen's Fields', above).



Configure Screen Page with Field Tabs

Add Tab

1. The "Add Tab" is located on the bottom right of the "Configure Screen" page.
2. Enter the name of the new tab on this form.
3. Click on the "Add" button.

Moving fields between Tabs

1. In the "Move to Tab" column next to the field you wish to move, select the desired tab.
2. Repeat this for all the fields you wish to move.

3. Click on the "Move" button located at the bottom of the table in the "Move to Tab" column.
4. All the selected fields will be moved to the appropriate tabs.

Note:

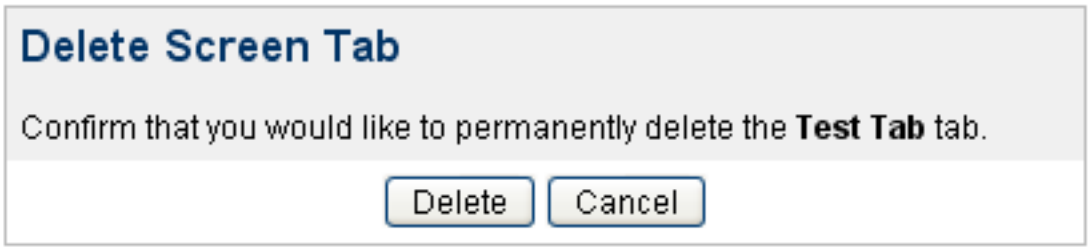
Please note that the system fields on the default View Issue screen (e.g. Summary, Security Level, Issue Type, etc.) are fixed and cannot be moved on to a separate tab. However, any custom fields that have been added to the View Issue screen can be moved on to a separate tab.
This restriction only applies to the screen associated with the View Issue operation, i.e. system fields can be moved on to other tabs for screens associated with operations such as Create Issue, Edit Issue, etc.

Navigating between Tabs

To navigate between the Tabs of a Screen, simply click on the links on the top left of the "Configure Screen" form.

Deleting a Tab

1. Navigate to the Tab you wish to remove.
2. Click on the "Delete" tab link. You will now be directed to the "Delete Tab" confirmation page.
3. Click on the "Delete" button to confirm. You will now be returned to the "Configure Screen" Page



Delete Tab Page

Renaming a Tab

1. Navigate to the desired Tab.
2. The "Rename" text field is located in the top left of the "Configure Screen Tab" form.
3. Enter the new name of the Tab.
4. Press 'Enter'.

Reordering Tabs

It is possible to configure the horizontal order of Tabs by clicking on the arrows to move the selected Tab left or right.

1.8.11.8. Activating a Screen

To make a Screen available to users, you can either:

- Associate the Screen with an **issue operation** (e.g. 'Create Issue'), via a **Screen Scheme** — see 'Associating Screens with Issue Operations'.
- (*Professional and Enterprise editions*) Associate the Screen with a **Workflow Transition** (e.g. 'Resolve Issue') — see 'Configuring Workflow'. Note that in *Standard edition* workflow is not customisable, so you cannot associate a new Screen with a workflow transition; you can, however, modify (see above) the Screen that is already associated with each workflow transition.

1.8.12. Associating Screens with Issue Operations

1.8.12.1. What is a 'Screen Scheme'?

A **Screen Scheme** allows you to choose which Screen will be shown to a JIRA user when they perform a particular *issue operation*. There are 3 issue operations for which you can choose a Screen:

- 'Create Issue' — choose the Screen that is shown when an issue is being created.
- 'Edit Issue' — choose the Screen that is shown when an issue is edited.
- 'View Issue' — choose the Screen that is shown when a user views an issue.

You can specify the same screen for each of these issue operations, or choose different screens for each operation.

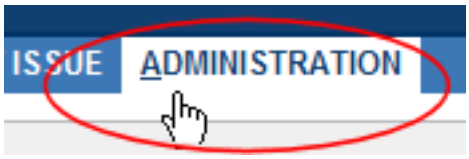
Please note that in the *Professional* and *Enterprise editions* of JIRA you can create multiple Screen Schemes (see Managing Multiple Screen Schemes below); in the **Standard edition** you can only edit the Default Screen Scheme. Additionally, *Professional edition* supports project-specific Screen Schemes, while *Enterprise edition* adds extra flexibility by supporting Screen Schemes per project per issue type, using Issue Type Screen Schemes. The method of activating a Screen Scheme also differs depending on the JIRA edition — please see 'Activating a Screen Scheme' (below).

Note:

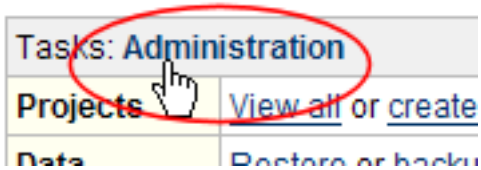
For more information about configuring Fields and Screens, please see the Overview.

1.8.12.2. Configuring a Screen Scheme

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

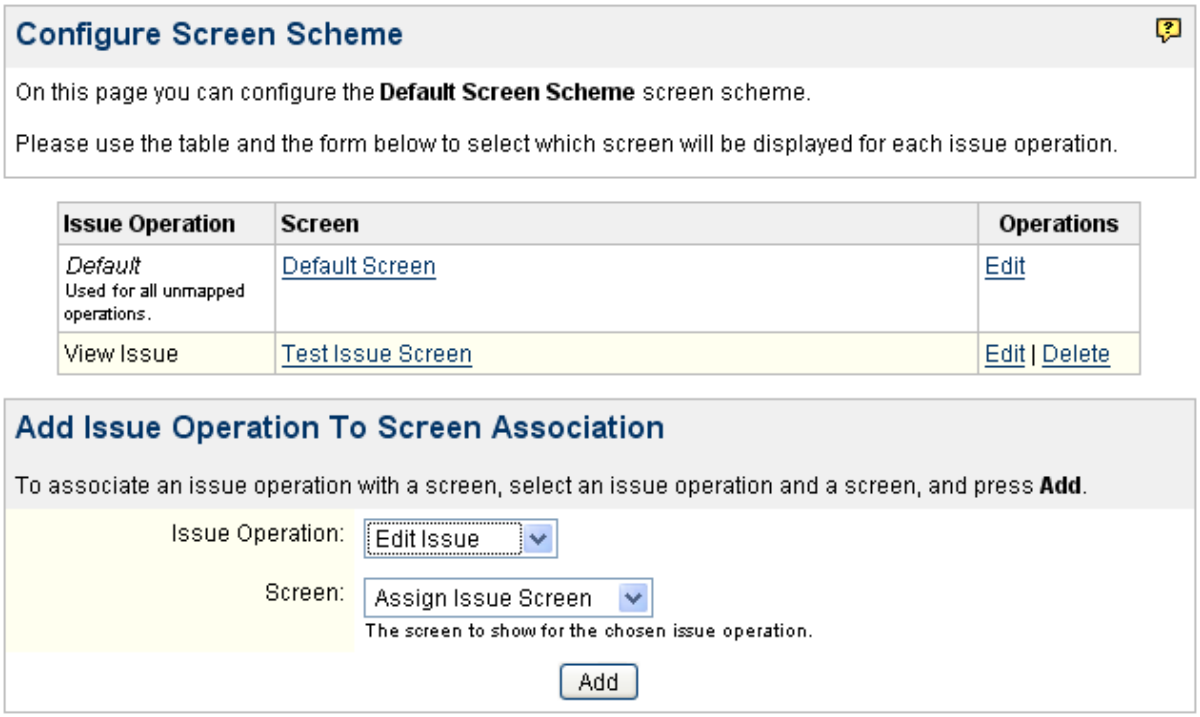


Link to Administration section



Link to Administration section

3. Click the "Issue Fields" sub-menu on the left hand side if it is not open already, and choose "Screen Schemes" from the list.
4. If you are using the *Standard Edition* you will be brought to the Configure screen for the Default Field Screen. If you are using the *Professional* or *Enterprise Editions* you will be brought to the "View Screen Schemes" page, where you will need to select a Screen Scheme by clicking on the "Screens" link next to one of the schemes.
5. You will now be brought to the "Configure Screen Scheme" page.



Configure Field Screen Scheme page

Associating an Issue Operation with a Screen

1. The "Add Issue Operation to Screen Association" is located at the bottom of the "Configure Screen Scheme" page.
2. Select the Issue Operation with which you wish to associate a Screen.
3. Select the desired Screen.
4. Click the "Add" button and the new association will be added to the list of associations.

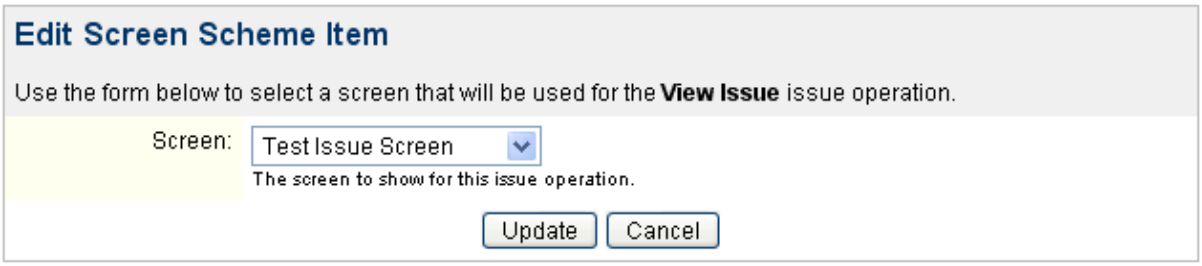
Note:
The "View Issue" operation only allows one to control the layout of **custom fields** in the middle of the "View Issue" page. The "View Issue" page ignores all the non-custom fields on the Screen.

Note:
There can only be one association for an issue operation per Screen Scheme. If all operations have been associated with a Screen, use the "Edit" link next to each operation to change the Screen it is associated with.

Note:
If an issue operation does not have a specific mapping to a Screen, the screen that is associated with the *Default* entry will be used for that operation. The *Default* entry cannot be deleted from a Screen Scheme. You can use the "Edit" link next to the *Default* entry to change the Screen that it associated with it.

Editing an Association

1. Click on the "Edit" link next to the issue operation you wish to edit.
2. You will be brought to the "Edit Screen Scheme Item" page.
3. Select the screen you wish to change the association to.
4. Click on the "Update" button and you will be returned to the screen scheme page.



Edit Screen Scheme Item Page

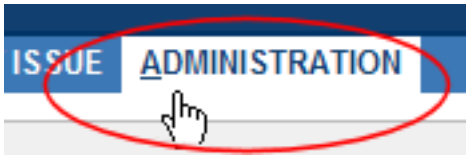
Deleting an Association

1. Click on the "Delete" link next to the issue operation you wish to remove.
2. The association will be automatically removed from the list.

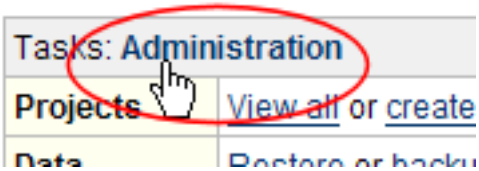
1.8.12.3. Managing Multiple Screen Schemes

In **Professional** and **Enterprise** Editions it is possible to create multiple Screen Schemes; in the **Standard** Edition you can only edit the Default Screen Scheme. The Screen Scheme operations are available from the View Screen Schemes page.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

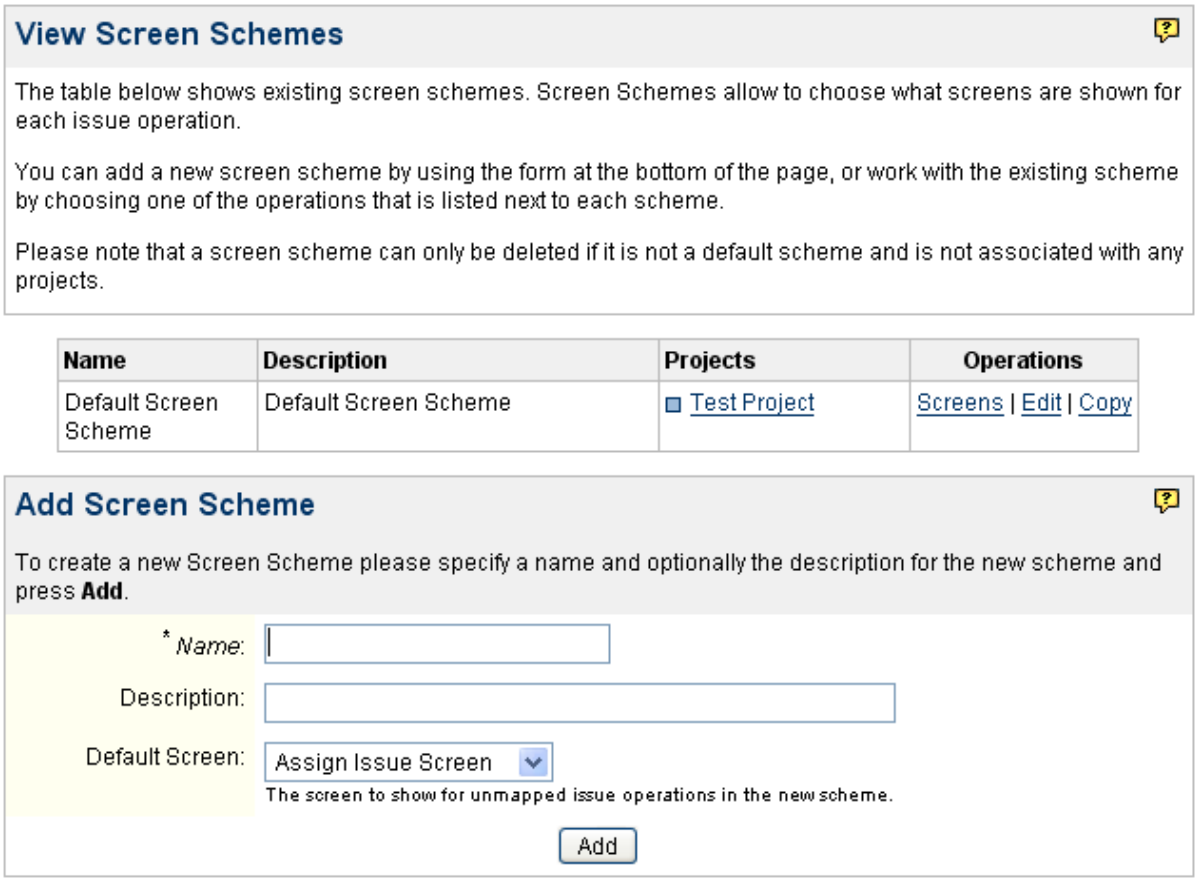


Link to Administration section



Link to Administration section

3. Click the "Issue Fields" sub-menu on the left hand side if it is not open already, and choose "Screen Schemes" from the list.
4. This will bring you to the "View Screen Schemes Page".



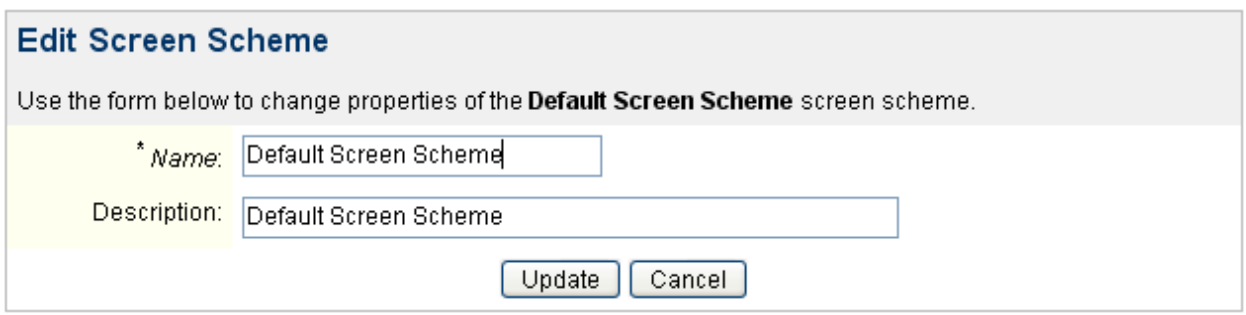
View Screen Schemes Page

Adding a Screen Scheme

1. The "Add Screen Scheme" form is located at the bottom of the Field Screen Schemes Page.
2. Enter the name of the new field screen.
3. You can optionally add a description of the field screen.
4. Select a default field screen. The default screen will be used for views that do not have an association.
5. Click on the "Add" button. The screen will automatically update the field screen schemes list with the new field screen scheme.

Editing a Screen Scheme's details

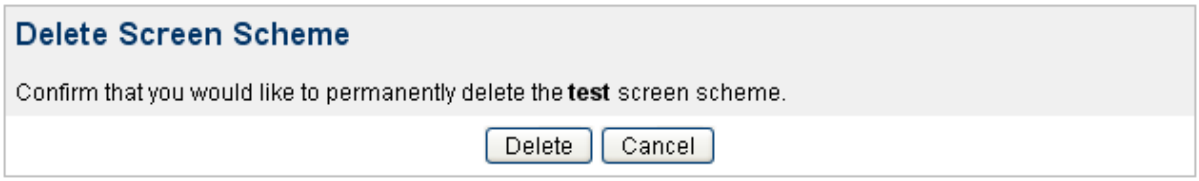
1. Click on the "Edit" link next to the selected Screen Scheme.
2. You will now be directed to the "Edit Screen Scheme" page where you can edit the Screen Scheme's name and description and the Screen that is associated with the *Default Entry* of the scheme.
3. Click on the "Update" button.
4. You will be brought back to the Screen Schemes page with your updates now applied to the Screen Schemes list.



Edit Screen Scheme Page

Deleting a Screen Scheme

1. Click on the "Delete" link next to the Screen Scheme you wish to delete.
2. Click on the "Delete" button to confirm this action.
3. You will be brought back to the Screen Schemes page with the Screen Scheme removed from the screen schemes list.



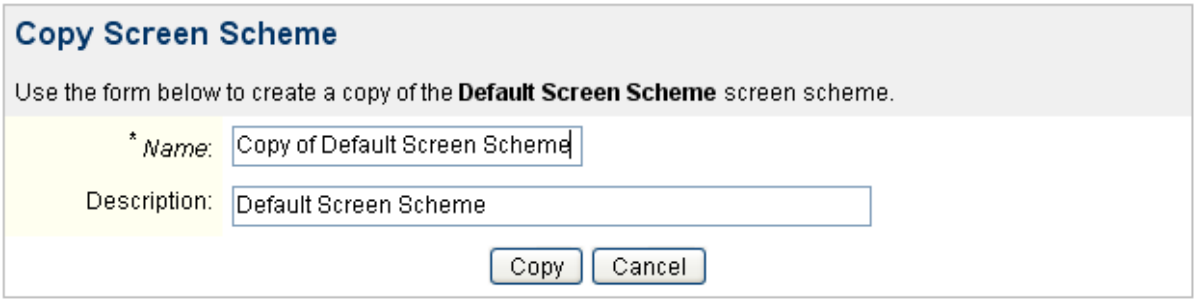
Delete Screen Scheme Page

Note:

Screen Schemes that are associated with at least one project in the Professional Edition or are used in an Issue Type Screen Scheme Enterprise Edition, cannot be deleted.

Copying a Screen Scheme

1. Click on the "Copy" link next to the Screen Screen you wish to copy.
2. You will now be directed to the "Copy Screen Scheme" page.
3. Enter the name and description of the new Screen Scheme.
4. Click on the "Copy" button.
5. You will be brought back to the Screen Schemes page, and the newly added Screen Scheme will have the same settings as the original Screen Scheme.



Copy Screen Scheme Page

1.8.12.4. Activating a Screen Scheme

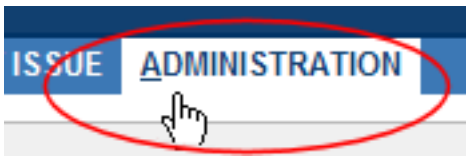
Standard Edition

The Standard Edition of JIRA supports only one Screen Scheme (the 'Default Screen Scheme'). As soon as the Default Screen Scheme is modified, the changes affect all projects and issue types in the system. No other activation is necessary.

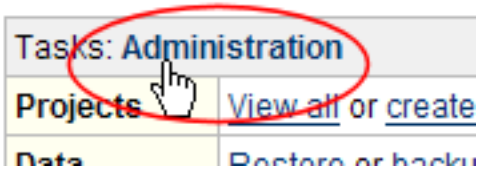
Professional Edition: associating a Screen Scheme with a project

Professional Edition allows for project-specific Screen Schemes. To activate a Screen Scheme, associate it with one or more projects as follows:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

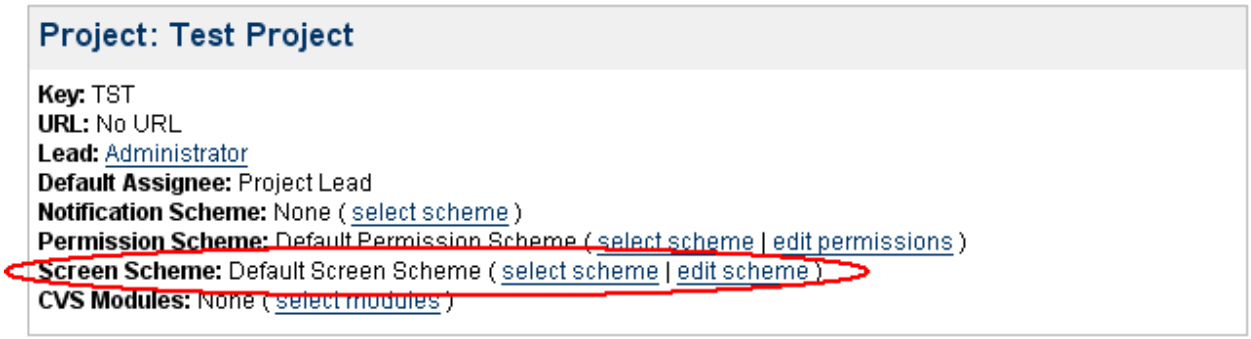


Link to Administration section



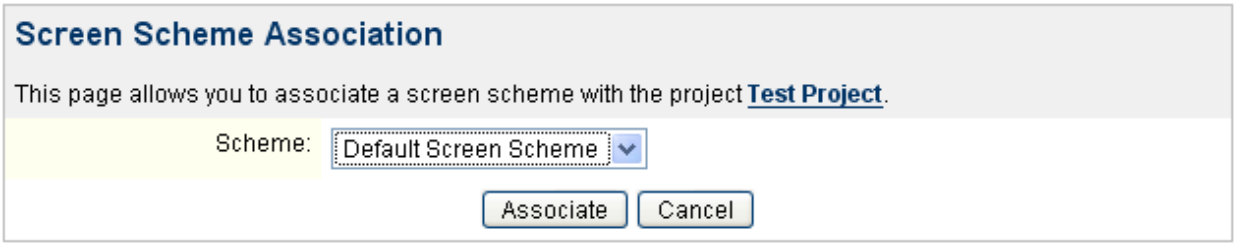
Link to Administration section

3. On the administration page is a list of projects which this user is allowed to manage. Select the project of interest.
4. Click on the "Select Scheme" link on the beside the Screen Scheme caption.



Project Admin Page

5. Select the screen scheme you wish to associate with this project.



Associate Screen Scheme to Project

6. Click on the "Associate" button.

Enterprise Edition

Enterprise Edition supports Screen Schemes per project per issue type, using Issue Type Screen Schemes. To activate a Screen Scheme, configure an Issue Type Screen Scheme to use the Screen Scheme, then associate the Issue Type Screen Scheme with a project. For details of both procedures, see 'Associating screens with Issue Types'.

1.8.13. Associating Screens with Issue Types

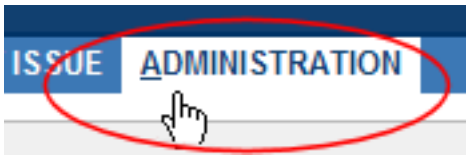
1.8.13.1. What is an 'Issue Type Screen Scheme'?

In the *Enterprise* edition of JIRA it is possible to define **Issue Type Screen Schemes**. Issue Type Screen Schemes associate Screen Schemes with issue types, allowing you to specify different Screens for the same operation (e.g. 'Create Issue') in the same project for issues of different issue types.

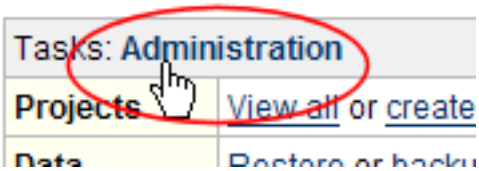
To configure and manage an Issue Type Screen Scheme you need to access the "View Issue Type

Screen Schemes" page:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

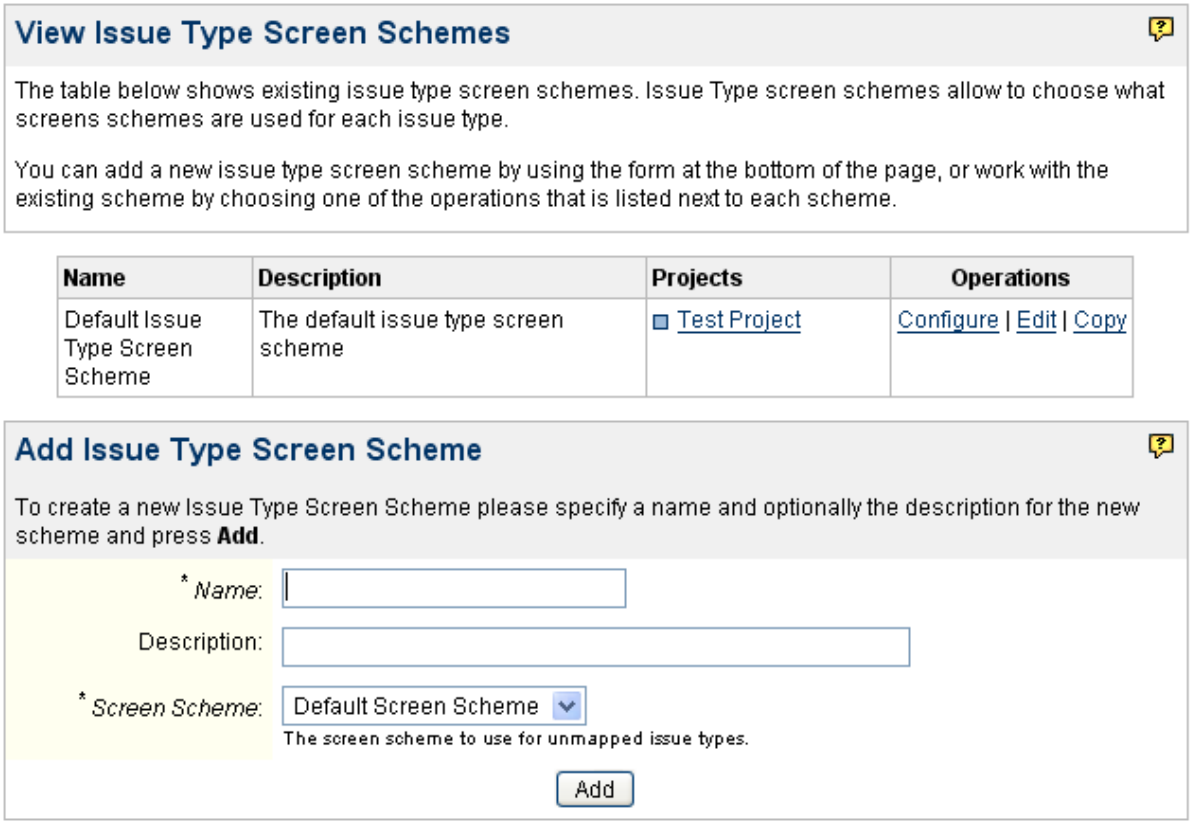


Link to Administration section



Link to Administration section

3. Click the "Issue Fields" sub-menu on the left hand side if it is not open already, and choose "Issue Type Screen Schemes" from the list.
4. This will bring you to the "View Issue Type Screen Schemes" Page.



View Issue Type Screen Schemes Page

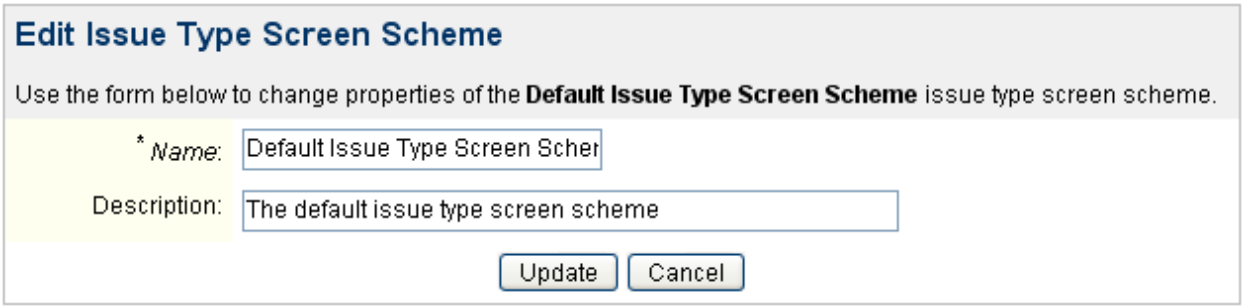
1.8.13.2. Add an Issue Type Screen Scheme

1. The "Add Issue Type Screen Scheme" form is located at the bottom of the Issue Type Screen Schemes page.
2. Enter the name for the new scheme.
3. You can optionally add a description.
4. Select a Screen Scheme for the *Default* entry in the new scheme. The Screen Scheme mapped to the *Default* entry will be used for issue types that do not have a specific mapping in the

scheme.
5. Click on the "Add" button. The screen will automatically update the Issue Type Screen Schemes list with the new Issue Type Screen Scheme.

1.8.13.3. Edit an Issue Type Screen Scheme

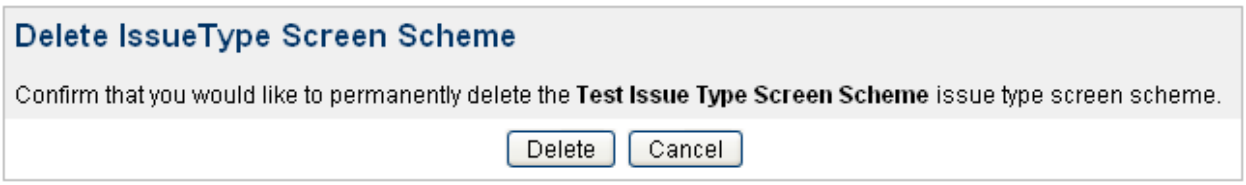
1. Click on the "Edit" link next to the selected Issue Type Screen Scheme.
2. You will now be directed to the "Edit Issue Type Screen Scheme" page where you can edit the Issue Type Screen Scheme's name and description as well as the Screen Scheme of the *Default* entry.
3. Click on the "Update" button.
4. You will be brought back to the Issue Type Screen Schemes Page with your updates now applied to the Issue Type Screen Schemes list.



Edit Issue Type Screen Scheme Page

1.8.13.4. Delete an Issue Type Screen Scheme

1. Click on the "Delete" link next to the Issue Type Screen Scheme you wish to delete.
2. Click on the "Delete" button to confirm this action.
3. You will be brought back to the Issue Type Screen Schemes Page with the Issue Type Screen Scheme removed from the Issue Type Screen Schemes list.



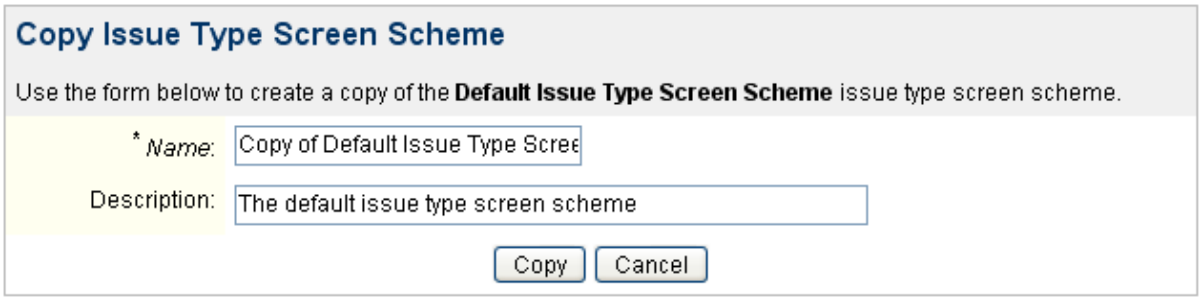
Delete Issue Type Screen Scheme Page

Note:

Issue Type Screen Schemes that are associated with a project cannot be deleted.

1.8.13.5. Copy an Issue Type Screen Scheme

1. Click on the "Copy" link next to the field screen you wish to copy.
2. You will now be directed to the "Copy Issue Type Screen Scheme" page.
3. Enter the name and description of the new Issue Type Screen Scheme.
4. Click on the "Copy" button.
5. You will be brought back to the Issue Type Screen Schemes Page and the newly added Issue Type Screen Scheme will have the same scheme settings as the original Issue Type Screen Scheme.

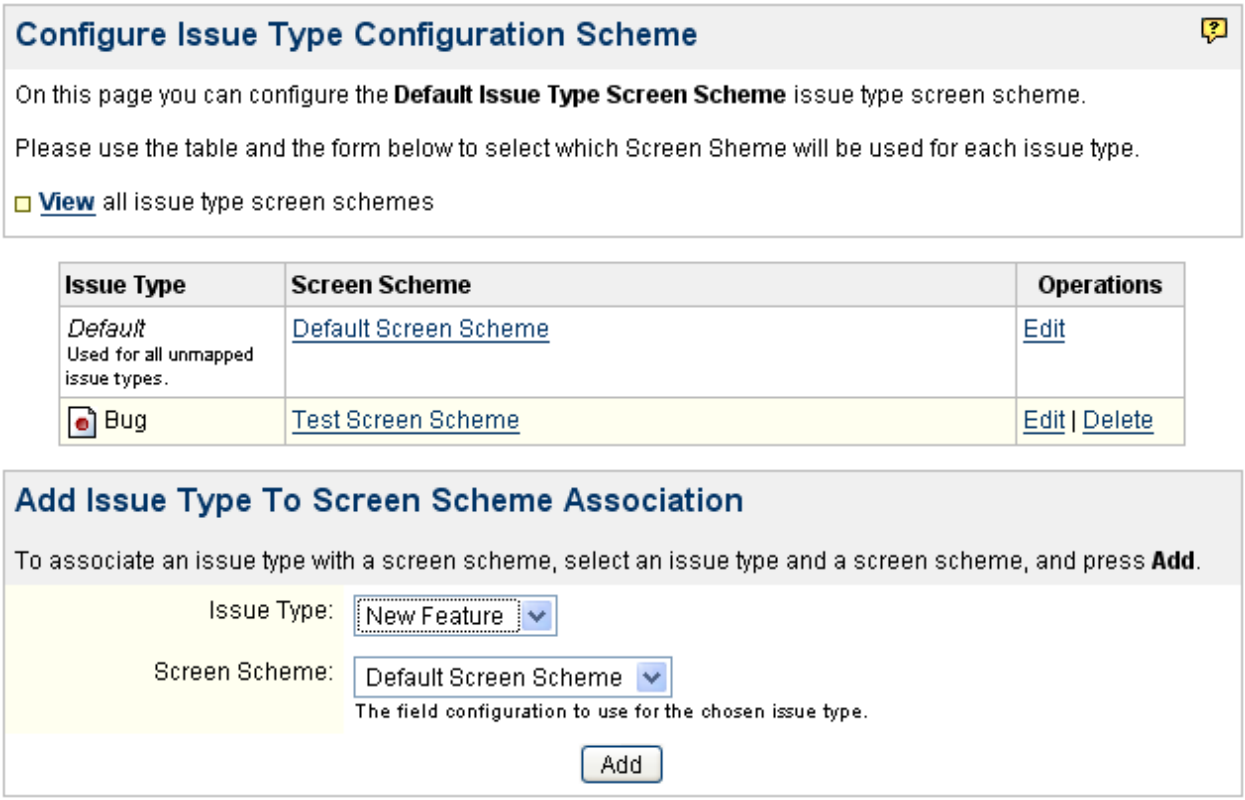


Copy Issue Type Screen Scheme Page

1.8.13.6. Configure Issue Type Screen Scheme

The configuration of an Issue Type Screen Scheme involves associating an issue type to a particular Screen Scheme. For example, associating the Bug issue type to the "Default Screen Scheme" and then associating Improvement to the "Improvement Screen Scheme".

To configure a given Issue Type Screen Scheme click on the "Configure" link next to the selected Issue Type Screen Scheme from the "View Issue Type Screen Schemes" Page. You will then be directed to the Configure Issue Type Screen Scheme Page



Configure Issue Type Screen Scheme Page

Associate an Issue Type to a Screen Scheme

1. The "Add Issue Type to Screen Scheme Association" is located at the bottom of the Configure Issue Type Screen Scheme Interface.
2. Select an issue type you wish to associate a Screen Scheme with.
3. Select the desired scheme.
4. Click the "Add" button and the new association will be added to the association list above.

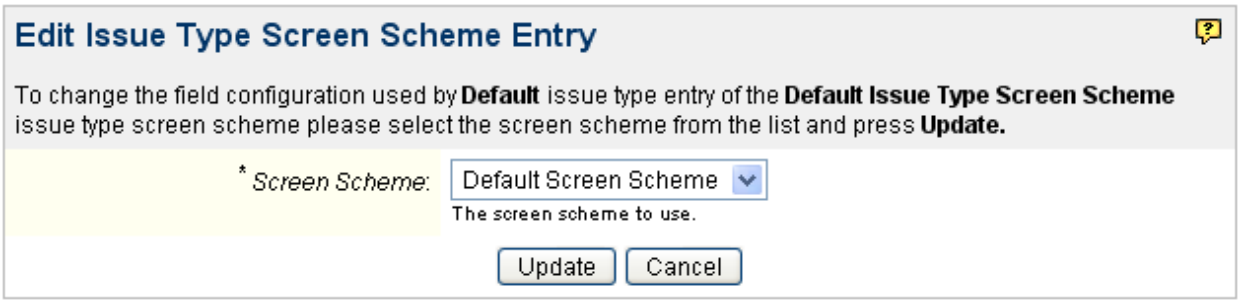
Note:

There can only be one association for each issue type. If all issue types have been associated with a Screen Scheme you can use the 'Edit' link next to each entry to change the Screen Scheme that is associated with it.

Note:
If there is no specific entry for an issue type, the Screen Scheme associated with the *Default* entry will be used.

Edit an Association

1. Click on the "Edit" link next to the issue type you wish to edit.
2. You will be brought to the "Edit Issue Type Screen Scheme Entry" page.
3. Select the screen you wish to change this association to.
4. Click on the "Update" button and you will be returned to the Issue Type Screen Scheme interface.



Edit Issue Type Screen Scheme Entry Page

Delete an Association

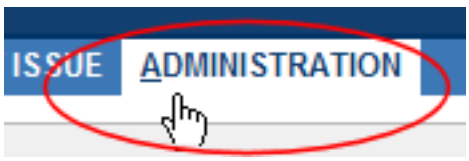
1. Click on the "Delete" link next to the issue operation you wish to remove.
2. The association will be automatically removed from the table.

Note:
The *Default* entry is used for all issue types that do not have a specific entry in the scheme. It cannot be deleted.

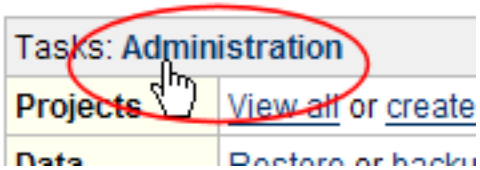
1.8.13.7. Associating an Issue Type Screen Scheme with a Project

Once you have created and configured an Issue Type Screen Scheme to your desired settings you can now associate the scheme with a Project. This will apply your chosen Screen Scheme to each issue type within the selected project.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

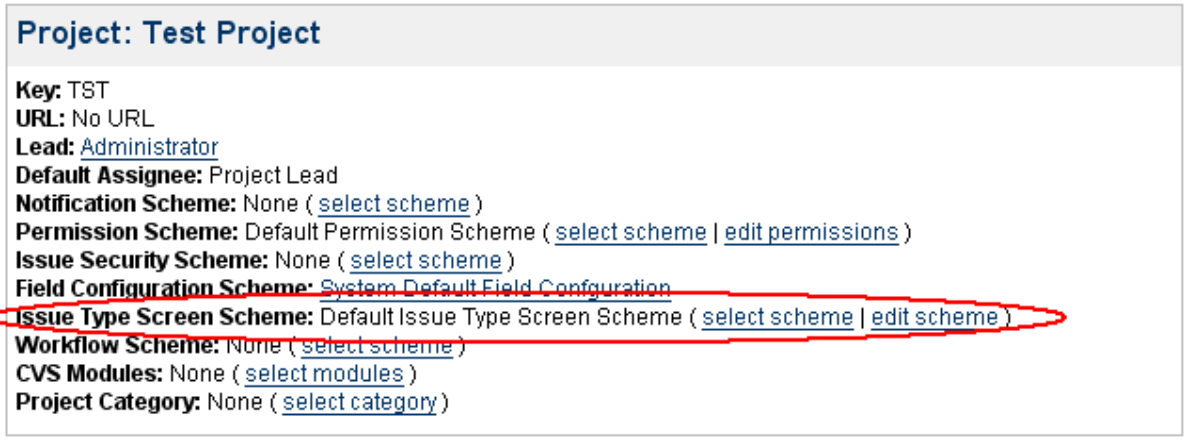


Link to Administration section



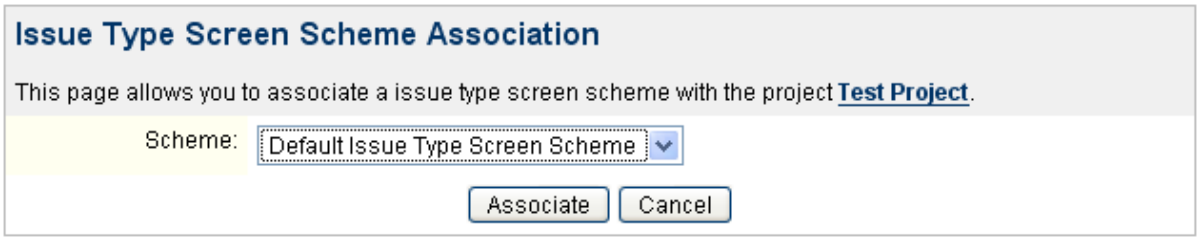
Link to Administration section

3. On the Administration page is a list of projects which this user can manage. Select the project of interest.
4. Click on the "Select Scheme" link on the beside the Issue Type Screen Scheme caption.



Project Admin Page Enterprise

5. Select the screen scheme you wish to associate with this project.



Associate Issue Type Screen Scheme to Project

6. Click on the "Associate" button.

Note:
To control which issue types apply to a project, please see 'Associating Issue Types with Projects'.

1.8.14. Creating & configuring Custom Fields

1.8.14.1. Custom Fields Overview

Custom field types were introduced in JIRA 2.0 to allow greater customisability of the types of data collected with your issue. In 3.0, the number types have been expanded and you can even add your own custom field types. JIRA 3.2 adds a new level of flexibility to your custom fields. You can now configure your custom fields to only appear for certain issue types in certain projects or multiple issue types over multiple projects. On top of that, you can even configure each custom field differently for each context.

This page outlines some of the key concepts relating to custom fields.

To build your own custom field types, check out the tutorial at the JIRA Development Hub.

Note:
Custom fields are always optional fields. This means you can add custom fields without requiring existing issues to be changed. The current issues contain no value for the custom field, even if a default is defined.

Custom Field Types

JIRA now ships with over 20 custom field types and you can find more custom field types and other examples in the JIRA Extensions space (e.g. JIRA Toolkit). A sample of the types are listed below.

Custom Field Type	Description

Cascading Select	Multiple select lists where the options for the second select list dynamically updates based on the value of the first
Date Picker	Input field allowing input with a date picker and enforcing valid dates
Date Time	A custom field that stores dates with a time component.
Free Text Field (unlimited text)	Multiple line text-area enabling entry of longer text strings
Multi Checkboxes	Checkboxes allowing multiple values to be selected
Multi Select	Select list permitting multiple values to be selected
Number Field	Input field storing and validating numeric (floating point) values
Project Picker	Select list displaying the projects viewable by the user in the system
Radio Buttons	Radio buttons ensuring only one value can be selected
Select List	Single select list with a configurable list of options
Text Field	Basic single line input field to allow simple text input of less than 255 characters
URL Field	Input field that validates a valid URL
User Picker	Choose a user from the user base via a popup picker window.
Multi User Picker	Choose one or more users from the user base via a popup picker window.
Group Picker	Choose a user group using a popup picker window.
Multi Group Picker	Choose one or more user groups using a popup picker window.
Single Version Picker	Choose a single version from available versions in the project.
Version Picker	Choose one or more versions from available versions in the project.

Search templates

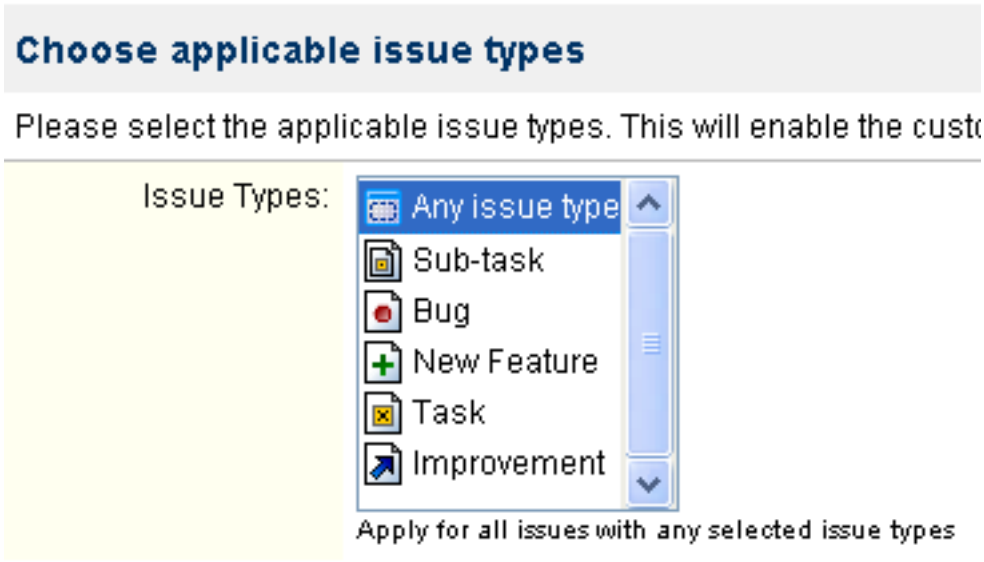
Search templates are responsible for indexing a custom field as well as making it searchable through the Issue Navigator (note that custom fields are not searchable via QuickSearch). Each of the default custom field types has a related pre-configured search template.

Custom field context

The custom field context (introduced in JIRA 3.2) allows your custom field to be configured (that is, enabled) for any numerous different combinations of issue types and projects. You can have

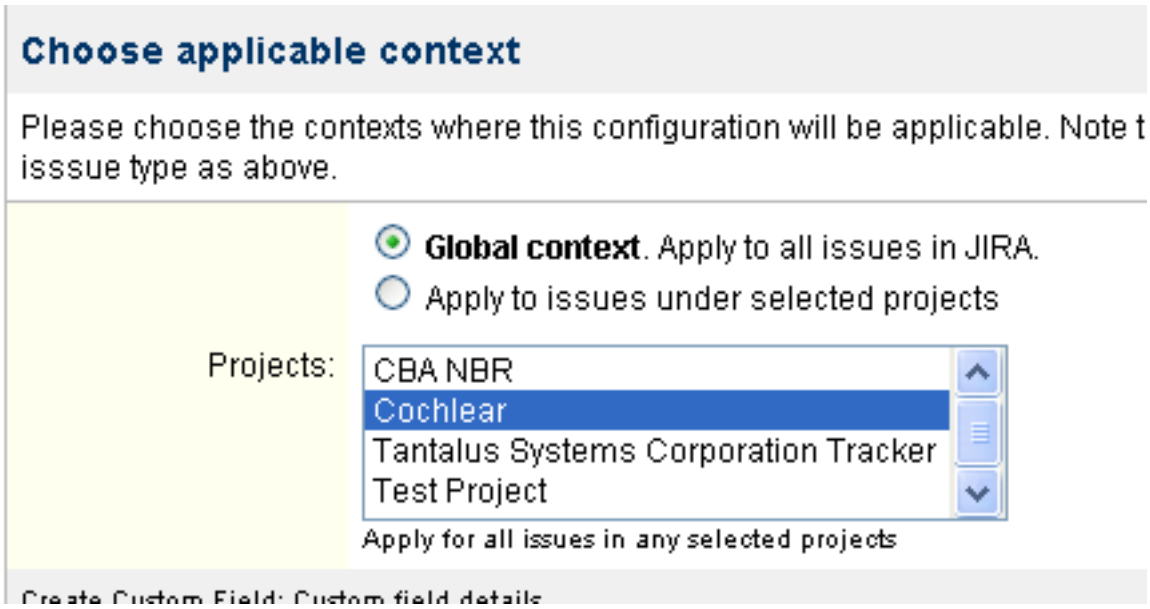
different default values in different projects, different options for different projects and the like.

The context is made up of an issue type component and a project component. You can select multiple issue types and multiple projects or declare the custom field to be global.



Issue type context

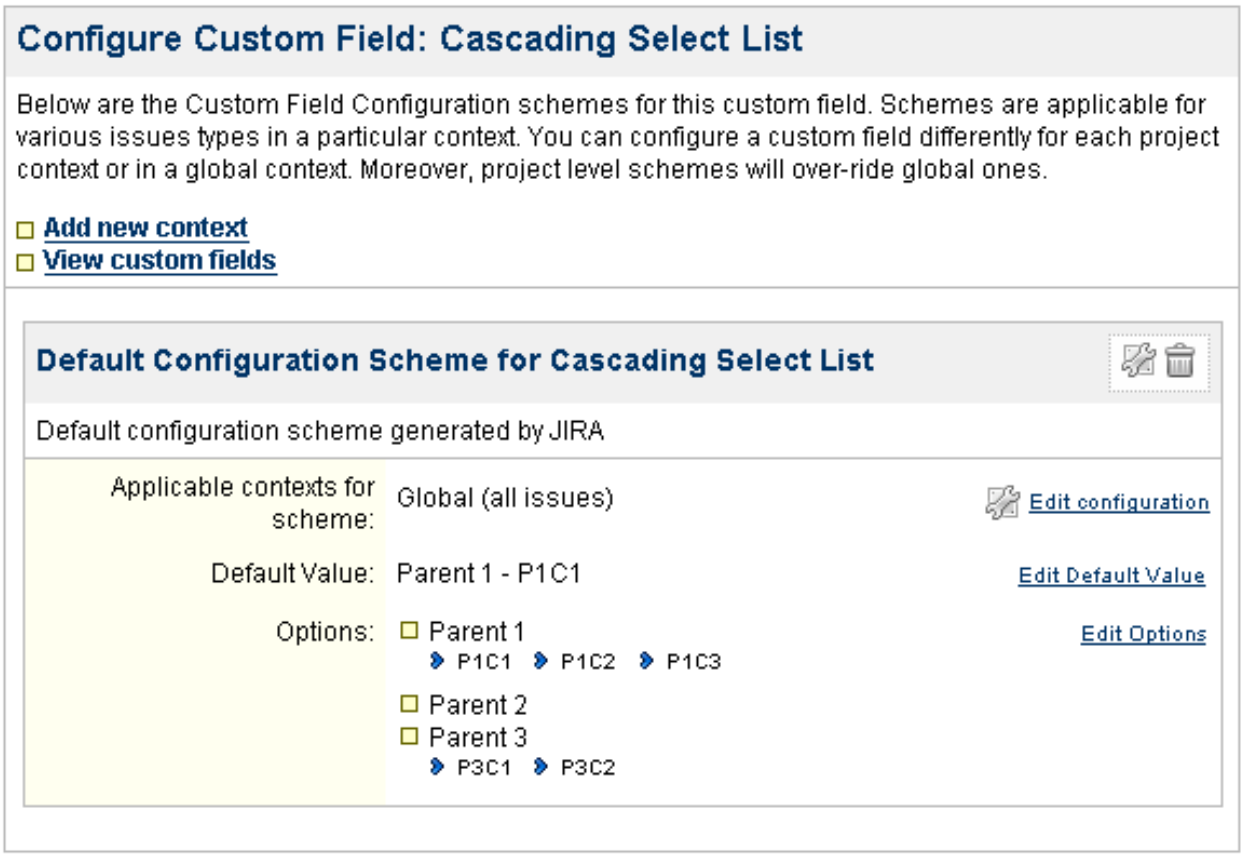
The context itself can now be modified at any time. You can change the project or issue type applicable for the custom field at any time.



Project context

Custom field configuration schemes

If you start digging deeper into custom fields (or indeed, any part of JIRA) you'll notice many references to schemes. Custom field configuration schemes are how JIRA allow you to manage custom field contexts and configuration. A configuration scheme is configuration set for a group of issue types for a set of projects. If you have two different default values, you'd need two configuration schemes and so on.



Project context

Specific project based configuration schemes will override configurations from a Global project context. So you could configure a **default** global scheme for all projects and the configure for each projects that are different. You may for example, have a global select lists that have values that applies for 80 of your 81 projects but is different in the other. You'd configure a one configuration scheme for global context and other for the specific field that is different.

Also note that to avoid conflicts, a project can only be part of a single configuration scheme. Once you've selected a specific project to be part of a scheme, it will be removed from the list of selectable options

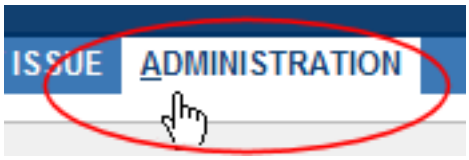
Additional Resources

- Adding a custom field tutorial video — Watch this short tutorial video to see how to add and configure a new custom field. Please note the JIRA version and JIRA edition of the tutorial video before watching.

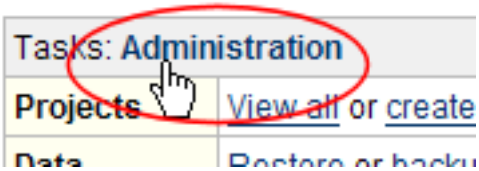
1.8.14.2. Adding Custom Fields

Steps to define a custom field

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

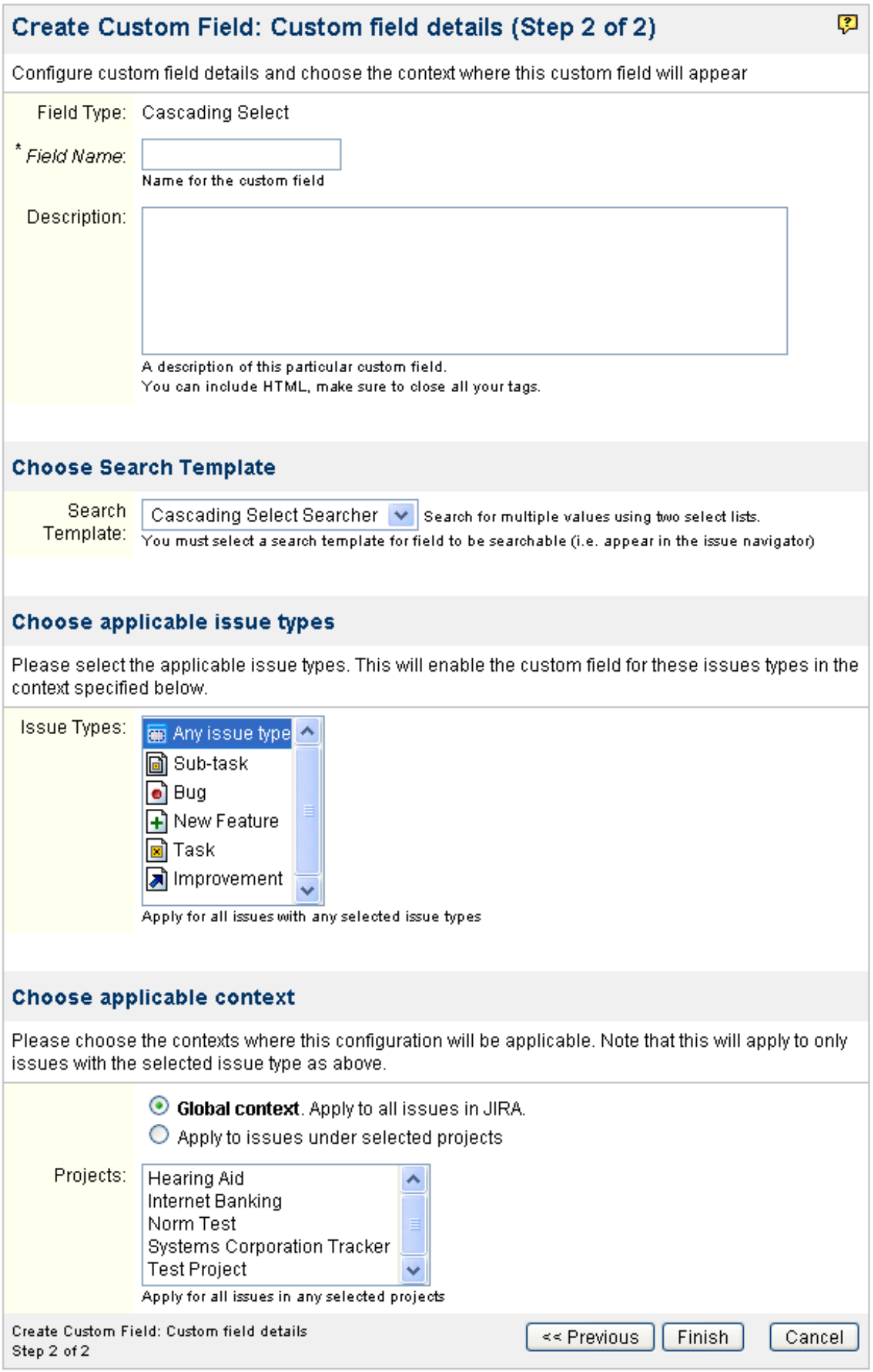


Link to Administration section

3. On the menu bar on the left, under "Issue Fields" sub-menu click the Custom Fields link, and then click on the Add Custom Field link on the presented page.

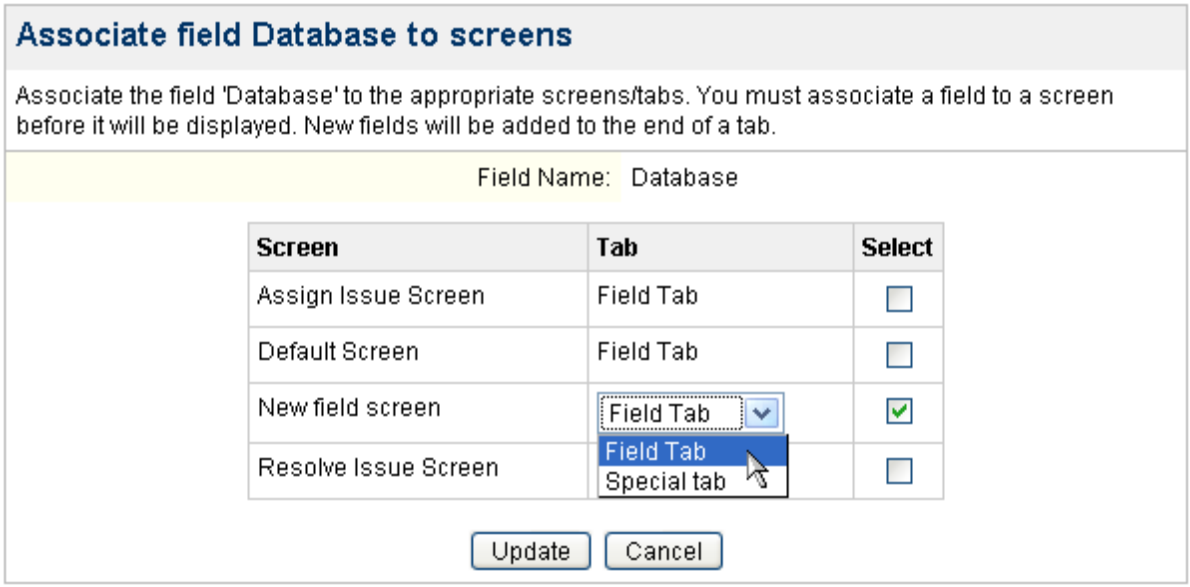
Custom field creation screen

4. Select from the list the appropriate custom field type.
5. Click on the next button.



Custom field creation screen

6. Fill in the Field Name and Field Description. The Field Name will appear as the custom field's title in both entering and retrieving information on issues. The Field Description is displayed beneath the data entry field when entering new issues and editing existing issues, but not when browsing issues.
7. Select an appropriate Search Template. Pre-configured search templates are available for each shipped custom field type. A description of each search template will appear next to the select list when you select one.
8. Select one or any number of issue types that this custom field will be applicable for. You can change this value in the future if you need to.
9. Select the applicable project context. The custom field will be available to the selected projects. If issue types were chosen, it will only appear for those issue types for that project
10. Click **Finish**.
11. This will bring you to the screen association page where you can put your newly created custom field onto a screen in JIRA. You can associate the field any screens or tabs in JIRA. You must associate a field to a screen before it will be displayed. New fields will be added to the end of a tab.



Screen association field

12. Clicking **Update** will return you to the View Custom Fields page that displays a summary of the custom fields in the system. You can edit, delete or configure custom fields here. This page is also directly accessible from the menu bar to the left of all Administration pages.

Additional Resources

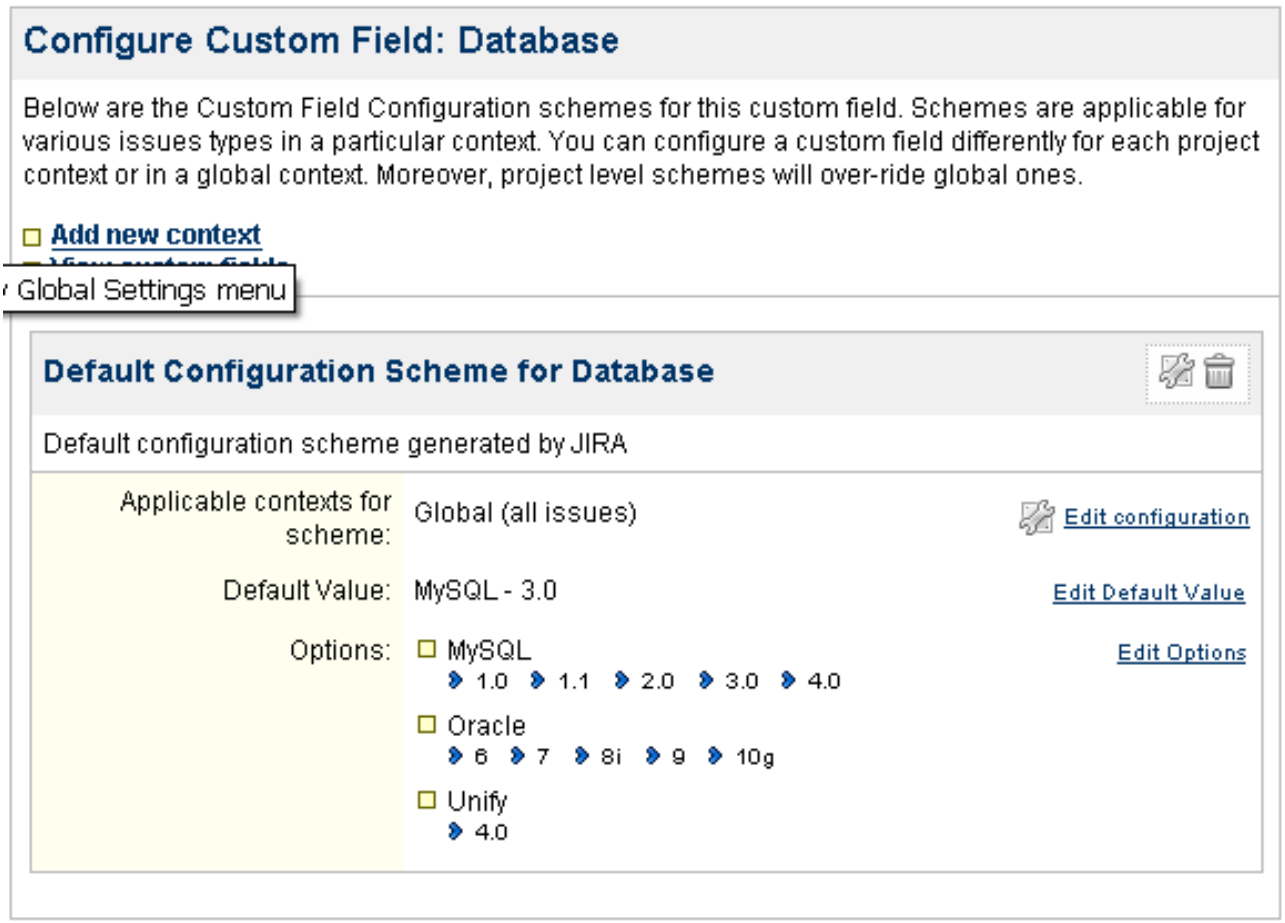
- Adding a custom field tutorial video — Watch this short tutorial video to see how to add and configure a new custom field. Please note the JIRA version and JIRA edition of the tutorial video before watching.

1.8.14.3. Configuring Custom Fields

On this page, we show you how you can configure your custom field after you've created them.

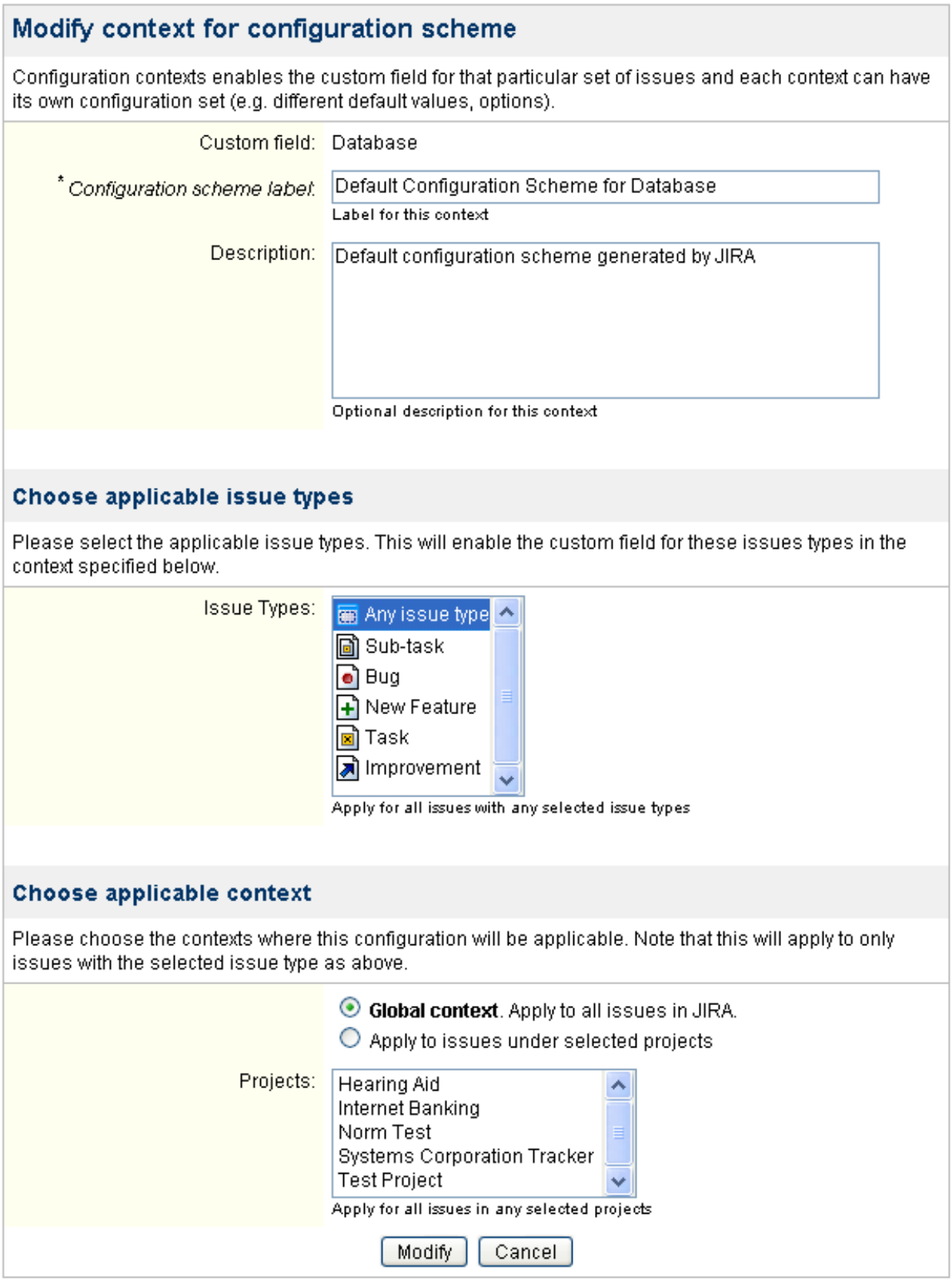
Configuring Custom Fields

For most custom fields, you can configure its default values and set the options for lists. When you click on the *configure* link, you'll be faced with a page similar to the one below.



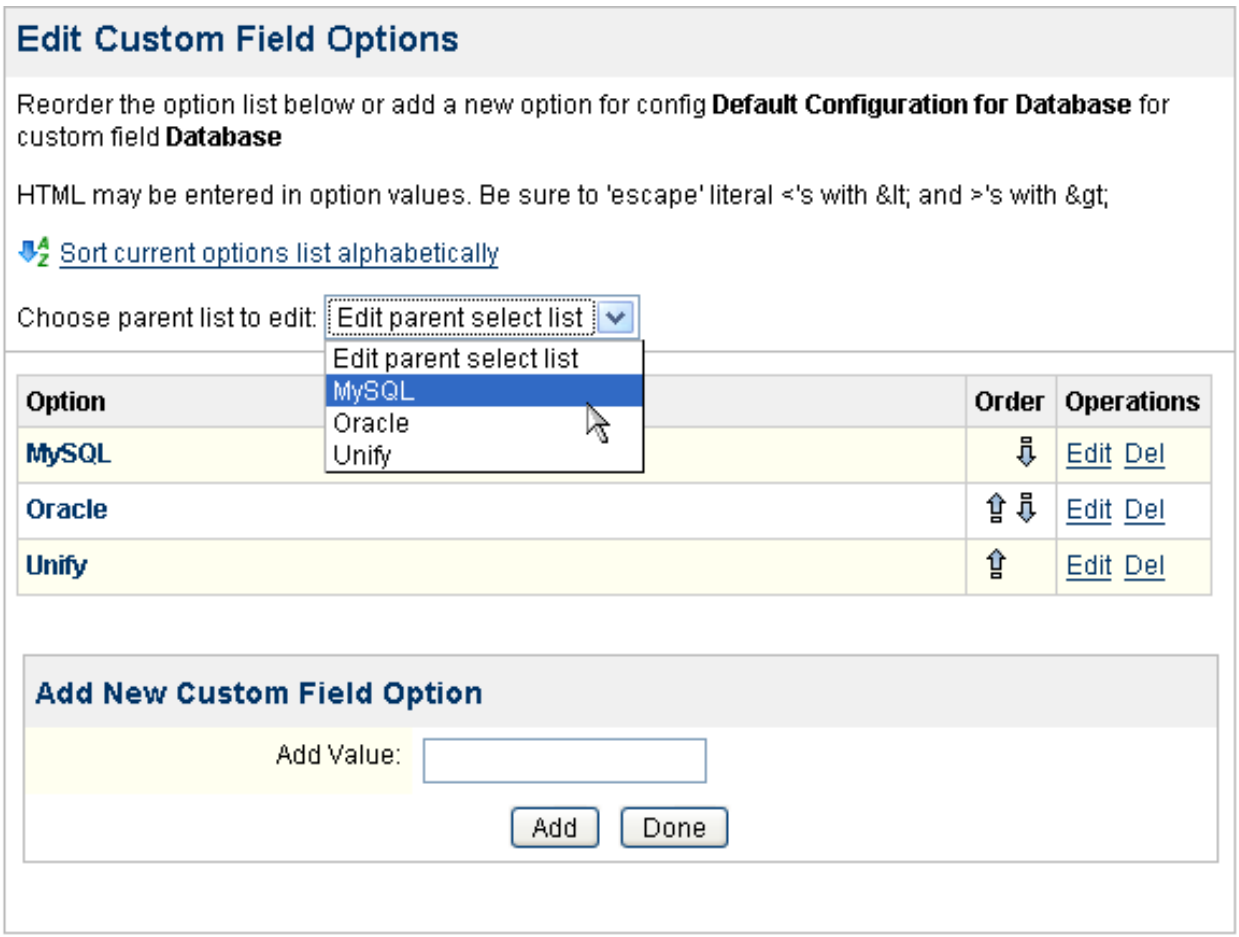
Configuring a custom field

You'll notice that there is a configuration scheme named *Default Configuration Scheme for...*. This is the configuration scheme created automatically by JIRA when you initially added your custom field. The *Applicable contexts for scheme* refers to context that this scheme will be applied to; which projects & issue types the defaults and options will be displayed. For most people this will be only thing you need to know about custom field configuration schemes. You can edit the context by clicking *Edit configuration* or the edit icon on the top left hand corner.



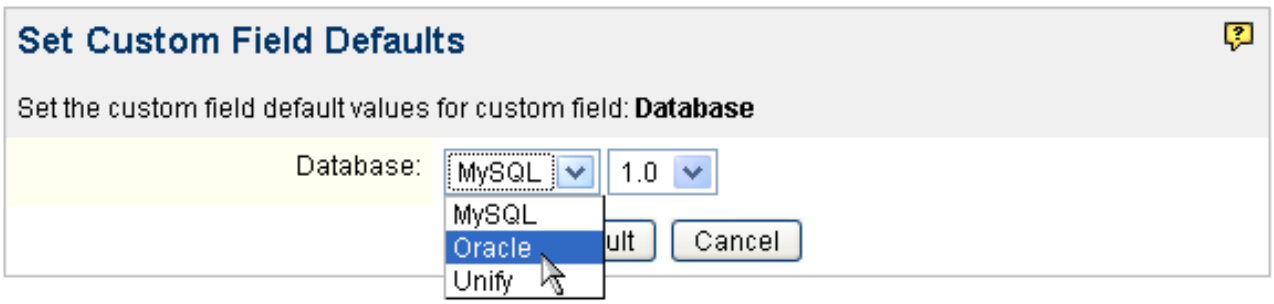
Configuring a custom field configuration scheme

Here you can edit the label and description of the configuration scheme. These are used for administrative purposes only and isn't shown to the end users. You can also change the context that this scheme is to be applied to here. It's now easier than ever to change the projects or issue types for the configuration context.



Options of a custom field

Select lists, multi select lists and cascading selects lists can have their options manipulated. You can add, remove and sort the options alphabetically. You can also have HTML in an option value. Be sure to use complete all tag pairs and ensure that it will display correctly.



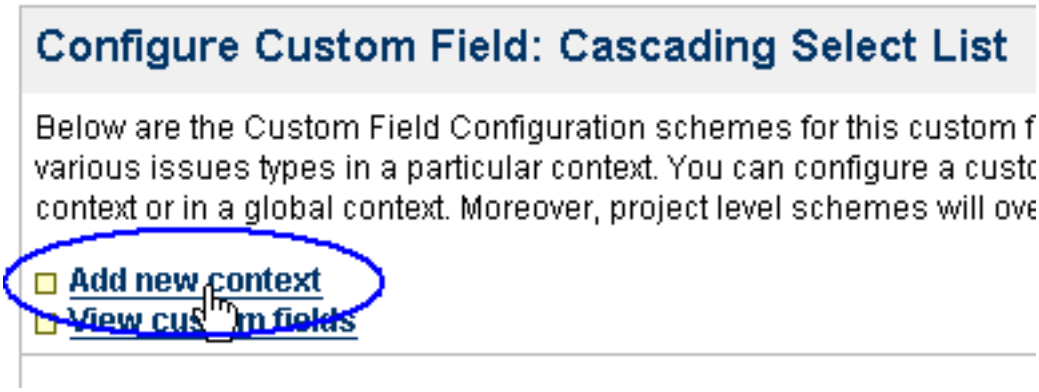
Defaults of a custom field

Click the *Edit defaults* to modify the default value of a custom field for this configuration scheme. Setting the defaults of will take you to a screen that is particular to the issue type. Certain custom fields such as calculated custom fields may not allow for defaults to be selected and will not have the "edit defaults" link.

Managing multiple configuration schemes

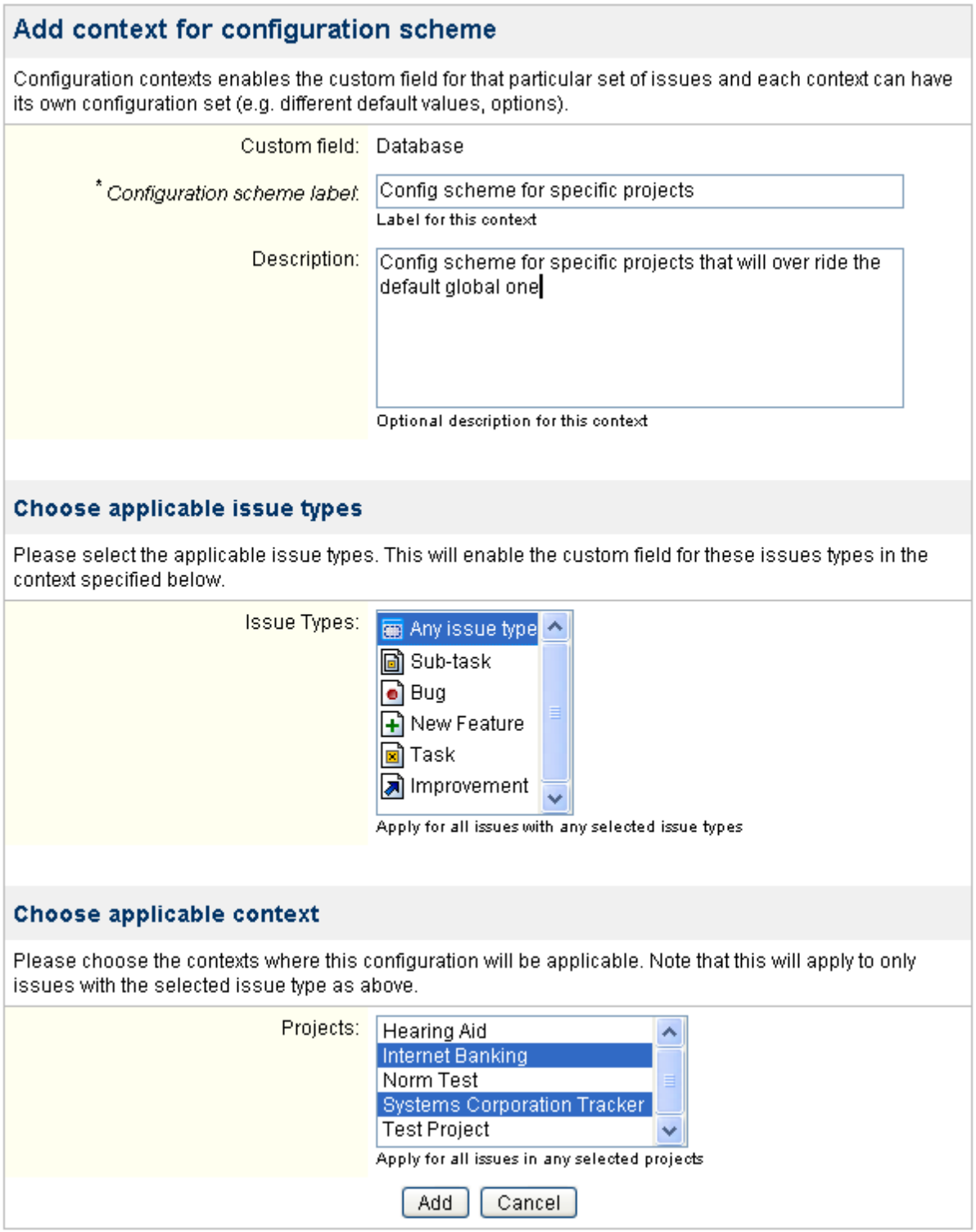
Since JIRA 3.2, it is now possible to configure a custom field differently for different issue types and project combinations. This can be achieved through configuring different custom field schemes.

1. Click on the *Add context* link to create a new custom field configuration scheme.



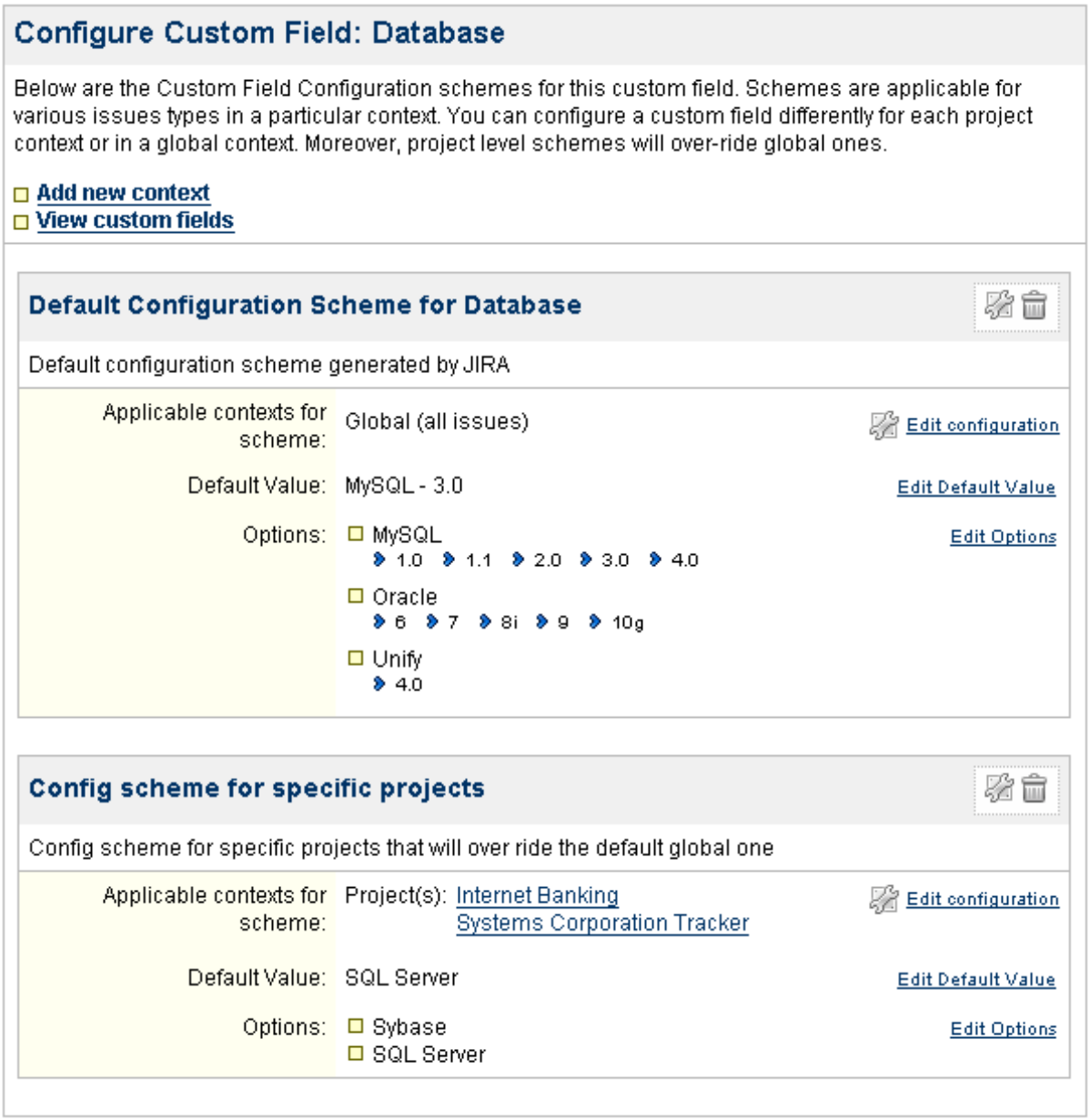
Adding a new custom field configuration scheme

2. You'll see a screen that very similar to the edit configuration scheme. Here you can select the applicable context for your new configuration scheme. One difference is that the project context will only show projects and options that have not be previously selected in another configuration scheme. So if you already have an existing "Global" configuration scheme, that option will not be available. Moreover, be aware that project specific configuration scheme will override Global configurations. For example, if you have one global configuration scheme with one default value and a project configuration scheme with a different value, the project value will be used instead.



Setting context for configuration scheme

3. Click *Add*
4. You will now have a new configuration scheme that can be reconfigured separately to the default scheme. You can add different default values and options for each one. In the screen show below, the *Database* custom field have specialised options and defaults for two to the projects, which will over-ride the global options list.



Multiple configuration scheme

Additional Resources

- Adding a custom field tutorial video — Watch this short tutorial video to see how to add and configure a new custom field. Please note the JIRA version and JIRA edition of the tutorial video before watching.

1.9. Configuring Workflow

1.9.1. JIRA Workflow

A JIRA *workflow* is the set of *steps* and *transitions* an issue goes through during its lifecycle. Workflows typically represent business processes.

All editions of JIRA are shipped with one default workflow. The default workflow cannot be edited, but in JIRA Enterprise and Professional editions you can customise the issue lifecycle by creating additional workflows:

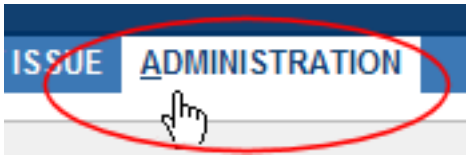
- **JIRA Enterprise** supports multiple active workflows. Each workflow can be associated with

particular projects and (optionally) particular issue type(s).
- **JIRA Professional** supports only one active workflow. That is, in JIRA Professional any number of workflows can be defined, but all issues in the system must use only one of the defined workflows at any point in time.

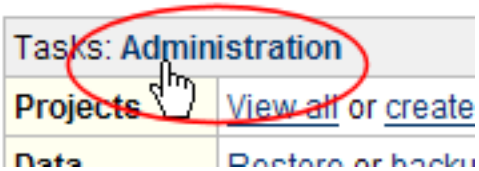
1.9.1.1. On this page:

1.9.1.2. Creating a workflow

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

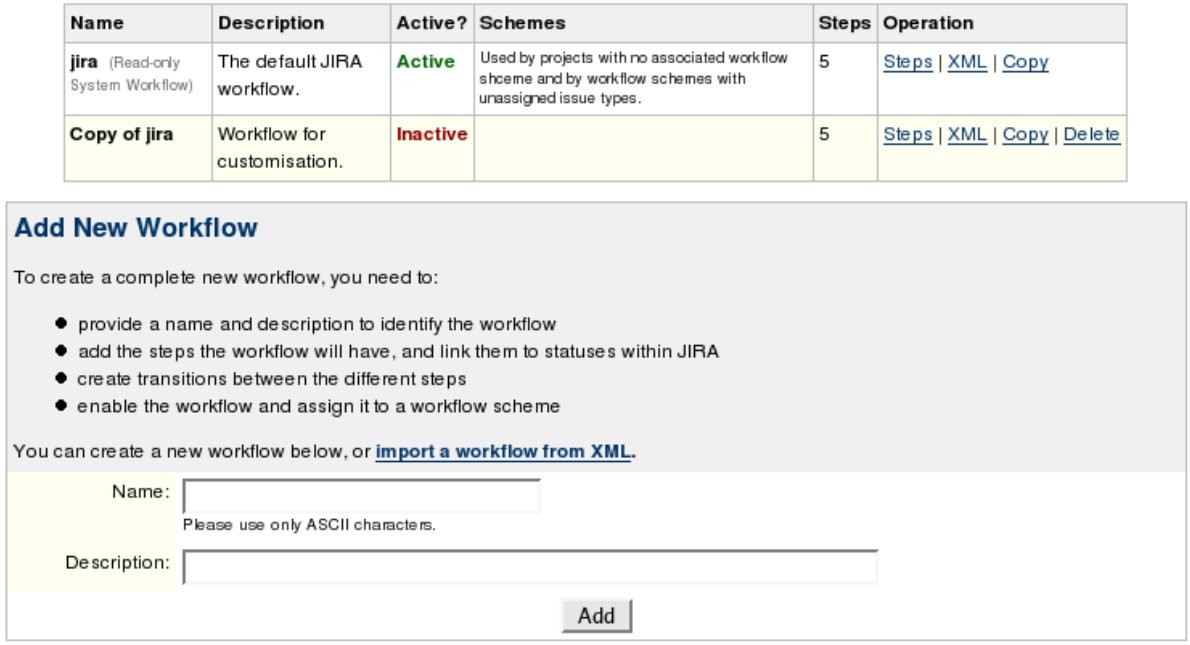


Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed, listing all the workflows that are currently defined in your JIRA system:



Name	Description	Active?	Schemes	Steps	Operation
jira (Read-only System Workflow)	The default JIRA workflow.	Active	Used by projects with no associated workflow shceme and by workflow schemes with unassigned issue types.	5	Steps | XML | Copy
Copy of jira	Workflow for customisation.	Inactive		5	Steps | XML | Copy | Delete

Add New Workflow

To create a complete new workflow, you need to:

- provide a name and description to identify the workflow
- add the steps the workflow will have, and link them to statuses within JIRA
- create transitions between the different steps
- enable the workflow and assign it to a workflow scheme

You can create a new workflow below, or import a workflow from XML.

Name:

Please use only ASCII characters.

Description:

Add

View Workflows page

Note:

This screenshot shows how the page would appear in the Enterprise edition of JIRA. JIRA Professional does not support Workflow Schemes and would therefore not have the '**Schemes**' column.

5. To create a new workflow in JIRA, either:
   - **Create a 'blank workflow** by using the '**Add New Workflow**' form at the bottom of the page:
     1. In the '**Name**' field, type a name (usually 2-3 words) to identify your new workflow.
     2. *(Optional)* In the '**Description**' field, type a detailed description of your new workflow.
     3. Click the '**Add**' button.

     Your new workflow will contain one step, called '**Open**', which has an incoming transition called '**Create**'.
   - **Copy an existing workflow** (this is useful if you already have a workflow that is similar to what you need) by clicking the '**Copy**' link next to an existing workflow:
     1. In the '**Name**' field, type a name (usually 2-3 words) to identify your new workflow.
     2. *(Optional)* In the '**Description**' field, type a detailed description of your new workflow.
     3. Click the '**Copy**' button.

     Your new workflow will contain the same steps and transitions as the workflow you copied.

Note:

If you are copying the default JIRA workflow and wish to rename the transitions, you will need to delete the '**jira.i18n.title**' and '**jira.i18n.description**' properties from all of the transitions. Otherwise, the default names will persist. Read more about transition properties.

6. Once you have created your new workflow you may want to customise it by adding or editing steps and transitions (see below) — especially if you have created a blank workflow.
7. When you have finished customising your new workflow, see *Activating Workflow* for how to use it.

1.9.1.3. Editing a Workflow

The process for editing a workflow differs depending on whether you are editing an inactive workflow or an active workflow. Restrictions are placed on the modifications you can make to an active workflow, due to the impact the changes will have on projects and/or issue types that the workflow is applied to.

Editing an Inactive Workflow

Editing a workflow means that you are modifying the steps and transitions that make up a workflow. Read more about modifying steps and transitions on this page.

Editing an Active Workflow

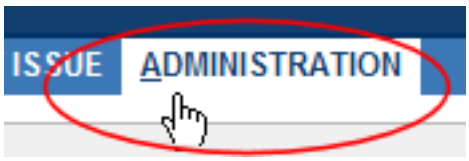
An active workflow can be edited in a manner similar to editing an inactive workflow. You will be able to make quick edits to the workflow in a live draft with the benefit of real-time validations. Once you publish your changes, you also have the option of saving your old workflow as an inactive backup.
Please note however, that the following limitations will apply when editing an active workflow:

- Existing workflow steps cannot be deleted.
- Associated status for each existing step cannot be edited.
- Step IDs for existing steps cannot be changed.

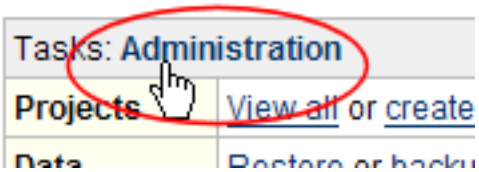
If you wish to make any of the modifications listed above, then you will need to copy the workflow (see *'Creating a Workflow'* above), modify the copy and then activate it. Please note, this method will be **significantly slower**, particularly for large instances of JIRA.

To edit an active workflow:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

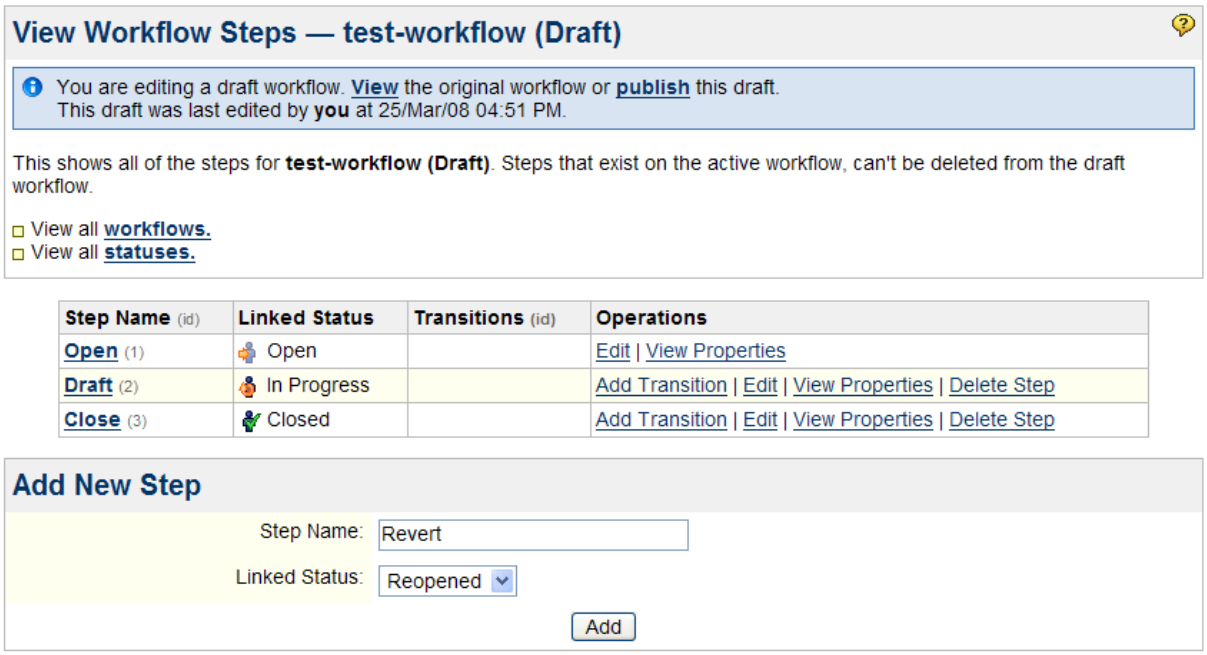


Link to Administration section



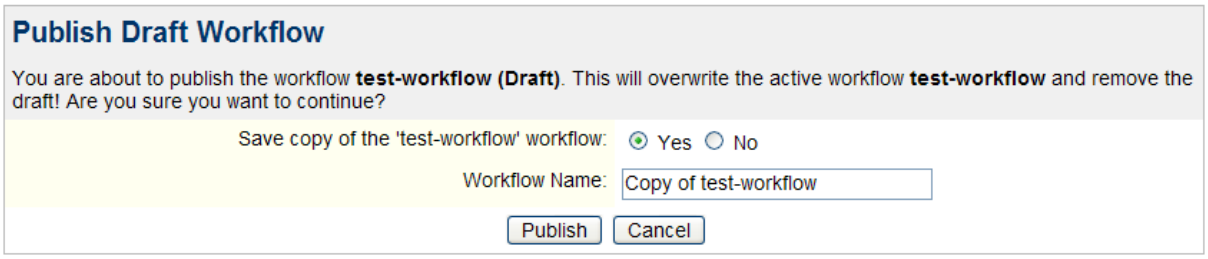
Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow that you wish to edit.
5. The '**Workflow Steps**' page will be displayed. Click the **'Create a draft workflow**' link in the information message displayed at the top of the screen.
6. The '**Workflow Steps**' page will be reloaded, as shown below. You will now be able to edit a draft of the workflow as described in the sections above. Any changes that you make to this draft will not affect the active workflow until you publish your draft.



Screen shot of JIRA draft workflow

7. When you have completed your changes, click the '**publish this draft**' link in the information message displayed at the top of the screen.
8. A confirmation screen will display, as shown below:



Screen shot of publish draft workflow confirmation

Select whether you wish to save the original workflow as an inactive copy. If you choose to retain the original workflow, enter a name for the inactive copy. Click '**Publish**' to publish your draft (i.e. commit your changes to the active workflow).

1.9.1.4. About steps and transitions

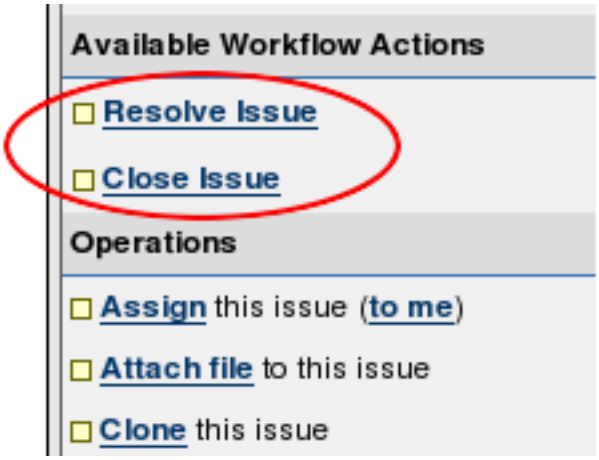
A workflow consists of **steps** and **transitions**:

- A **step** represents a stage in a workflow for an issue. An issue can exist in only one step at any point in time. Each workflow step corresponds to (and is usually named after) a 'linked' status. When an issue is moved into a particular step, its '**Status**' field is updated to the value of the step's 'linked' status.

  When defining a step, you can optionally specify **properties** — these allow you to make an issue uneditable while it is in this step.

- A **transition** is a link between two steps. A transition allows an issue to move from one step to another step. For an issue to be able to progress from one particular step to another, a transition must exist that links those two steps. Note that a transition is a *one-way* link, so if an issue needs to move back and forth between two steps, two transitions need to be created.

  The available workflow transitions for an issue are listed on the issue's '**View Issue**' page. A user can execute a transition (i.e. move the issue through workflow) by clicking one of the available links, e.g.:



Available Workflow Transitions screen

When defining a transition, you can optionally specify:

- A **screen** to be displayed to the user — this is useful if you need the user to provide input before completing the transition.
- **Conditions** — these control who can perform a transition (i.e. who can see the transition link on the '**View Issue**' page).
- **Validators** — these check that any user-supplied input is valid before performing the transition.
- **Post Functions** — these perform particular actions after the transition is complete, e.g.:
  - Assign the issue to a particular user.
  - Send an email notification.
  - Update a field in the issue.

In the diagram of the default workflow, the five boxes represent steps/statuses ('**OPEN**', '**IN PROGRESS**', '**CLOSED**', etc) and the arrows represent transitions.

A note about 'open' and 'closed' issues

Within JIRA (e.g. in the '**Assigned To Me**' portlet and other portlets), an issue is determined to be 'open' or 'closed' based on the value of its '**Resolution**' field — not its '**Status**' field.

- An issue is determined to be 'open' if its '**Resolution**' field has not been set.
- An issue is determined to be 'closed' if its '**Resolution**' field has a value (e.g. '**FIXED**', '**CANNOT REPRODUCE**').

This is true regardless of the current value of the issue's '**Status**' field ('**OPEN**', '**IN PROGRESS**', etc).

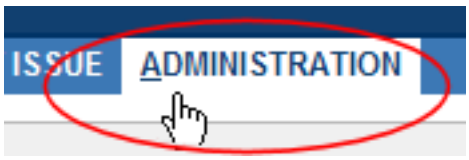
So if you need your workflow to force an issue to be 'open' or 'closed', you will need to set the issue's '**Resolution**' field during a transition. There are two ways to do this:

- Set the '**Resolution**' field automatically via a post function.
- Prompt the user to choose a '**Resolution**' via a screen.

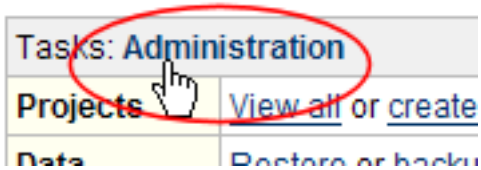
1.9.1.5. Adding a step

To add a new step to a workflow:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow to which you wish to add a step.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**:

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Open	Start Progress (4) >> In Progress Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
In Progress (3)	In Progress	Stop Progress (301) >> Open Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
Resolved (4)	Resolved	Close Issue (701) >> Closed Reopen Issue (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties
Reopened (5)	Reopened	Resolve Issue (5) >> Resolved Close Issue (2) >> Closed Start Progress (4) >> In Progress	Add Transition | Delete Transitions | Edit | View Properties
Closed (6)	Closed	Reopen Issue (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties

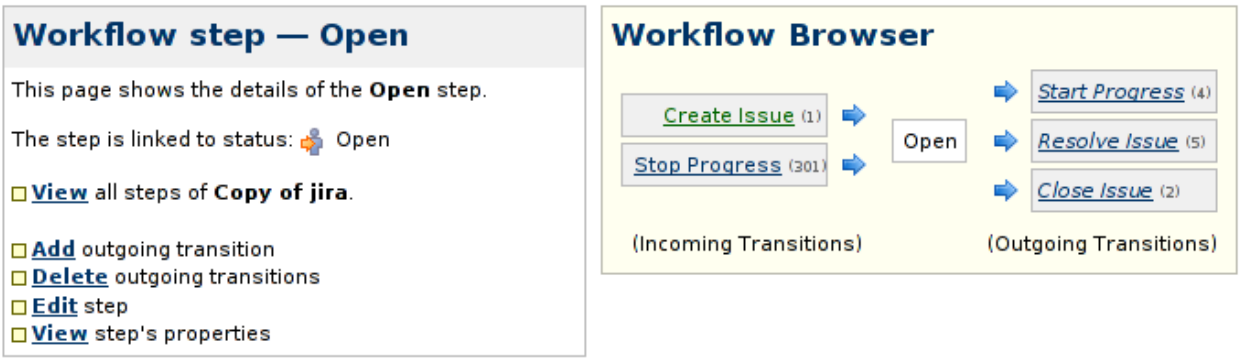
Workflows Steps page

The '**Add New Step**' form appears below the list of steps. (Note: this form will only be shown if the workflow is inactive or you are editing an active workflow.)

6. In the '**Step Name**' field, type a short name for the step. (Note: it is often useful to use the name of the corresponding status.)
7. In the '**Linked Status**' field, select the status that corresponds to this step. Note that each status can only correspond to one step in each workflow, so if all the statuses are already linked to steps in this workflow, you may need to define a new status.
8. Click the '**Add**' button. The '**Workflow Steps**' page will now show your new step in the list.
9. If you wish to view the details of your new step, click the step name. The '**View Workflow**

Step' page will be displayed, showing the step's:

- **Linked Status** ('**Open**' in the screenshot below).
- **Incoming Transitions** — that is, transitions whose **Destination Step** is this step.
  - To allow issues to move into this step, there must be at least one incoming transition.
- **Outgoing Transitions** — that is, transitions whose **Originating Step** is this step.
  - To allow issues to move out of this step, there must be at least one outgoing transition.



Viewing a workflow step

From this page you can:

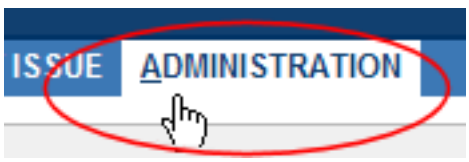
- Edit the step's **Name** or **Linked Status**, by clicking the '**Edit**' link.
- View and edit the step's **Properties** (see *'Using step properties*' below).
- View and edit any of the step's **Incoming Transitions** or **Outgoing Transitions**, by clicking the name of a transition. See *'Adding a condition*', *'Adding a validator*' and *'Adding a post function*' (below).
- Add an **Outgoing Transition** to the step (see *'Adding a transition*' below).
- Delete an **Outgoing Transition**.

Using step properties

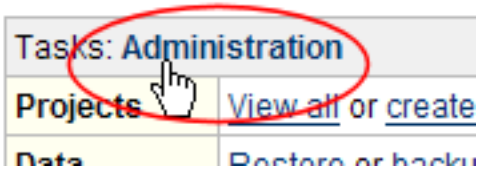
You can use step properties to prevent issues from being edited when they are in a particular workflow step(s). For example, in the default JIRA workflow, issues in the '**Closed**' step/status cannot be edited, even by users who have the '**Edit Issue**' permission. Note that issues which cannot be edited cannot be updated using Bulk Edit either.

To stop issues from being editable in a particular step, set the '**jira.issue.editable**' property of the step to '**false**' as follows:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.

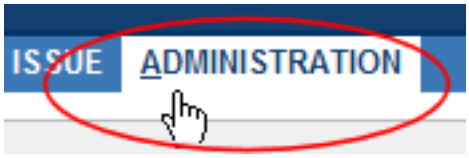
4. The '**View Workflows**' page will be displayed as shown in *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow whose step you wish to make uneditable.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow.
6. Click the '**View Properties**' link that corresponds to the relevant step.
7. The '**View Workflow Step Properties**' page will be displayed, showing the step's existing properties (if any). The '**Add New Property**' form appears below the list of steps. (Note: this form will only be shown if the workflow is inactive or you are editing an active workflow.)
8. In the '**Property Key**' field, type: `jira.issue.editable`.
9. In the '**Property Value**' field, type: `false`.
10. Click the '**Add**' button.

1.9.1.6. Deleting a step

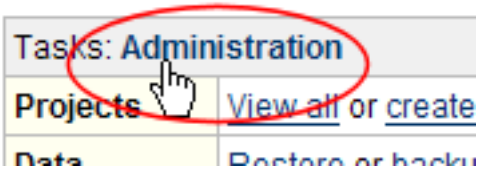
Note: a step can only be deleted if it has no incoming transitions.

To delete a step from a workflow:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



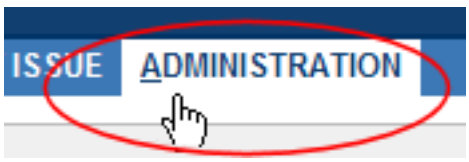
Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow from which you wish to delete a step.
5. The '**Workflow Steps**' page will be displayed.
6. Click the '**Delete**' link that corresponds to the relevant step. (Note: this link will only be shown if the step has no incoming transitions. A workflow step cannot be deleted if it is the destination of a transition.)

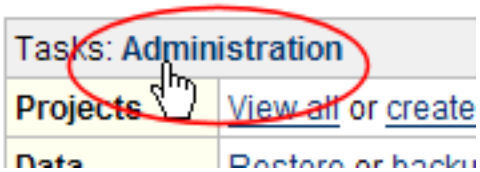
1.9.1.7. Adding a transition

To add a new transition to a workflow:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



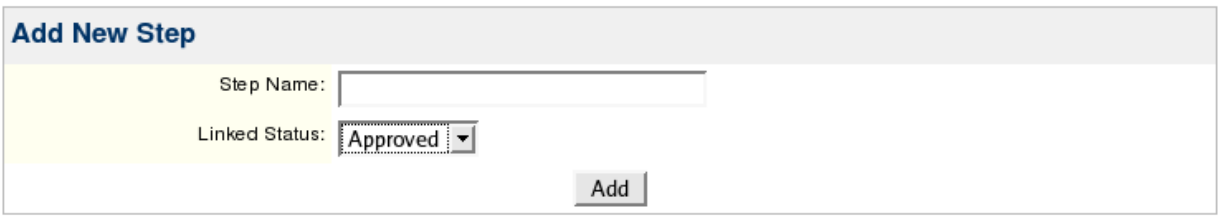
Link to Administration section



Link to Administration section

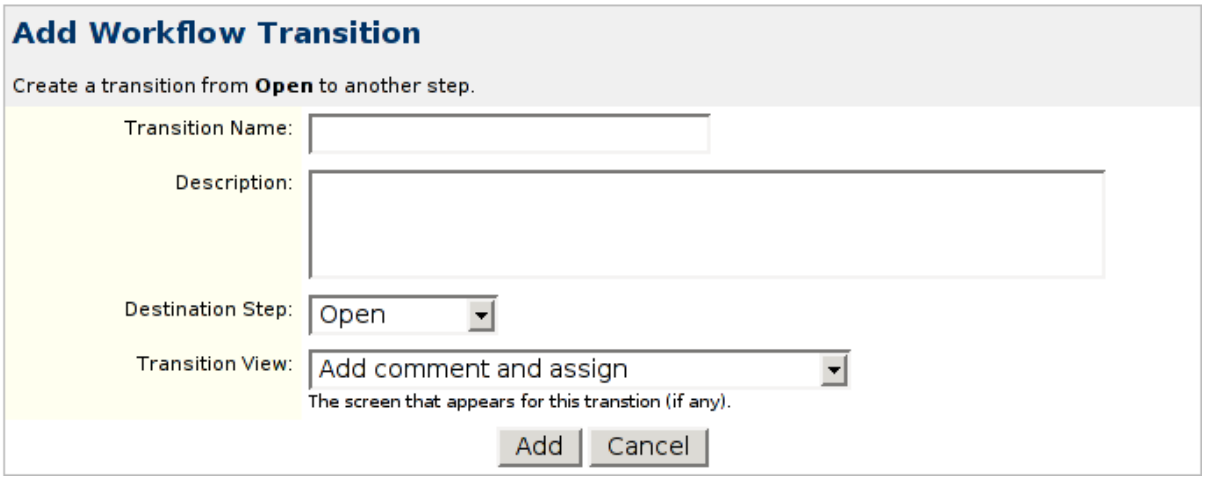
3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow to which you wish to add a transition.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**:

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Open	Start Progress (4) >> In Progress Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
In Progress (3)	In Progress	Stop Progress (301) >> Open Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	Add Transition | Delete Transitions | Edit | View Properties
Resolved (4)	Resolved	Close Issue (701) >> Closed Reopen Issue (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties
Reopened (5)	Reopened	Resolve Issue (5) >> Resolved Close Issue (2) >> Closed Start Progress (4) >> In Progress	Add Transition | Delete Transitions | Edit | View Properties
Closed (6)	Closed	Reopen Issue (3) >> Reopened	Add Transition | Delete Transitions | Edit | View Properties



Workflows Steps page

6. Identify the step from which your new transition will originate, and click the '**Add Transition**' link next to the step. The '**Add Workflow Transition**' page will be displayed:



Add Transition page

7. In the '**Transition Name**' field, type a short name for the transition. (Note: this name will be shown to users as the transition link in the list of '**Available Workflow Actions**' on the '**View Issue**' page.)
8. *(Optional)* In the '**Description**' field, type a short description of the purpose of the transition.
9. In the '**Destination Step**' field, choose the step to which issues will move when this transition is executed.
10. In the '**Transition View**' field, choose either:
    - '**No view for transition**' — choose this if you don't need to prompt the user for input before the transition is executed (i.e. the transition will occur instantly when the user clicks the transition link).
    - The name of a screen that will be shown to users, asking for input before the transition is executed. You can choose one of JIRA's default screens (note: many of these are used in the default workflow and are named after its transitions, e.g. '**Start Progress**' and '**Resolve Issue**'), or any other screen you have created. If no existing screen is suitable, you may want to create a new screen.

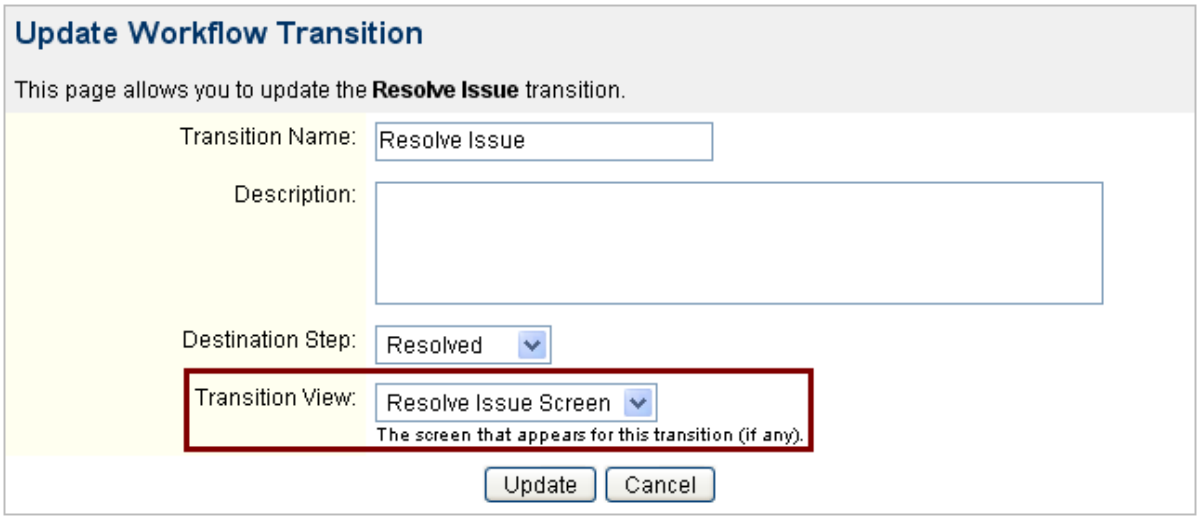
Using a screen

You can use a screen to gather input from a user before a particular transition is executed.

Example: using a screen to set the 'Resolution' field

For a particular step in a workflow, you might need to create a transition that will move the issue to a 'closed' status (e.g. '**CLOSED**', '**RESOLVED**', etc) — see 'open' and 'closed' issues. As part of this transition, you might need the user to set the '**Resolution**' field. To do this:

1. Create a screen, e.g. named '`Resolve Issue Screen`', that contains the '**Resolution**' field (and any other fields you want to display).
2. Create/edit your transition, and choose '`Resolve Issue Screen`' in the '**Transition View**' field:



Resolution Transition

Adding a condition

Conditions control who can perform a transition, and under what circumstances. If a condition fails, the user won't see the transition link on the *View Issue* page.

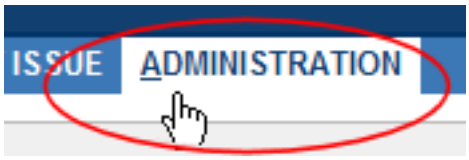
JIRA ships with the following built-in conditions, which are available for you to add to transitions:

Condition	Explanation
Only Assignee Condition	Only allow the issue's current assignee to execute the transition.
Only Reporter Condition	Only allow the issue's reporter to execute the transition.
Permission Condition	Only allow users with a given permission to execute the transition.
Sub-Task Blocking Condition	Block the parent issue transition depending on sub-task status.
User Is In Group	Only allow users in a given group to execute the transition.
User Is In Group Custom Field	Only users in a given custom field (of type "Group") to execute a transition.
User Is In Project Role	Only allow users in a given project role to execute a transition.

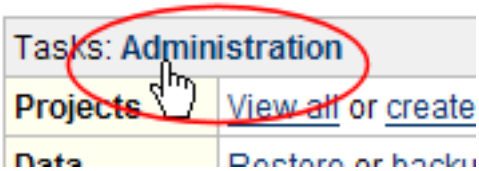
(You can also create your own conditions via the plugin system. See the *Workflow Plugin Guide* for details.)

To add a condition to a transition:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

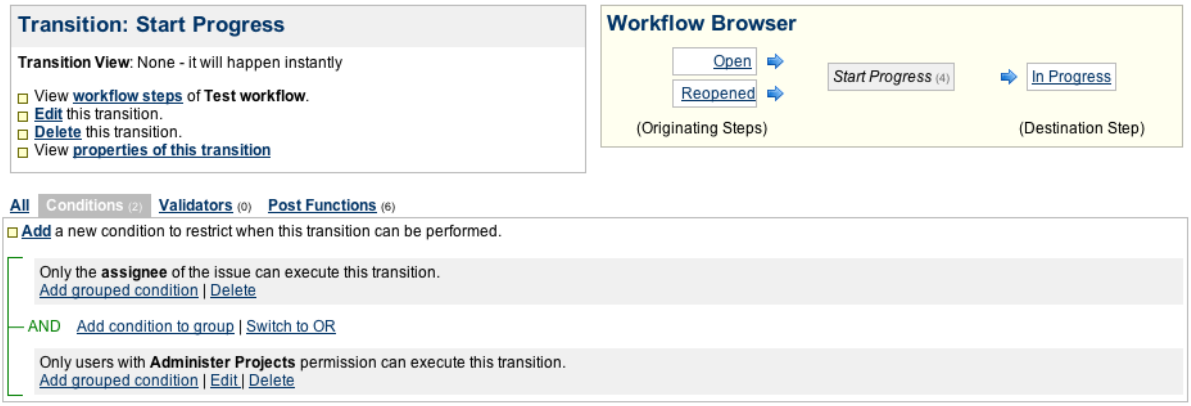


Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow to which you wish to add a condition.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**.
6. Click the name of the transition to which you wish to add a condition. The '**View Workflow Transition**' page will be displayed:



View Workflow Transition screen

7. Click the '**Conditions**' tab. A list of the transition's existing conditions will be displayed.
8. Click the '**Add**' link. A list of all available conditions will be displayed.
9. Select a condition from the list and click the '**Add**' button.
10. If the condition requires one or more configuration parameters (e.g. the name of a group or project role), the '**Add Parameters To Condition**' page will be presented. Enter your criteria and click the '**Add**' button.
11. The '**Conditions**' tab will be displayed, showing your new condition at the bottom of the list of conditions.

    Note: from here you can:

    - Click the '**Edit**' link next to the condition's name to edit its configuration parameters (if there are any).
    - Click the '**Delete**' link next to the condition's name to remove the condition
    - Combine your conditions into 'AND'/'OR' groups (see below).

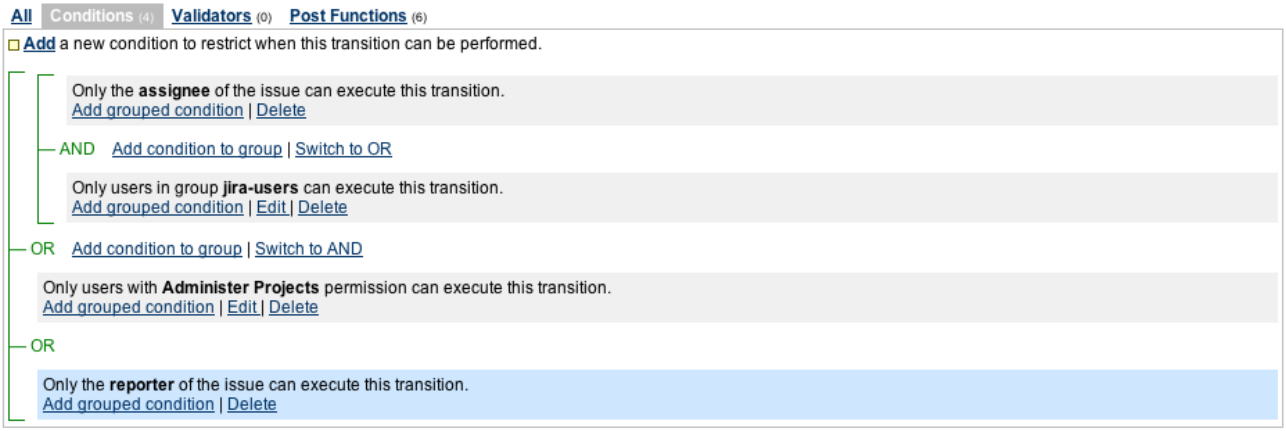
Combining conditions into groups

You can construct complex conditions by combining individual conditions together to form 'condition groups', using a boolean *AND* or *OR*. For example, the following condition group could be constructed:

- Only the assignee of this issue can execute this transition
- **AND**
- Only users in group **jira-users** can execute this transition

The condition will pass if the user is the assignee of the issue **AND** the user is in the group **jira-users**.

Multiple condition groups can be combined to construct even more complex conditions. Each pair of condition groups can be combined using a boolean **AND** or **OR**. Depending on the structure of the overall condition and its groups, the condition will pass once one or all condition groups have been satisified, e.g:



View Workflow Transition Condition Groups screen

Adding a validator

Validators check that any user-supplied input is valid before performing the transition. For example, a validator can be used to ensure that the comment entered by a user on the transition's screen meets a certain criteria. If a validator 'fails', the Post Functions of the transition will not be executed and the issue will not progress to the destination step of the transition.

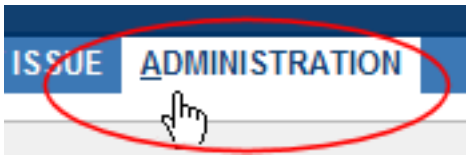
JIRA ships with a number of default validators, which are available for you to add to your transitions. You can also create your own validators via the plugin system.

Note:

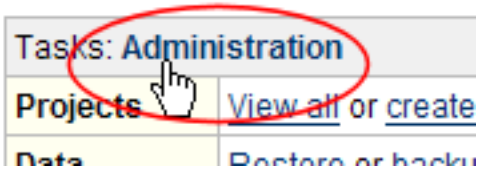
What is the difference between conditions and validators? Conditions are used to determine whether a transition is 'allowed' to be executed. Conditions cannot validate input parameters that are provided by the user on the transition's screen, since if the condition fails the user is not allowed to start executing the transition and so will not see the transition's screen. Validators have access to the input that has been gathered from the user via the transition's screen, and thus can validate that input.

To add a validator to a transition:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above.

Click the '**Steps**' link next to the workflow to which you wish to add a condition.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**.
6. Click the name of the transition to which you wish to add a validator. The '**View Workflow Transition**' page will be displayed.
7. Click the '**Validators**' tab.
8. Click the '**Add**' link. A list of all available validators will be displayed.
9. Select a validator from the list and click the '**Add**' button.
10. If the validator requires one or more configuration parameters (e.g. the name of a group or project role), the '**Add Parameters To Validator**' page will be presented. Enter your criteria and click the '**Add**' button.
11. The '**Validators**' tab will be displayed, showing your new validator at the bottom of the list of validators.

    Note: from here you can:

    - Click the '**Edit**' link next to the validator's name to edit its configuration parameters (if there are any).
    - Click the '**Delete**' link next to the validator's name to remove the validator.

Adding a post function

Post functions carry out some processing immediately after a transition is executed (hence the name *post* function), such as updating an issue's fields, generating change history for an issue, adding a comment to an issue, generating an event (e.g. an email notification).

The JIRA default workflow includes a number of default transitions. Additionally, JIRA ships with the following 'essential' post functions, which are automatically added to every newly-created transition:

Essential post function
Set issue status to the linked status of the destination workflow step.
Add a comment to an issue if one is entered during a transition.
Update change history for an issue and store the issue in the database.
Re-index an issue to keep indexes in sync with the database.
Fire an event that can be processed by the listeners.

The 'essential' post functions cannot be deleted, or reordered relative to each other, as this could compromise other functionality. However, you can insert other post functions between them.

JIRA ships with four built-in post functions which you can optionally add to your transitions:

Optional post function	Explanation
Assign to Current User	Assigns the issue to the user who is executing the transition. (Note: This post function will be ignored unless the user has the '**Assignable User**' permission. You may want to use a condition to ensure that the logged-in user has this permission before executing the transition.)
Assign to Lead Developer	Assigns the issue to the component lead (if one exists) or project lead.
Assign to Reporter	Assigns the issue to the user who created the issue.
Update Issue Field	Updates one of the issue's fields to a given

	value. Updateable fields are: • '**Assignee**' • '**Description**' • '**Environment**' • '**Priority**' • '**Resolution**' • '**Summary**' • '**Original Estimate**' • '**Remaining Estimate**' Note that this post function cannot update custom fields.

(You can also create your own post functions via the plugin system. See the *Workflow Plugin Guide* for details.)

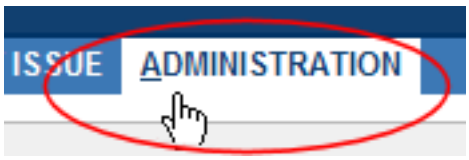
Note that the four optional post functions must be positioned *before* the '**Update change history for an issue and store the issue in the database**' post function, except when used in the '**Create**' transition.

Note:

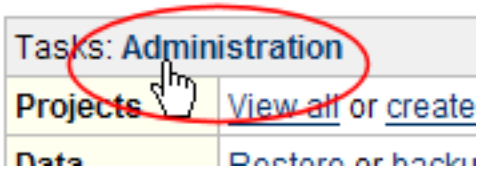
A note regarding the 'Create' transition:
Sometimes it is useful to perform particular processing (e.g. set a particular field) when an issue is first created. You can do this by adding post functions to the workflow's 'initial transition', which is executed whenever a user creates an issue, and puts the newly-created issue into the workflow's 'initial step'. The 'initial step' is simply the first step in a workflow; every workflow has one, and only one, initial step (called '**Open**' by default, i.e. if you created a blank workflow or copied the default workflow). The 'initial transition' (called '**Create**' by default) is the first incoming transition of the 'initial step'.
When adding one of the optional post functions to the workflow's '**Create**' transition (e.g. you might use the '**Update Issue Field**' transition to set the '**Assignee**' field to a particular user when an issue is created), note that you need to put it *before* the '**Create**' transition's default '**Creates the issue originally**' post function.
Special case:
If you need to set the '**Resolution**' field when creating an issue, put the '**Update Issue Field**' post function *after* the default '**Creates the issue originally**' post function, and use the '**Issue Store**' post function after that. Note that use of the '**Issue Store** post function (which is available only for the '**Create**' transition) should be kept to a minimum, as it does not generate change history and is incapable of persisting fields that have a one-to-many relationship with the issue (e.g. '**Version**' or '**Component**'). However, for setting the '**Resolution**' field during issue creation, this post function is useful.

To add a post function to a transition:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



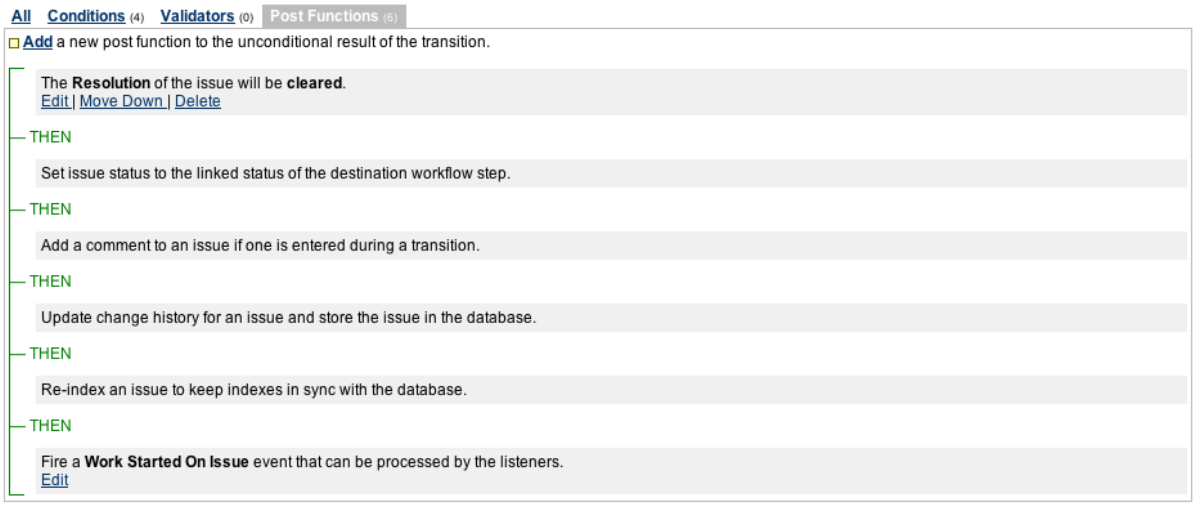
Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow to which you wish to add a condition.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**.
6. Click the name of the transition to which you wish to add a post function. The '**View Workflow Transition**' page will be displayed.

7. Click the '**Post Functions**' tab. A list of the transition's existing post functions (if any) will be displayed. For example, the default workflow has the following built-in post functions for the '**Start Progress**' transition:



View Workflow Transition Post-Functions screen

8. Click the '**Add**' link. A list of all available post functions will be displayed.
9. Select a post function from the list and click the '**Add**' button.
10. If the post function requires one or more configuration parameters (e.g. the name of an event), the '**Add Parameters To Post Function**' page will be presented. Enter the appropriate information and click the '**Add**' button.
11. The '**Post Functions**' tab will be displayed, showing your new post function at the bottom of the list of post functions.

    Note: from here you can:

    - Click the '**Edit**' link next to the post function's name to edit its configuration parameters (if there are any).
    - Click the '**Delete**' link next to the post function's name to remove the post function.
    - Click the '**Move Up**' link to move the post function higher up in the list (i.e. it will be executed earlier).
    - Click the '**Move Down**' link to move the post function lower down in the list (i.e. it will be executed later).

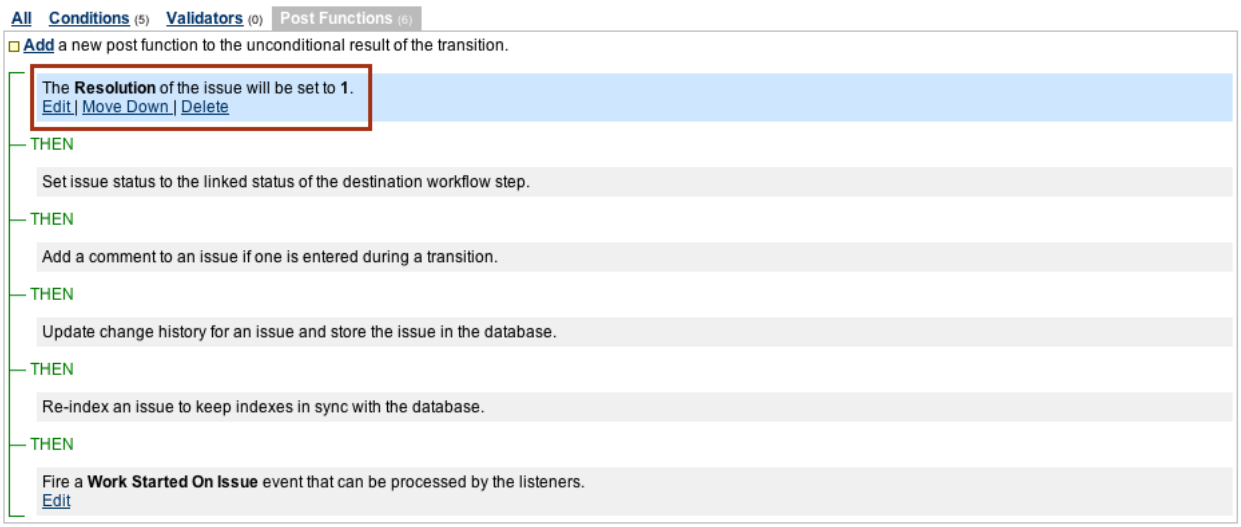
Using a post function to set a field

You can use a post function of type '**Update Issue Field**' to set the value of an issue's field(s) after a particular transition is executed.

Example: using a post function to set the 'Resolution' field

For a particular step in a workflow, you might need to create a transition that will move the issue to a 'closed' status (e.g. '**CLOSED**', '**RESOLVED**', etc) — see 'open' and 'closed' issues. As part of this transition, you might want to automatically set the '**Resolution**' field. To do this:

1. Create/edit your transition. In the '**Transition View**' field, either select '**No View For Transition**' or choose a screen that does not contain the '**Resolution**' field (e.g. the '**Add Comment And Assign**' screen).
2. Add a new post function of type '**Update Issue Field**'. Select '**Resolution**' from the '**Issue Field**' select list and select a suitable resolution from the '**Field Value**' select list.
3. Once completed, the transition's list of post functions will appear as follows:



Resolution Post Function

To create a transition that unsets the '**Resolution**' field, follow the same steps but select '**None**' from the '**Field Value**' select list when adding the post function. The list of post functions for this transition will include the following statement:

The Resolution of the issue will be cleared.

Each time one of these transitions is executed, the '**Resolution**' of the issue is automatically set or unset as specified in these post functions.

Using a post function to send a notification

You can use a post function of type '**Fire an event that can be processed by the listeners**' to fire the '**Generic Event**'. The '**Generic Event**' is a built-in JIRA event whose purpose is to allow you to to send email notifications after a particular transition is executed.

Alternatively, you could fire a custom event that you have created specifically for this transition.

When a transition is performed, JIRA will:

- Look up the notification scheme associated with the issue's project, and identify the users associated with the fired event;
- Send an email notification to each user.

(Note that the fired event is also propagated to all registered listeners.)

Example: using a post function to fire the 'Generic Event'

You can use the '**Generic Event**' to send email notifications. To do this:

1. Create/edit your transition.
2. Go to the transition's '**Post Functions** tab and edit the '**Fire an event that can be processed by the listeners**' post function.
3. On the '**Add Parameters To Post Function**' page, select '**Generic Event**' from the list of events.

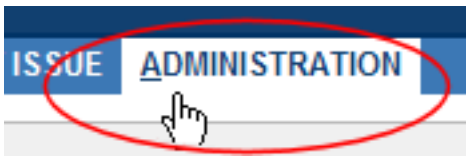
Working with transition properties

Properties are key-value pairs that are can be used to further customise transitions. For example, transition properties help to extend the default workflow to allow language translations.

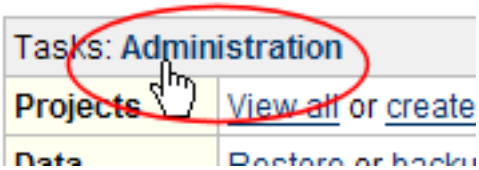
To view the properties of a transition:

1. Log in as a user with the '**JIRA Administrators**' global permission.

2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown under *'Creating a workflow'* above. Click the '**Steps**' link next to the workflow to which you wish to add a condition.
5. The '**Workflow Steps**' page will be displayed, showing the steps that make up the workflow, and each step's **Linked Status** and **Outgoing Transitions**.
6. Click the name of the transition for which you wish to view the properties. The '**View Workflow Transition**' page will be displayed.
7. Click the '**View properties of this transition**' link. The '**View Workflow Transition Properties**' page will display listing the properties currently set up for the transition. You can also add and delete properties for this transition on this page.

1.9.1.8. Using 'common transitions'

A 'common transition' is a transition that is defined only once in the workflow, but can be used more than once. That is, a common transition can have more than one originating step. The advantage of common transitions is that if a transition needs to be updated, the update only has to be done in one place.

You can edit common transitions in JIRA, but they cannot be created by the method described in *'Adding a transition'* (above). Instead, to create common transitions, you can either:

- **Copy the default workflow** — the default workflow contains common transitions. Although you cannot edit the default workflow, you can copy it and then edit its steps and transitions to suit your requirements.
- **Create your workflow in XML** — see '*Using XML to create a workflow*' (below).

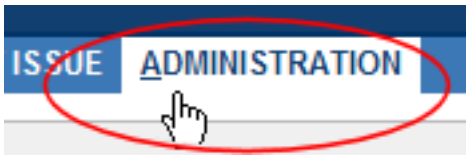
1.9.1.9. Using XML to create a workflow

JIRA uses OSWorkflow, a flexible and customisable workflow engine. JIRA's workflow editor generates OSWorkflow XML definition files that are stored in JIRA's database. If you need to take advantage of some OSWorkflow feature that is not available in JIRA's workflow editor (such as 'common' transitions — see above), you can define the workflow in XML and then import it into JIRA as described below.

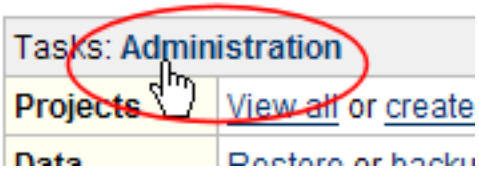
Once the XML workflow has been imported, JIRA's workflow editor should be able to display most OSWorkflow definitions even if it does not support creating or editing them. For example, conditional results of workflow transitions are displayed in the '**Other**' tab on the '**View Workflow Transition**' page. The '**Other**' tab is only visible if a transition has elements that the editor does not directly support.

To import an XML workflow into JIRA:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under '**Global Settings**', click the '**Workflows**' link.
4. The '**View Workflows**' page will be displayed as shown in *'Creating a Workflow'* (above). Locate the '**Add New Workflow**' form at the bottom of the page.
5. Click the '**Import a workflow from XML**' link. The '**Import Workflow**' page will be displayed.
6. In the '**Name**' field, type a name (usually 2-3 words) to identify your new workflow.
7. *(Optional)* In the '**Description**' field, type a detailed description of your new workflow.
8. Either:
   - In the '**Workflow Definition (XML)**' field, paste the contents of the workflow XML file; or
   - In the '**File**' field, type the full path to the file (note that the path must be local, i.e. you will need to first copy the file to your JIRA server).
9. Click the '**Import**' button.

Copying a workflow between systems

Sometimes it is useful to create a workflow in a test system and then copy it into a production system. To do this:

1. In the test system, export the workflow to XML by clicking the '**XML**' link next to the workflow in the list shown on the '**View Workflows**' page, and save the output into a file.
2. In the production system, import the file by clicking the 'Import a workflow from XML' link as described in '*Using XML to create a workflow*' (above).

Warning:

When importing an XML workflow into JIRA:
JIRA's XML workflow definitions contain references to JIRA meta attributes. For example, the id of the linked JIRA status of each workflow step is stored as a 'jira.status.id' meta attribute in the step's definition. Therefore, when manually creating workflows in XML, please ensure that all referenced external entities exist before you import the workflow into JIRA.

Warning:

When copying a workflow between systems:
Please note that conditions, validators and post functions can have parameters that might be valid in one system and not in another. For example, different systems might contain different sets of values for the '**Resolution**' field (since it is possible to define your own values). This would be a problem if the '**Update Issue Field**' post function is used to set the '**Resolution**' field to a value that exists in one system but not the other.

1.9.2. Activating Workflow

Once you have created a new workflow or modified an inactive workflow, you will need to *activate* it.

The method of activating workflows depends on the edition of JIRA:

- **Standard** edition only supports the default workflow, which is always active.
- **Professional** edition can only have one active workflow in the system at a time.
- **Enterprise** edition supports multiple active workflows through workflow schemes. Workflow schemes associate particular workflows with particular projects and (optionally) particular issue types. Therefore it is possible to use a different workflow for every project/issue type combination, if you wish.

Some terminology:

- "**Active**" workflows are those that are currently being used.
- "**Inactive**" workflows are those that:
  - *in Professional edition*, are not the active workflow for the system.
  - *in Enterprise edition*, are not associated with any workflow schemes, or are associated with workflow schemes that are not associated with any projects.

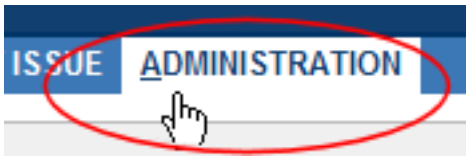
Note:

Please note that if you edit an active workflow, it does not need to be re-activated after your changes. Read more about editing active workflows.

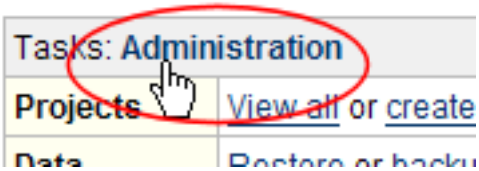
1.9.2.1. Activating a workflow in JIRA Professional Edition

To activate a workflow in JIRA Professional edition:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under "**Global Settings**", click the "**Workflows**" link.
4. The "**Workflows**" page will be displayed, showing a list of all existing workflows in your system. Click the "**Activate**" link next to the workflow that you wish to activate.
5. Follow the activation wizard, which will guide you through migrating all existing issues to this workflow.

Name	Description	Active?	Steps	Operation
jira (Read-only System Workflow)	The default JIRA workflow.	Active	5	Steps | XML | Copy
Copy of jira	Workflow for testing	Inactive	5	Steps | XML | Copy | Delete | Activate

Activate Workflow in JIRA Professional screen

1.9.2.2. Activating workflows in JIRA Enterprise Edition

To activate a workflow in JIRA Enterprise edition, you need to:

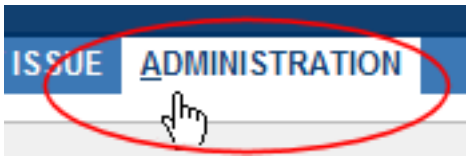
1. Create a Workflow Scheme that references your workflow, and (optionally) associate it with the relevant issue type(s).

2. Associate the Workflow Scheme with the relevant project(s).

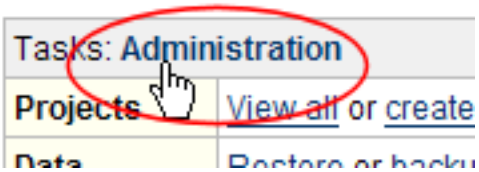
Creating a workflow scheme

To create a workflow scheme:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

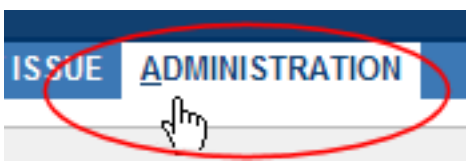
3. In the left-hand navigation panel, under "**Schemes**", click the "**Workflow Schemes**" link.
4. The "**Workflow Schemes**" page will be displayed, showing a list of all existing workflow schemes in your system. Click the "**Add Workflow Scheme**" link.
5. The "**Add Workflow Scheme**" page will be displayed. Type a **Name** and (optionally) a short **Description** for the new workflow scheme, then click the "**Add**" button. This will create the scheme.
6. The "**Edit Workflows**" page will be displayed, showing your newly-created scheme. Click the "**Assign a workflow to an issue type**" link.
7. The "**Add Workflow To Scheme**" page will be displayed.
   - In the "**Issue Type**" drop-down list, select an issue type that is relevant to your workflow. Note: you can also select "All Unassigned Issue Types" to associate your workflow with all issue types that do not have a specific association in this workflow scheme.
   - In the "**Workflow**" drop-down list, select the name of your new workflow.
   - Click the "**Add**" button.
8. Repeat the previous step until your new workflow has been associated with all the relevant issue types. Note that you can choose different workflows for some issue types if you wish.

Once a Workflow Scheme is fully defined you need to associate it with one or more projects (see below) so that the scheme's workflows are actually used by your JIRA system.

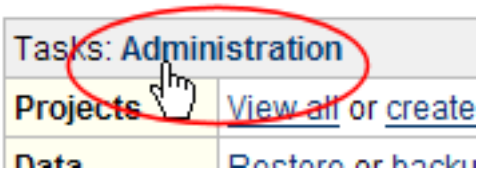
Associating a workflow scheme with a project

To associate a workflow scheme with a project:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

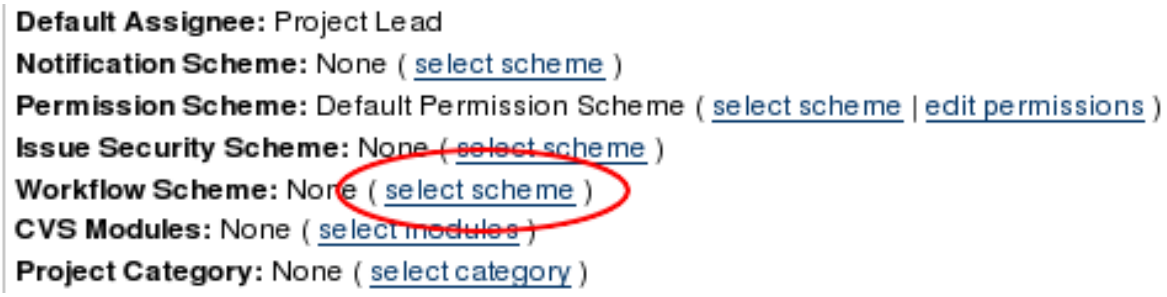


Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, click the "**Projects**" link.
4. A list of projects will be displayed. Click the name of the project in which you are interested.
5. The "**Administer Project**" page will be displayed. Click the "Select" link next to the "Workflow Scheme" property of the project.



Select Workflow Scheme

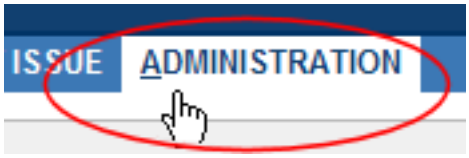
6. The "**Associate Workflow Scheme to Project**" page will be displayed. Select the relevant scheme from the list and click the "**Associate**" button.
7. Follow the wizard, which will guide you through migrating all the project's issues to the new scheme's workflows.

Note:
You can associate a workflow scheme with more than one project.

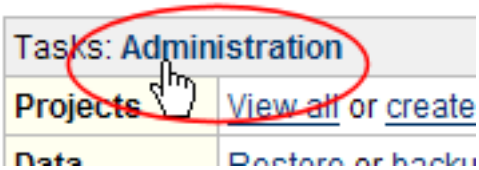
Editing a workflow scheme

To edit a workflow scheme, i.e. to change which workflows are associated with which issue types:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, under "**Schemes**", click the "**Workflow Schemes**" link.
4. The "**Workflow Schemes**" page will be displayed, showing a list of all existing workflow schemes in your system. Click the "**Edit**" link next to the workflow scheme in which you are interested.
5. The "**Edit Workflows**" page will be displayed.

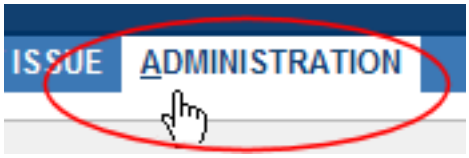
- To associate a workflow with an issue type, click the "**Assign a workflow to an issue type**" link.
- To disssociate a workflow from an issue type, click the "**Delete**" link.

Note:
It is not possible to edit an active workflow scheme, that is, a workflow scheme that is currently associated with one or more projects. Instead: copy it, edit the new copy and then associate all the relevant projects with the new copy.

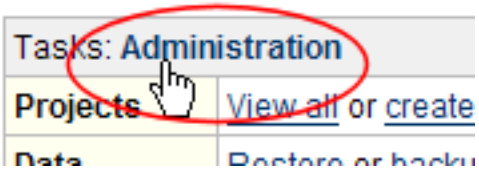
Disassociating a workflow scheme from a project

To disassociate a workflow scheme from a project:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

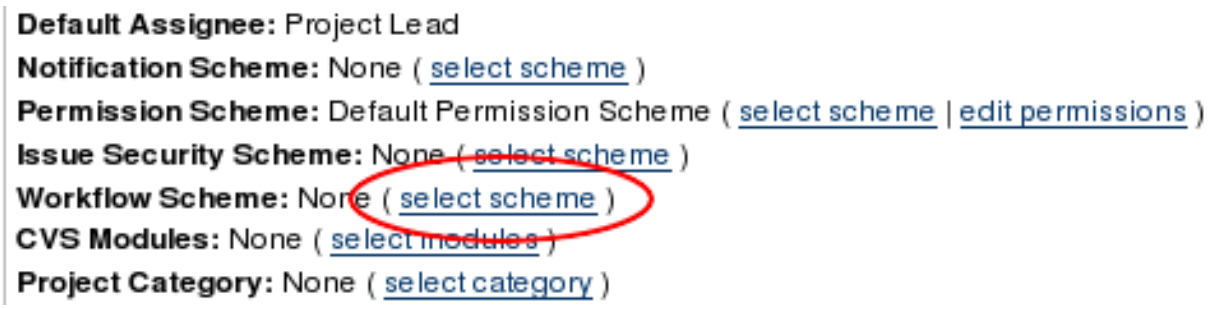


Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, click the "**Projects**" link.
4. A list of projects will be displayed. Click the name of the project in which you are interested.
5. The "**Administer Project**" page will be displayed. Click the "Select" link next to the "Workflow Scheme" property of the project.



Select Workflow Scheme

6. Select "**None**" from the presented list and click the "**Associate**" button.
7. Follow the wizard, which will guide you through migrating all of the project's issues to the default workflow.

Note:
All projects that do not have an associated workflow scheme use JIRA's default workflow.

1.9.2.3. Additional Resources

- Workflow scheme overview tutorial video — Watch this short tutorial video to see how to create a new workflow scheme and associate it to a project in JIRA. Please note the JIRA version and JIRA edition of the tutorial video before watching.

1.9.3. JIRA Events

JIRA uses an event-listener mechanism to alert the system that something has happened, and to peform appropriate action (e.g. send an email notification) based on the event that has occurred. Every *issue operation* within JIRA is associated with a particular *event* — e.g. the `Issue Created` event is fired when an issue has been created.

A Listener can execute a specified action once it has been notified that a particular event has been fired. For example, the MailListener can send an `Issue Created` email to a list of recipients defined in the appropriate Notification Scheme, whenever an issue is created.

Some events are fired by JIRA internally — e.g. an `Issue Updated` or `Issue Moved` event. Other events are fired from within workflow transition post-functions — e.g. an `Issue Resolved` event, or a Custom Event (see below).

1.9.3.1. Event Types

There are two types of events within JIRA:

- **System** - System events are used throughout JIRA internally, and cannot be added or deleted. You can, however, make them Inactive (see below).
- **Custom** - *[JIRA Enterprise & Professional editions only]* Custom events are used to generate an email notification (or invoke a listener) from a particular workflow transition's post-function. You can add/delete as many custom events as you need. Note that only *inactive* custom events can be deleted.

An event can be in either of the following states:

- **Active** — the event is associated with at least one notification scheme or workflow transition post-function
- **Inactive** — the event is not associated with any notification schemes or workflow transition post-functions.

Note that the event state does not indicate whether the event is able to be fired. A **custom event** will only be fired if it is associated with a transition post-function for an active workflow (see 'Activating Workflow').

System events

JIRA's built-in system events are:

Issue Created:	An issue has been entered into the system.
Issue Updated:	An issue has had its details changed.
Issue Assigned:	An issue has been assigned to a new user.
Issue Resolved:	An issue has been resolved (usually after being worked on and fixed).
Issue Closed:	An issue has been closed. (Note that an issue may be closed without being resolved; see Statuses).
Issue Commented:	An issue has had a comment added to it.
Issue Comment Edited:	An issue's comment has been modified.
Issue Reopened:	An issue has been re-opened.
Issue Deleted:	An issue has been deleted.

Issue Moved:	An issue has been moved into this project.
Work Logged On Issue:	An issue has had hours logged against it (i.e. a worklog has been added).
Work Started On Issue:	The Assignee has started working on an issue.
Work Stopped On Issue:	The Assignee has stopped working on an issue.
Issue Worklog Updated:	An entry in an issue's worklog has been modified.
Issue Worklog Deleted:	An entry in an issue's worklog has been deleted.
Generic Event:	The exact nature of this event depends on the workflow transition post-function(s) which invoke it. As with Custom Events, you can use the Generic Event to generate an email notification (or invoke a listener) from a particular workflow transition's post-function (see Workflow and Notifications).

Custom Events

You can fire a **custom event** from a custom transition post-function in a custom workflow. The appropriate listeners will be alerted of the custom transition by the firing of this event. For example, the associated notification scheme can be configured to notify users of the workflow transition based on the firing of this custom event.

1.9.3.2. Configuring Notifications for a Custom Event

Custom events are most commonly used to generate notifications for custom workflow transitions. For example, your organisation might need you to modify the default workflow by adding a workflow step called "QA_Inspection" (e.g. between "Resolve Issue" and "Close Issue"). You would typically also need to generate an email notification to the QA team whenever an issue progresses to the "QA_Inspection" step of the workflow.

There are three overall steps to achieve this:

1. Add a custom event to the system (e.g. "Issue Awaiting QA").
2. Configure the notification scheme to send an email when the custom event is fired.
3. Configure the workflow transition post-function to fire the custom event.

Step 1. Add a Custom Event

1. Navigate to the **Administration** link from the main menu.
2. Select the **Events** link under the **Global Settings** section in the sub-menu.
3. Add a name and description for the new event.
4. Select a default email template to be associated with the event.
5. Click **Add**.

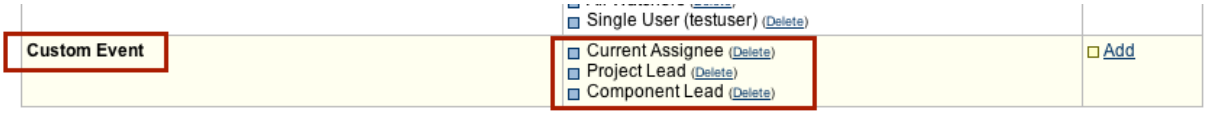
Add a Custom Event

The custom event must be associated with a default email notification template. A notification scheme configured to notify users of this event will use this email template when sending the notification.

The custom event will appear in the list of events defined within the system. Initially, the event will be marked as **inactive** as it is not associated with a notification scheme or workflow post-function.

Step 2. Configure Notification Scheme to send mail on Custom Event

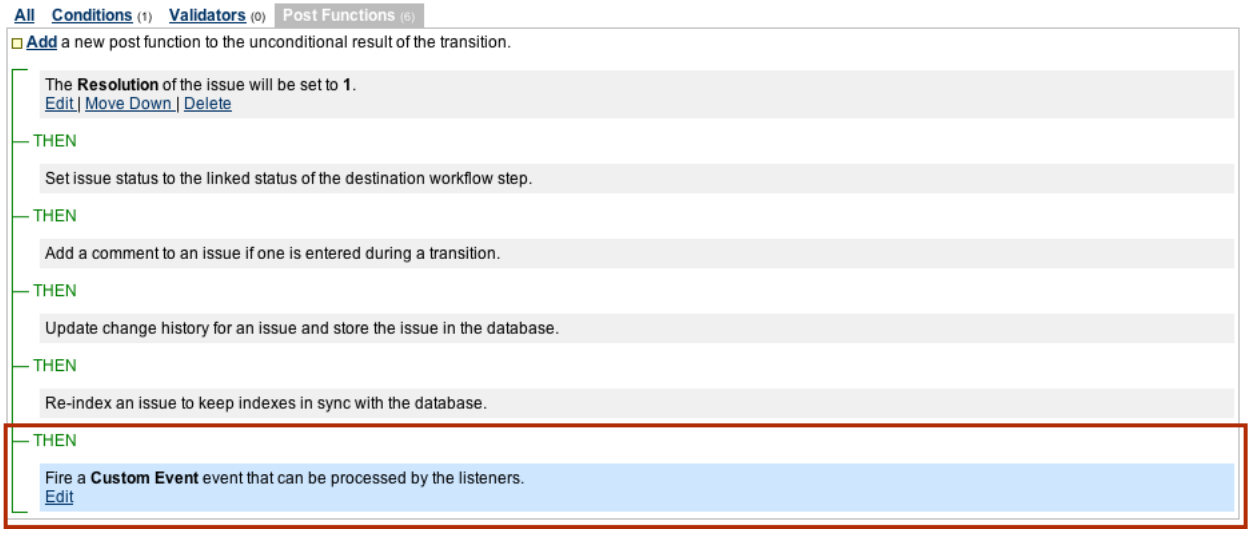
1. Navigate to the **Administration** link from the main menu.
2. Select the **Notification Schemes** link under the **Schemes** section in the sub-menu.
3. Select the notification scheme to edit.
4. Add the recipients for the custom event as required - further details available here.



Add Custom Event Recipients

Step 3. Configure Workflow Transition Post-Function to Fire Custom Event

1. Navigate to the **Administration** link from the main menu.
2. Select the **Workflows** link under the **Global Settings** section in the sub-menu.
3. Navigate to workflow transition post-function screen to be edited — further details available here.
4. Update the post-function to fire the custom event.
5. Activate or associate the workflow (and scheme) with the appropriate project (see 'Activating Workflow').



Fire Custom Event from Workflow Transition Post-Function

1.9.3.3. Updates to Workflows on Disk

As of JIRA 3.6, all event references are made through the `EVENT ID`. For pre-JIRA 3.6 data, all database tables (Workflow, Notification, etc.) are updated automatically. However, it is necessary to manually update event references in workflows saved to disk. This upgrade guide provides details on the changes required.

1.10. Importing from Other Systems

1.10.1. Importing Data from Bugzilla

JIRA can import your bugs from Bugzilla. Currently, the importer is compatible with Bugzilla 2.20 and above. Users of older Bugzilla versions will need to first upgrade the database tables to a supported version with Bugzilla's checksetup.pl script.

Note:

Due to a change to the database schema in Bugzilla versions 2.22.2 and later, attachments are not imported into JIRA. For details please see JRA-12389.

The data from the Bugzilla database is appended to the existing data in JIRA. The tool imports the following data from the Bugzilla database:

In Bugzilla	In JIRA	Special Notes
Bugs	Issues	• Attachments are extracted from the Bugzilla database and saved to disk. • Statuses, Bug Severity, Issue Types, and Resolutions in Bugzilla are mapped to the defaults in JIRA. • Statuses in Bugzilla are mapped to JIRA. Bugs in Bugzilla in the 'NEW', 'UNCONFIRMED', or 'ASSIGNED' status with no assignees are 'Open' in JIRA. • Issue Types of bugs from Bugzilla are all 'Bugs' or 'Improvements'. • Bug_Severity in Bugzilla is mapped to Priorities in JIRA. Bugs with 'ENHANCEMENT' severity in Bugzilla are treated as 'Improvement' issues with 'MINOR' Priority in JIRA. Note: if you have customized the Bugzilla list of priorities, you will need to edit the Importer source and define the new mappings. • The first description for a bug in Bugzilla is stored as JIRA's Description. All other descriptions are stored as comments logged to that issue. • If a user has voted one or more times for a Bugzilla issue, a JIRA vote is stored for that user.
Product	Project	• The project key and project lead can be set by the user.
Version	Version	• Versions for imported projects are imported from Bugzilla, and set to Un-Released and Un-Archived state. • The JIRA "Fix For" Version is

		set to the Bugzilla bug 'milestone', if it exists. Note: this code is not well tested - please let us know if you have problems.
Component	Component	
User	User	• Users are imported 'on demand', so users who have not interacted with the system in any way are not imported. • Passwords from Bugzilla are not imported for v2.16+ of Bugzilla (as they are hashed in the database). Users from Bugzilla will need to get their passwords emailed to them the first time they log into JIRA. • Users with no real name stored in Bugzilla will get the portion of their email address (login name) before the "@" character as their Full Name in JIRA. **Note:** If you are using External User Management, the import process will not be able to create JIRA users; instead, the importer will give you a list of any new users that need to be created. You will need to create the users in your external user repository before commencing the import (this way, votes etc can be imported correctly). **Note:** If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

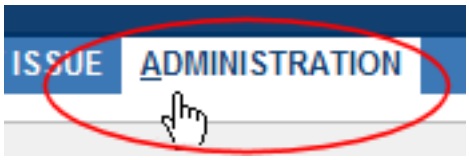
1.10.1.1. How to import from Bugzilla

Note: Before you begin, please backup your JIRA data.

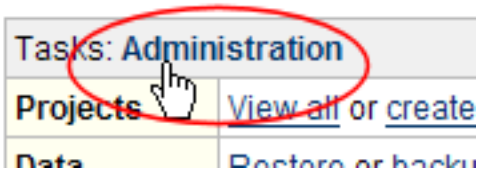
1. In your Bugzilla system, run the Bugzilla 'Sanity Check' to ensure your data is error-free.
2. Make sure that the Bugzilla database you wish to import from is running on MySQL.
3. Download and install the MySQL JDBC driver into JIRA. To do this, download the MySQL Connector/J driver. The package contains a file "mysql-connector-java-xxx.jar". Copy this to

the `common/lib/` directory in JIRA Standalone, or equivalent "lib" directory in your app server. Restart JIRA so the driver is loaded.

4. In JIRA's default permission scheme (associated with newly imported projects), ensure that the 'Browse', 'Create' and 'Comment' permissions are granted to the group 'jira-users' (or a group with the 'JIRA Users' global permission).
5. Log in as a user with the '**JIRA System Administrators**' global permission.
6. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

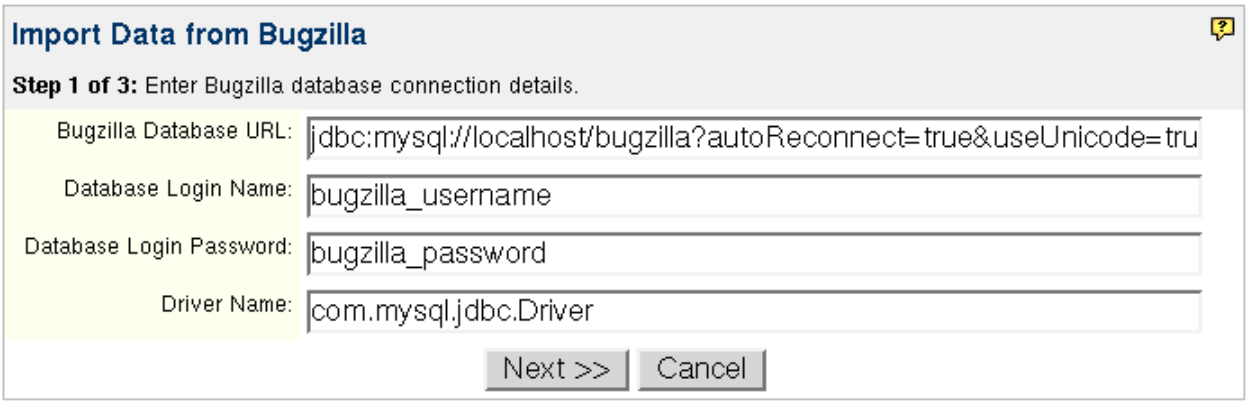


Link to Administration section

7. On the panel on the left, under the title 'Import & Export', click 'External System Import'

Link in Admin section

8. The 'Import Data' page will be displayed. Select 'Bugzilla'.
9. You will now be prompted for connection details to Bugzilla's MySQL database:



Bugzilla Importer Connection screen

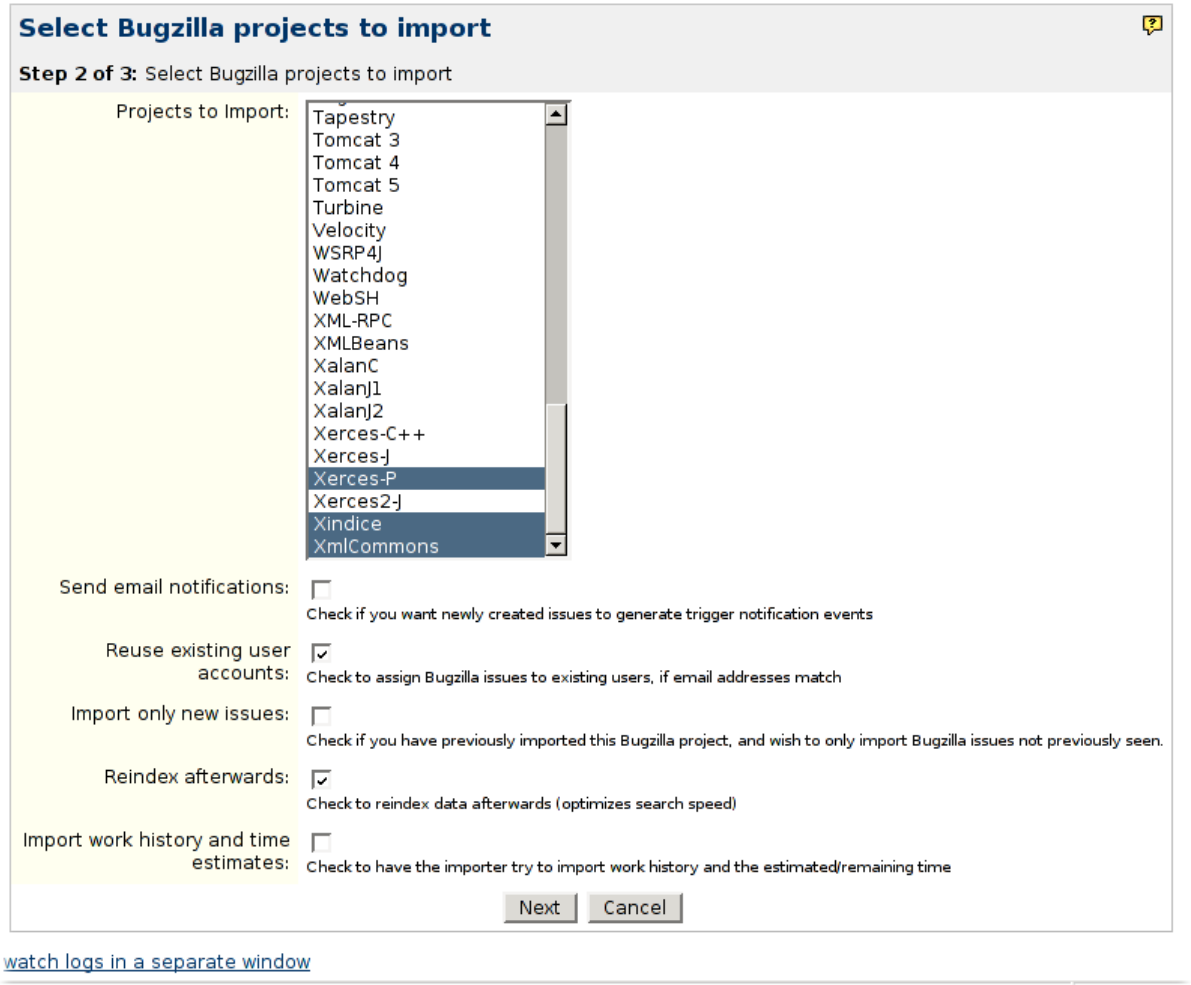
Set the JDBC URL, database username and password for your system. The JDBC URL is of the format:

```
jdbc:mysql:host[:port]/databasename?parameters
```

host is the server hosting JIRA, whose MySQL allows incoming TCP connections on port **port** (defaults to 3306). **databasename** is the MySQL database name (usually 'bugs').

The database name, username and user password can usually be found in the 'localconfig' file in Bugzilla's root directory, or in /etc/bugzilla/.

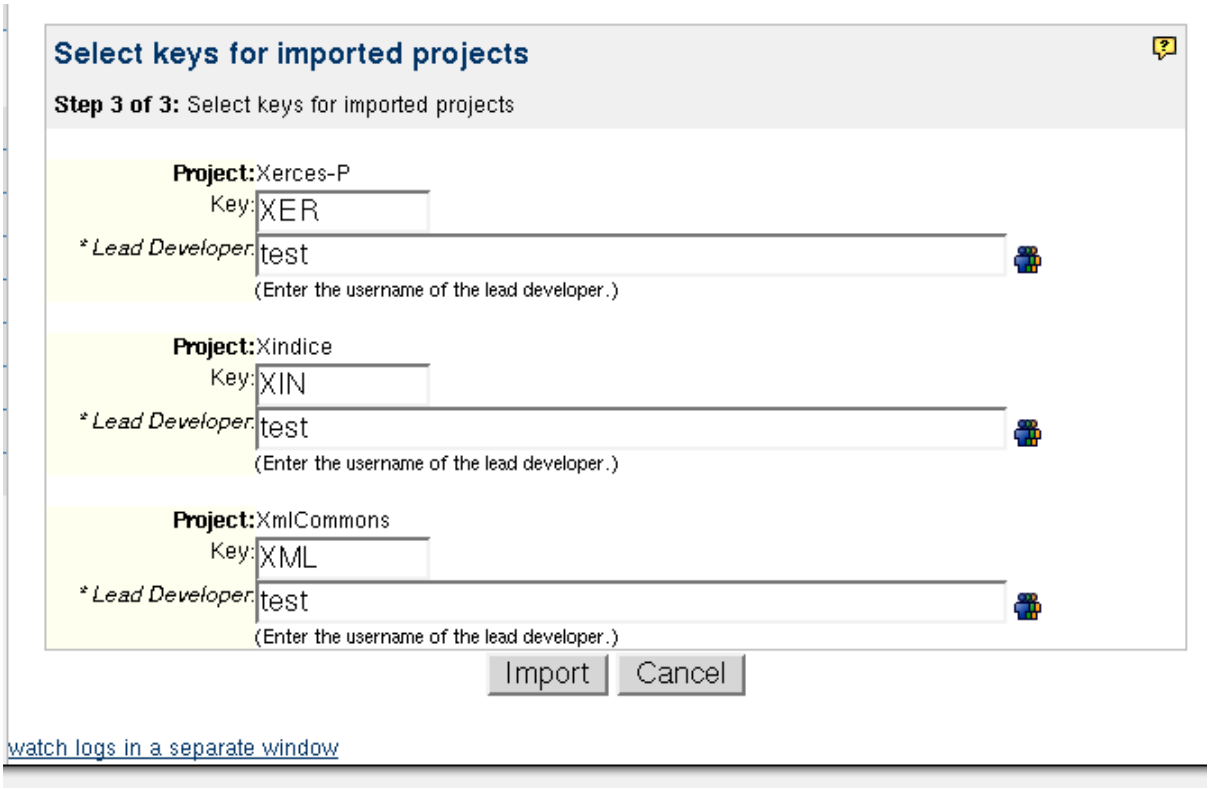
10. Click 'Next >>' to advance to the project selection page. If you have got the connection details wrong, you may have to wait about 30s for the connection attempt to time out.
11. You will now be presented with a list of projects in Bugzilla:



Choosing Bugzilla projects to import

Select the projects you wish to import (even if there is only one — select it!), and set import options via the checkboxes (the default checkbox settings is correct for most users). Click 'Next >>' to proceed.

12. You will now be prompted to choose a project key (the per-project prefix to attach to bug keys), and a project lead for each project to be imported. **Important:** do not use dashes or numbers in the project key!



Choosing JIRA project keys and leads for projects

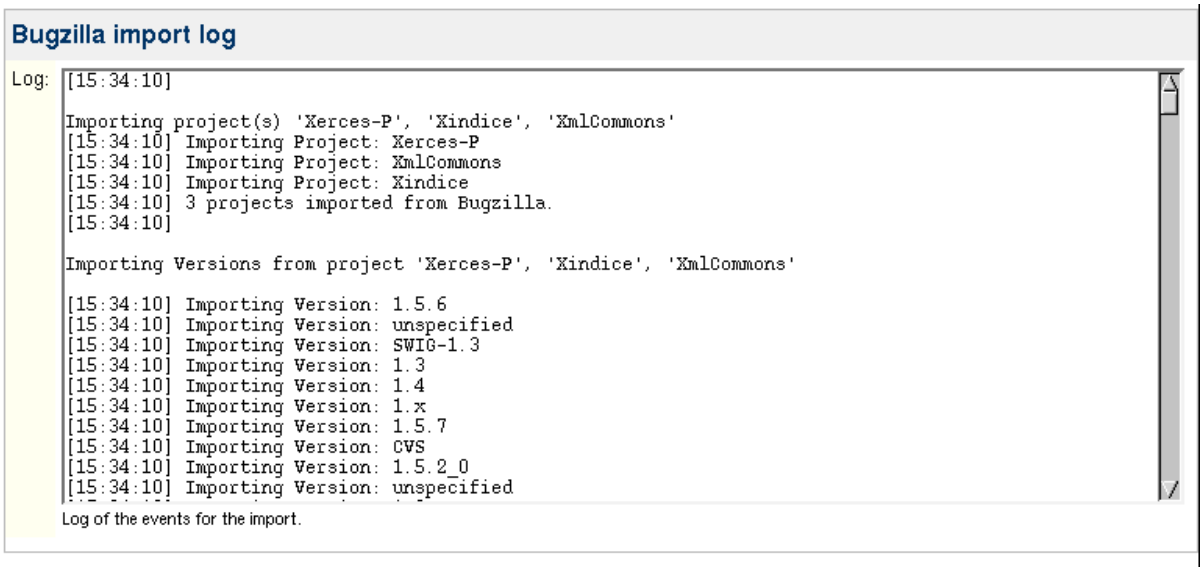
Click 'Import' when you are done to start the import. As imports frequently take a long time, you can watch the logs as they are generated by clicking the 'watch logs in separate window' link. Logs are also sent to stdout, and will appear in your app server's log:

```
Importing project(s) 'Xerces-P', 'Xindice', 'XmlCommons'
2004-02-27 15:34:10,031 INFO [atlassian.jira.util.BugzillaImportBean] Importing Project
2004-02-27 15:34:10,052 INFO [atlassian.jira.util.BugzillaImportBean] Importing Project
2004-02-27 15:34:10,061 INFO [atlassian.jira.util.BugzillaImportBean] Importing Project
2004-02-27 15:34:10,069 INFO [atlassian.jira.util.BugzillaImportBean] 3 projects import
2004-02-27 15:34:10,069 INFO [atlassian.jira.util.BugzillaImportBean]

Importing Versions from project 'Xerces-P', 'Xindice', 'XmlCommons'

2004-02-27 15:34:10,071 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version
2004-02-27 15:34:10,080 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version
2004-02-27 15:34:10,087 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version
2004-02-27 15:34:10,093 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version
2004-02-27 15:34:10,098 INFO [atlassian.jira.util.BugzillaImportBean] Importing Version
...
2004-02-27 15:34:10,228 INFO [atlassian.jira.util.BugzillaImportBean] Importing Compone
2004-02-27 15:34:10,292 INFO [atlassian.jira.util.BugzillaImportBean] Importing Compone
2004-02-27 15:34:10,346 INFO [atlassian.jira.util.BugzillaImportBean] Importing Compone
...
Importing Issues from project(s) 'Xerces-P', 'Xindice', 'XmlCommons'
2004-02-27 15:34:11,341 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue:
need Apache license"
2004-02-27 15:34:11,343 INFO [atlassian.jira.util.BugzillaImportBean] Importing User:
jason1@openinformatics.com
2004-02-27 15:34:16,430 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue:
getAttributes() in list context"
2004-02-27 15:34:16,545 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue:
must be updated"
2004-02-27 15:34:16,680 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue:
rohibits sub-classing of wrapped classes"
2004-02-27 15:34:16,878 INFO [atlassian.jira.util.BugzillaImportBean] Importing Issue:
Perl is broken (can't get attributes)"
2004-02-27 15:34:16,909 INFO [atlassian.jira.util.BugzillaImportBean] Importing User: d
....
```

```
2004-02-27 15:35:09,364 INFO [atlassian.jira.util.BugzillaImportBean] Importing User
2004-02-27 15:35:09,580 INFO [atlassian.jira.util.BugzillaImportBean] 144 issues imp
2004-02-27 15:35:09,581 INFO [atlassian.jira.util.BugzillaImportBean]

Importing Votes
2004-02-27 15:35:09,747 INFO [atlassian.jira.util.BugzillaImportBean] 0 votes import
2004-02-27 15:35:09,748 INFO [atlassian.jira.util.BugzillaImportBean] Reindexing (th
2004-02-27 15:35:11,282 WARN [jira.issue.index.AbstractDocument] Unable to index fie
with value: 0002-11-30 00:00:00.0
2004-02-27 15:35:29,170 INFO [atlassian.jira.util.BugzillaImportBean]
Import SUCCESS and took: 79142 ms.
```

It does not matter if your browser window times out — the import will continue regardless.

13. Once completed, you will see the 'Import SUCCESS' message, either in the logs (as above) or on the subsequent page:



Final Bugzilla importer screen

Congratulations, you have successfully imported your Bugzilla projects into JIRA!

If you have any questions or encounter any problems, please contact Atlassian.

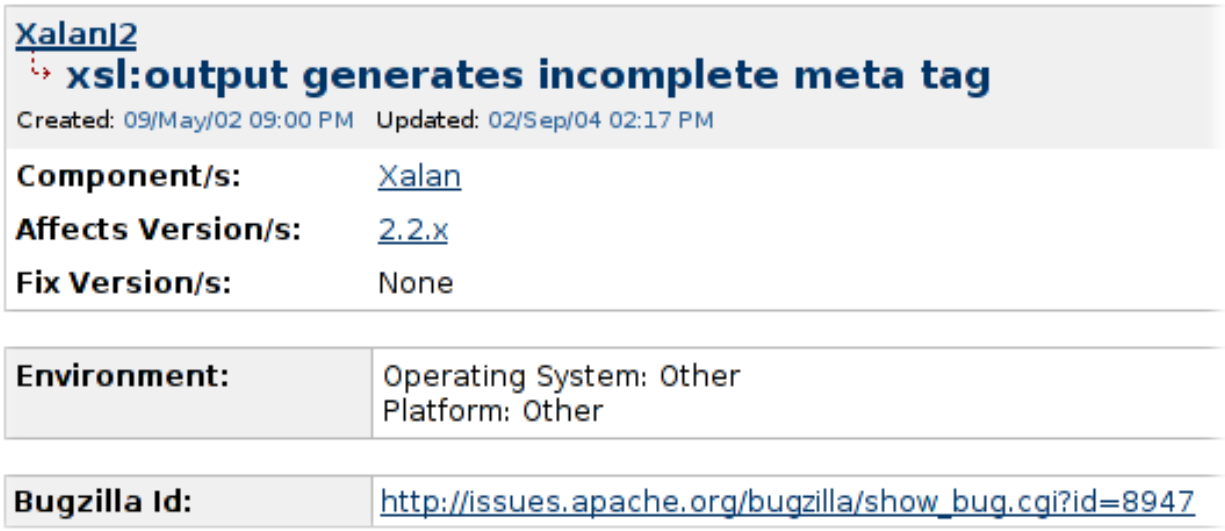
1.10.1.2. Importing only new bugs (repeated imports)

It is possible to re-import a Bugzilla project, and have JIRA import only 'new' bugs not previously imported. This allows for a transition period in which the imported JIRA project can be trialled, but bugs still logged in Bugzilla need not be lost.

To import only new bugs, click the 'Import only new issues' checkbox in the importer.

1.10.1.3. Searching by Bugzilla ID

The Bugzilla importer creates a 'Bugzilla ID' custom field for imported issues, linking back to the original Bugzilla bug URL.



Bugzilla ID custom field

If you intend to use this, you will need to configure the URL to Bugzilla in jira-application.properties. The custom field can also be made to display just the ID (unlinked) in jira-application.properties. If you don't need this custom field, delete it or hide it.

The custom field is searchable, so you can search for JIRA issues by their old Bugzilla ID. There is also a portlet which lets you search by Bugzilla ID:

Bugzilla ID Search Portlet on Dashboard

1.10.1.4. Importer Source code

The Bugzilla importer source code is available here (BugzillaImportBean.java). For some customisations you may wish to make, editing this source code is required, as described below. For users of JIRA Standalone, there is a mini build-system which you can use to quickly compile and test modifications.

1.10.1.5. Common Customisations

Importing custom priorities, statuses and resolutions

Bugzilla has a standard set of priorities, statuses and resolutions, but these can be augmented with new ones by editing a Bugzilla config file ('localconfig'). If your Bugzilla has custom statuses, JIRA will set the status of affected imported issues to "Open", and log a message ("... defaulting to JIRA status Open"). For other unknown fields (priorities, resolutions) JIRA will just not set the field, which may cause problems later (eg. issues Resolved but without a resolution are listed as open in the standard filters).

To avoid problems, it is a good idea to check whether your Bugzilla uses any custom resolutions, statuses or priorities:

```
mysql> select distinct(priority) from bugs;
+----------+
```

```
| priority |
+----------+
| P2       |
| P4       |
| P3       |
| P1       |
+----------+
4 rows in set (0.00 sec)
mysql> select distinct(resolution) from bugs;
+------------+
| resolution |
+------------+
| FIXED      |
| DUPLICATE  |
| WONTFIX    |
|            |
| LATER      |
| INVALID    |
| WORKSFORME |
| REMIND     |
+------------+
8 rows in set (0.00 sec)
mysql> select distinct(bug_status) from bugs;
+-------------+
| bug_status  |
+-------------+
| RESOLVED    |
| CLOSED      |
| NEW         |
| ASSIGNED    |
| REOPENED    |
| VERIFIED    |
| UNCONFIRMED |
+-------------+
7 rows in set (0.00 sec)
```

(the above are all standard). If your bugs use anything non-standard, you will need to edit the mappings in BugzillaImportBean.java:

```
        static
        {
            // bugzilla's priorities mapping to JIRA priorities
            priorityMap.put("blocker", "" + IssueFieldConstants.BLOCKER_PRIORITY_ID);
            priorityMap.put("critical", "" + IssueFieldConstants.CRITICAL_PRIORITY_ID);
            priorityMap.put("major", "" + IssueFieldConstants.MAJOR_PRIORITY_ID);
            priorityMap.put("normal", "" + IssueFieldConstants.MAJOR_PRIORITY_ID);
            priorityMap.put("enhancement", "" + IssueFieldConstants.MINOR_PRIORITY_ID);
            priorityMap.put("minor", "" + IssueFieldConstants.MINOR_PRIORITY_ID);
            priorityMap.put("trivial", "" + IssueFieldConstants.TRIVIAL_PRIORITY_ID);

            // bugzilla resolutions mapping to JIRA resolutions
            resolutionMap.put("", null);
            resolutionMap.put("FIXED", "" + IssueFieldConstants.FIXED_RESOLUTION_ID);
            resolutionMap.put("INVALID", "" + IssueFieldConstants.INCOMPLETE_RESOLUTION
            resolutionMap.put("WONTFIX", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID
            resolutionMap.put("LATER", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID);
            resolutionMap.put("REMIND", "" + IssueFieldConstants.WONTFIX_RESOLUTION_ID)
            resolutionMap.put("DUPLICATE", "" + IssueFieldConstants.DUPLICATE_RESOLUTIO
            resolutionMap.put("WORKSFORME", "" + IssueFieldConstants.CANNOTREPRODUCE_RE
            resolutionMap.put("NEEDTESTCASE", "" + IssueFieldConstants.INCOMPLETE_RESOL

            // bugzilla status mapping to JIRA status
            statusMap.put("UNCONFIRMED", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("NEW", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("ASSIGNED", "" + IssueFieldConstants.OPEN_STATUS_ID);
            statusMap.put("REOPENED", "" + IssueFieldConstants.REOPENED_STATUS_ID);
            statusMap.put("RESOLVED", "" + IssueFieldConstants.RESOLVED_STATUS_ID);
            statusMap.put("VERIFIED", "" + IssueFieldConstants.RESOLVED_STATUS_ID);
            statusMap.put("CLOSED", "" + IssueFieldConstants.CLOSED_STATUS_ID);
```

```
            // workflow Mappings
            wfStepMap.put("1", new Integer("1"));
            wfStepMap.put("2", new Integer("2"));
            wfStepMap.put("3", new Integer("3"));
            wfStepMap.put("4", new Integer("5"));
            wfStepMap.put("5", new Integer("4"));
            wfStepMap.put("6", new Integer("6"));

            wfStatusMap.put("1", "Open");
            wfStatusMap.put("3", "In Progress");
            wfStatusMap.put("4", "Reopened");
            wfStatusMap.put("5", "Resolved");
            wfStatusMap.put("6", "Closed");
        }
```

(Note: wfStepMap and wfStatusMap should usually not be touched, unless you are importing into project with a non-standard workflow).

Changing the imported username format

Bugzilla uses email addresses for usernames (eg. "joe@example.com"). You may wish to automatically strip everything after the '@' to form a shortened username ("joe"), or otherwise alter imported names (eg. read from a lookup table to conform to a company-wide standard). This requires editing the BugzillaImportBean source (see above). The relevant code to modify is included in BugzillaImportBean.java, but commented out:

```
    /**
     * Given a Bugzilla 'profile' user record, infer a JIRA username from it.
     * In Bugzilla your username is your email address, and this will become your JIRA use
     * is overridden to implement a different scheme.
     */
    protected String getUsernameFromBugzillaProfile(ResultSet bugzillaProfileResultSet)
            throws SQLException
    {
        return TextUtils.noNull(bugzillaProfileResultSet.getString("login_name")).toLowerC

        // Alternatively, use the first part ('joe' in 'joe@company.com')
//        String name = bugzillaProfileResultSet.getString("login_name");
//        name = TextUtils.noNull(name).trim();
//        int i = name.indexOf("@");
//        if (i != -1) name = name.substring(0, i);
//        return name;
     }
```

1.10.2. Importing Data From Mantis

JIRA is able to import data from Mantis, an open-source PHP-based bug tracker. The Mantis import is almost identical to the Bugzilla importer (just expecting a different database format), so please refer to the Bugzilla documentation for a walkthrough.

In addition to the Bugzilla importer's features, the Mantis importer also:

1. Creates a custom field, "Mantis Link", containing a link to the old Mantis bug URL. This field can be hidden when no longer useful.
2. Creates a custom field, "Mantis ID", containing the mantis bug ID (useful for searching)
3. Creates and uses a custom "Not a bug" resolution type.
4. Converts Mantis links (#1234) to JIRA links (TST-123) in text.

1.10.2.1. Importer Source code

The Mantis importer source code is available here (see also here) — fixes and improvements welcome.

Note: Before you begin, please backup your JIRA data.

1.10.3. Importing Data From FogBugz

1.10.3.1. Overview

The FogBugz importer allows you to import your bugs from an existing FogBugz installation.

Importing from a FogBugz file is a two step process. First, you will need to create a mapping file by running the FogBugz import wizard. (The mapping file is a plain text properties file that you can also manually edit. It will map your CSV fields to fields in JIRA.) Then, to perform the import, simply enter the connection details and the location of your configuration file.

1. Running the FogBugz Import Wizard
    1. Configure Project
    2. Assign field mappings
    3. Map field values
    4. Issue Links
    5. Saving the file
2. Importing your FogBugz file

Note: Before you begin, please backup your JIRA data.

Notes about the FogBugz Importer

1. **Attachments:** JIRA will import all attachments stored in the FogBugz dB. Any e-mail issues will be parsed for attachments and the e-mail text saved as a comment. The dates and user attaching the attachments will be retained.
2. **Link rewriting:** FogBugz allows for links to other issues to be automatically generated by using the format "bug issueId" or "case issue id". After import, any string matching this pattern will be rewritten to their new JIRA key. For example, a comment "Please see case 100" may be rewritten to "Please see IMP-100".

1.10.3.2. Step 1. Running the Import Configuration Wizard

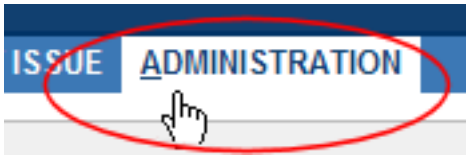
The wizard allows you to choose projects, custom fields and issue links to import. Once this is completed, you can save the file on the server and use it in the import process.

Note:

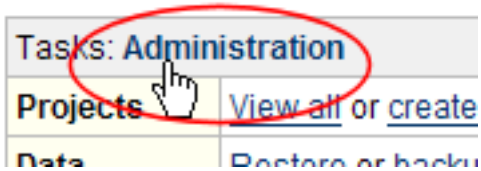
Before you start the FogBugz Import Wizard, please ensure that you have copied the JDBC driver for your database to `(your JIRA home)/common/lib/` . You will need to restart JIRA in order to allow the application to pick up the driver.

To run the FogBugz Import Wizard:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

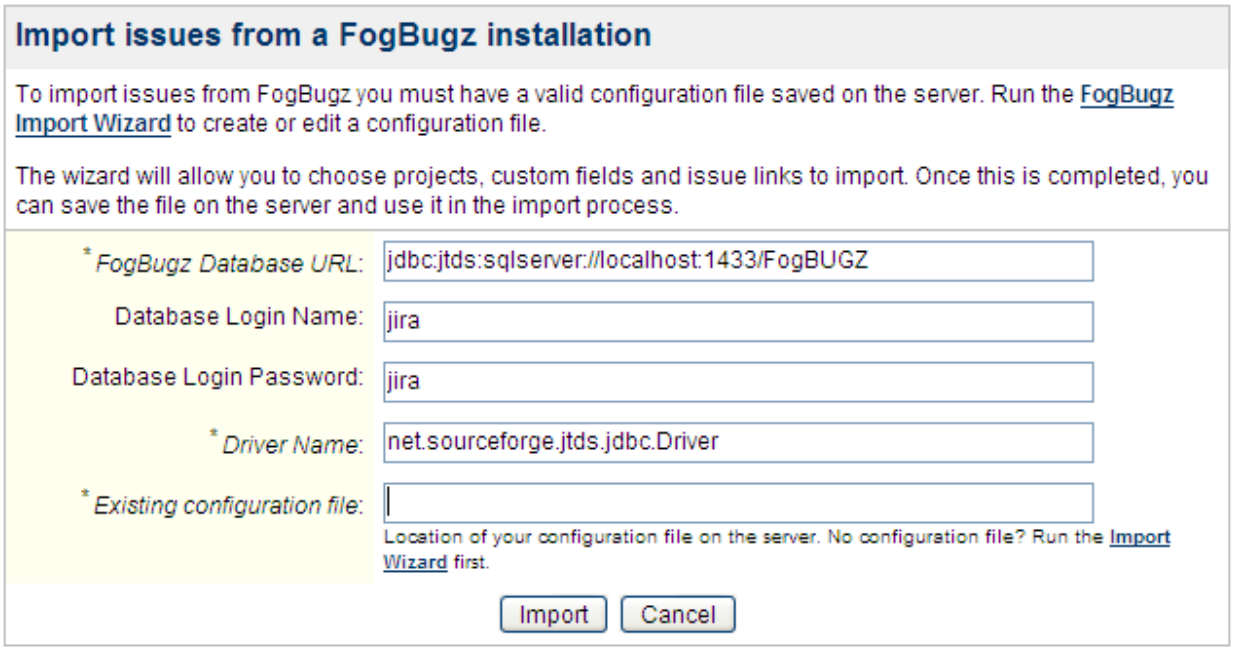


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title '**Import & Export**', click '**External System Import**.'
4. The '**Import Data**' page will be displayed. Select '**FogBugz**'.
5. The '**Import issues from a FogBugz installation**' page will be displayed:

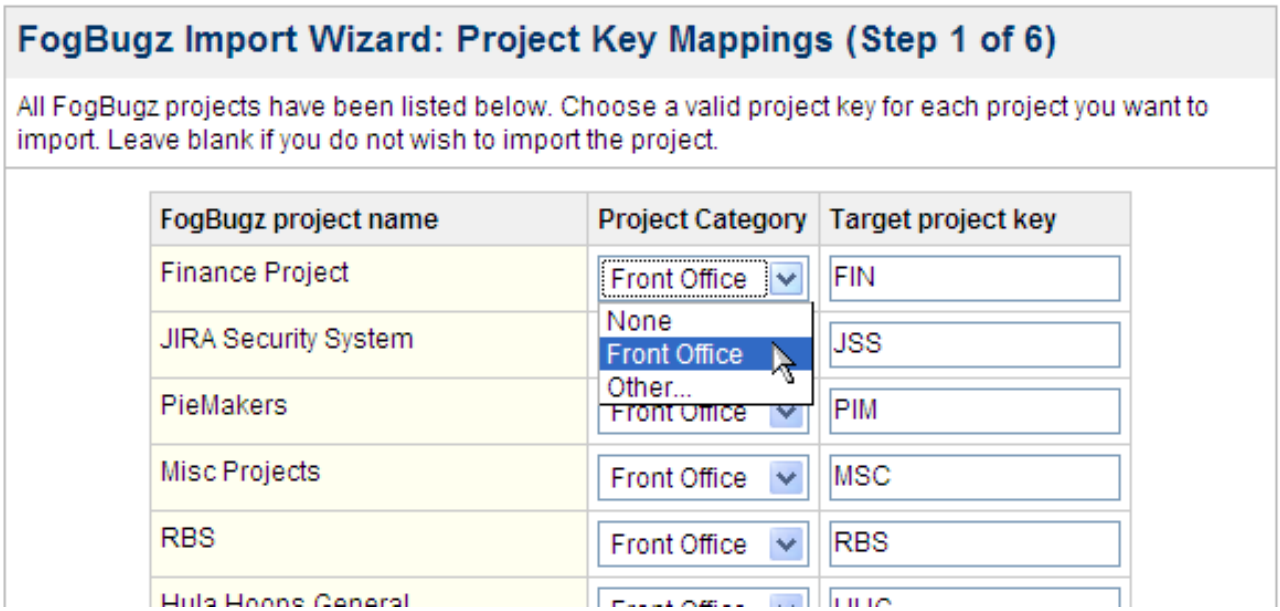


Import Configuration Wizard

6. Click the link '**Run the FogBugz Import Wizard**.

Project Configuration

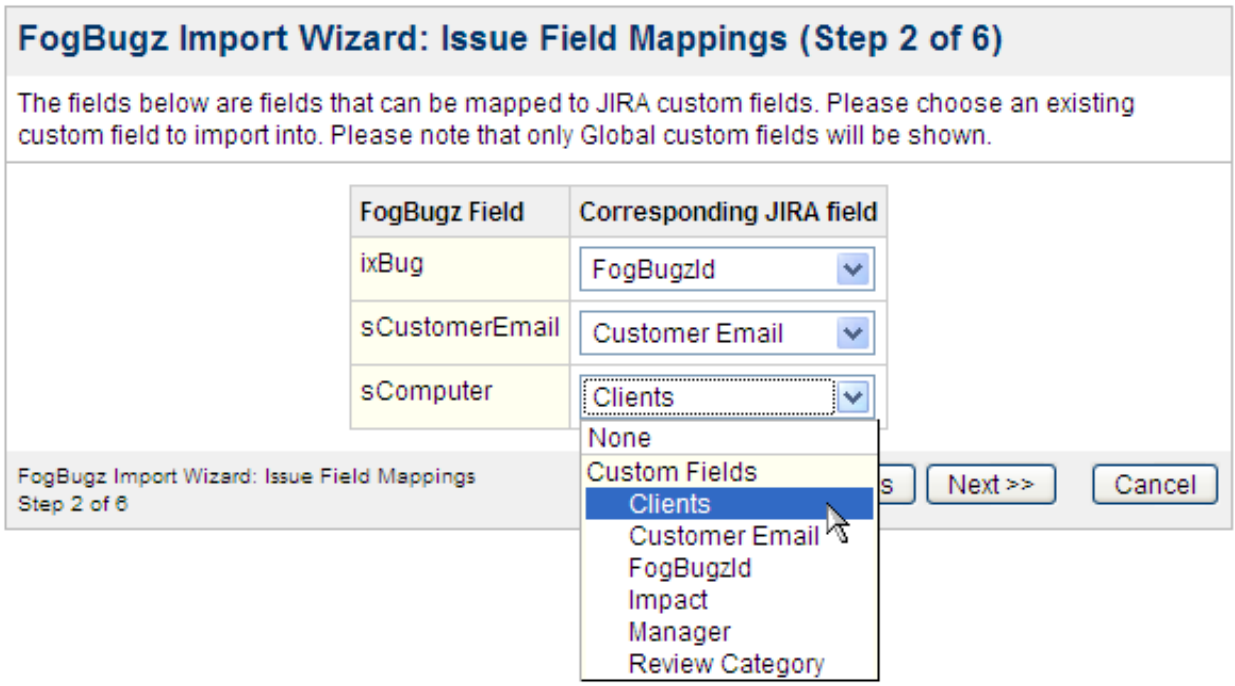
The first step is to choose which projects are to be imported. Choose a valid project key for each project you want to import, and leave blank if you do not wish to import the project. In JIRA Enterprise Edition, you can also choose a project category that the project should be created in. This will only apply if the project key does not exist and a new project is created.



Project Configuration

Issue Field Mappings

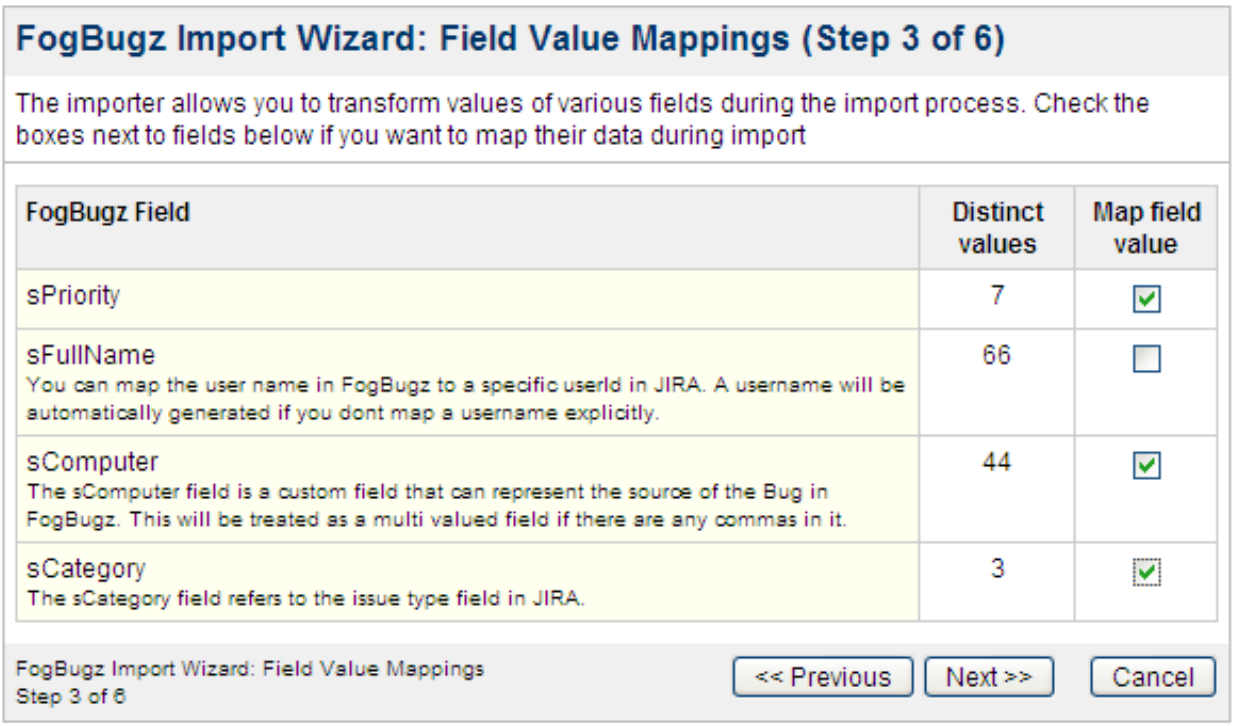
The second step is to decide which of the optional FogBugz fields you wish to import. Most of the fields in FogBugz are automatically mapped to appropriate JIRA fields. You can choose to map the fields **ixBug** (internal FogBugz issue id), **sCustomerEmail** (e-mail address of the issue if created by e-mail) and **sComputer** (custom field) to any existing global JIRA custom field.



Issue Field mappings

Value Mappings

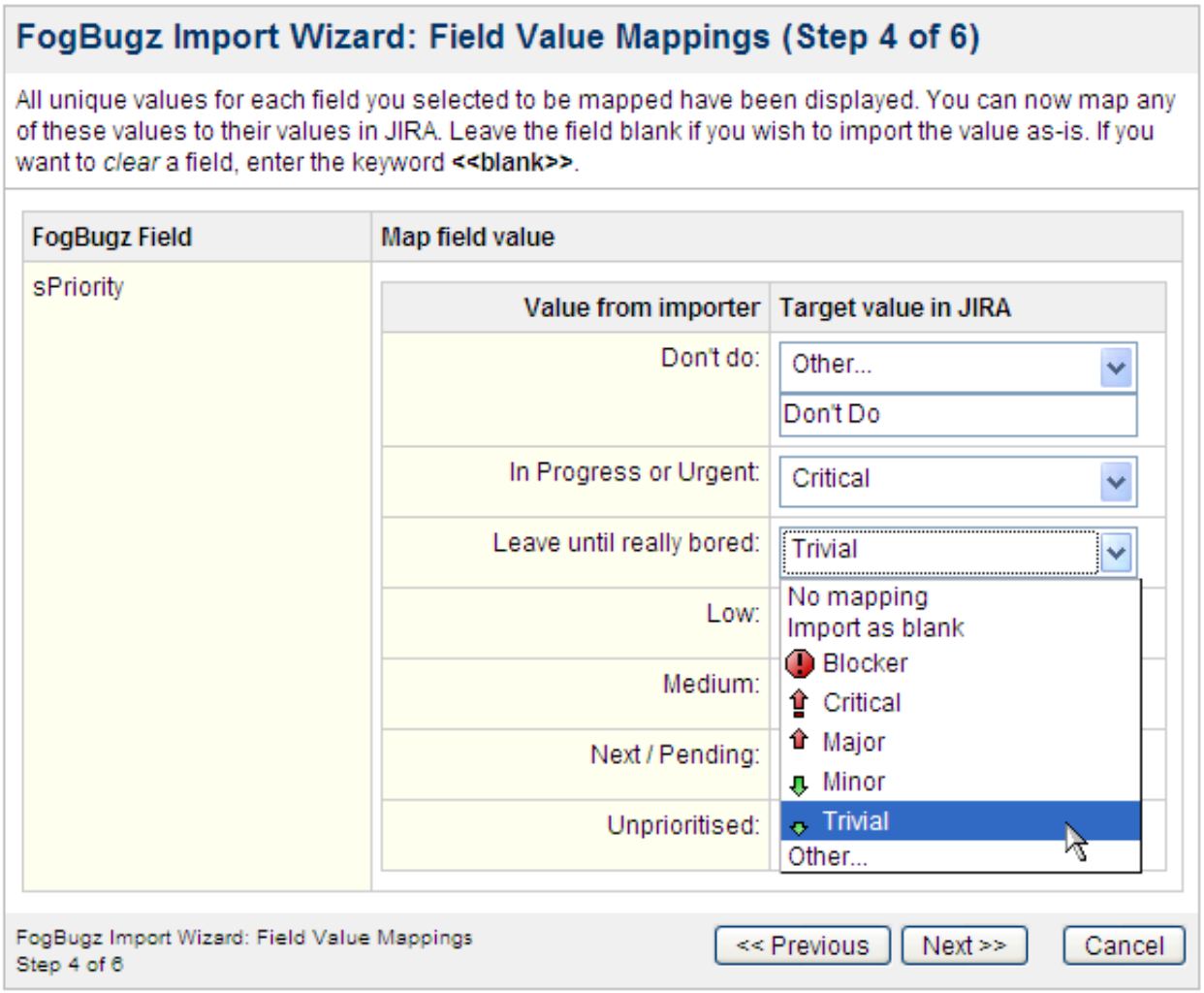
Value mappings are how values from your FogBugz importer can be 'transformed' during import. The fields **sPriority**, **sFullName**, **sComputer** and **sCategory** can have their values mapped. The first screen allows you to choose which fields you would like to map values.



Choose field for Value mappping

On the next screen, all unique values for each field you selected to be mapped have been displayed. You can now map any of these values to a new value. Leave the field blank if you wish to import the value as-is. If you want to clear a field, enter the keyword <<**blank**>>.

For **sPriority** and **sCategory** you will get a select list with the available values in JIRA. **sFullName** refers to the usernames that will be created in JIRA. If you don't specify any particular mappings, the user name will be created from the first letter of the first name and the last name, all in lowercase.



Value mappping

Note:

If you are using External User Management, the import process will not be able to create JIRA users; instead, the importer will give you a list of any new users that need to be created. You will need to create the users in your external user repository before commencing the import.

Note:

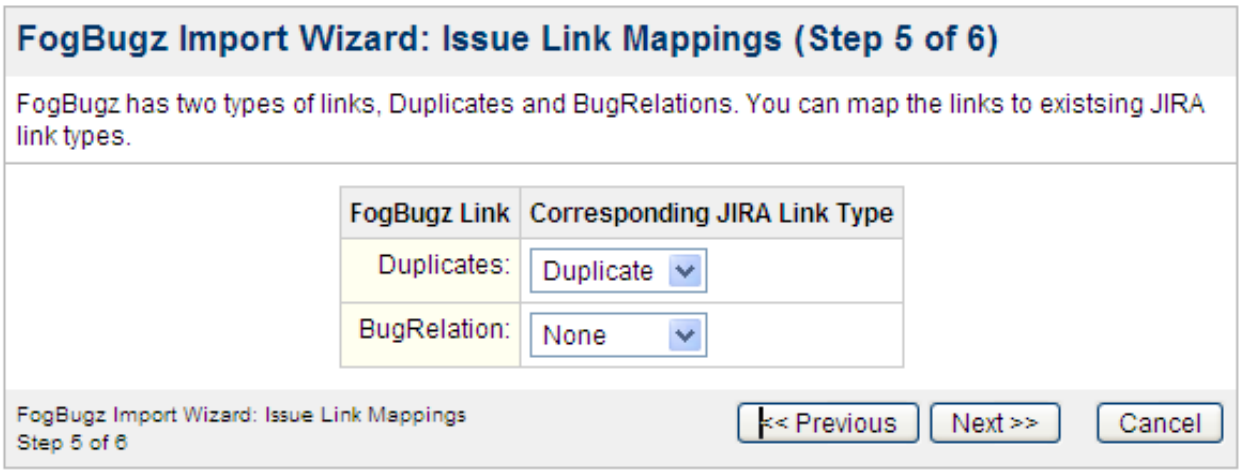
If you have a user limited license (e.g. personal license) and have reached your user limit, you can import additional users but any new users added will not be able to log in to JIRA.

Note:

If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

Issue Links

FogBugz has two types of links, **Duplicates** and **BugRelations**. On this screen, you can map the links to existing JIRA link types. Leave as **none** to not import the links.



Issue Links

Saving the configuration file

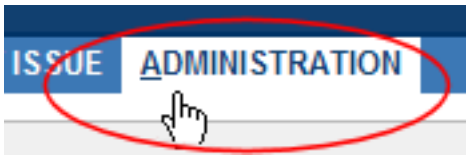
The final step of the Wizard is to save the file on your server. Please ensure you enter a valid path. You can also choose to continue on with the import without saving the file.

You can also see a preview of the mapping file that will be saved.

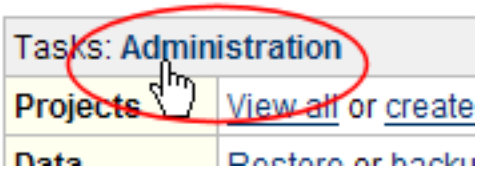
1.10.3.3. Step 2. Importing the FogBugz file

Once you have your configuration file, you can now import the FogBugz file into JIRA.

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

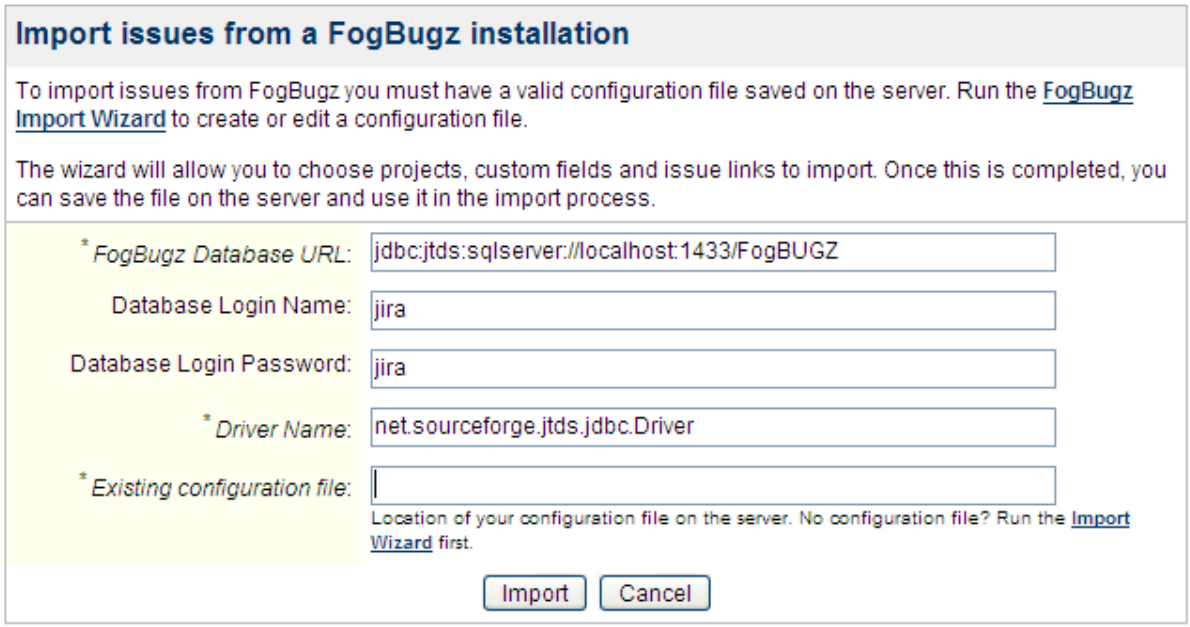


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title '**Import & Export**', click '**External System Import**.'
4. The '**Import Data**' page will be displayed. Select '**FogBugz**'.
5. The '**Import issues from a FogBugz installation**' page will be displayed:



Import Configuration Wizard

6. Type the location of your FogBugz file and your configuration file, and click the '**Import**' button.

The '**Settings**' page gives you precise control over what will be imported on each import run.

Once the import has begun you'll be able to follow live progress of the import, with the screen refreshing at around every 10 seconds. You can change this value by updating the field at the bottom of the page. The importer also give you stats about what objects have been imported and time elapsed so you can have an idea as approximately how long the import will take. You can also choose to Abort the import, which will cease importing after the current issue is done.

1.10.4. Importing Data From CSV

1.10.4.1. Overview

The CSV importer provides a powerful and flexible way to import data from a comma-separated file, which is a format supported by most applications (e.g. Microsoft Excel).

Note:

Please note that there a number of import methods available for importing data into JIRA from other issue tracking systems. It may be more appropriate to use a method other than the CSV importer, depending on what system you are importing data from. Details on other methods of importing data are available here.

Importing from a CSV file is a three step process. First, you need to prepare and verify your CSV file. Next, create a mapping file by running the CSV import wizard. (The mapping file is a plain text properties file that you can also manually edit. It will map your CSV fields to fields in JIRA.) Finally, to perform the import, simply enter the location of your import file and your configuration file.

1. Preparing your CSV file
2. Running the CSV Import Wizard
    1. Configure Project
    2. Assign field mappings
    3. Map field values
    4. Miscellaneous information
    5. Saving the file

3. Importing your CSV file

Note: Before you begin, please backup your JIRA data.

1.10.4.2. 1. Preparing your CSV file

The first thing you need to do is to ensure that your CSV is a valid CSV format. A good way to check is to import your file into a spreadsheet (e.g. Microsoft Excel, Open Office) and see if the data is as expected. This is also a good opportunity to do any massaging of the data, if you wish.

If you have values that signify a blank value (e.g. <blank> or None), it's best if you simply remove them in this step.

For *built-in JIRA fields* (e.g. **Fix-for version**, **Affects version**, **Component**), if you wish to set more then one value for an issue, you will need to have a value per column in your CSV, with each column given a distinct name. For example:

```
IssueType,Summary,FixVersion_1,FixVersion_2
bug,"First issue",v1,
bug,"Second issue",v2,
bug,"third issue",v1,v2
```

In this example, the third imported issue will have its Fix-for version set to multiple values.

For *custom fields* the situation is different, and multiple values are comma-separated. See below for details.

Valid file format

The CSV importer assumes a Microsoft Excel styled CSV file. Fields are separated by commas, and enclosed in quotes if they contain commas or quotes. Embedded quotes are doubled.

There are two requirements of the CSV, in addition to being well-formed in general:

- The CSV file **must contain a heading row**. The CSV configuration wizard uses the CSV header row extensively. The header values should **not have any punctuation** (beyond the commas separating records) such as apostrophes or the importer may not work correctly.
- As a minimum, the CSV file must contain a column for **Summary** data.

You can also have multi-lined CSV. For example, here is a valid file with a single record:

```
Summary,Description,Status
Login fails,"This is on
a new line",Open
```

CSV file encoding

JIRA will read the CSV file using the **system encoding**, which can be seen in Administration -> System Info. Make sure that you either save the CSV file with this encoding, or set -Dfile.encoding on startup to force the system encoding to be what you're using (utf8 is best).

Importing Comments from CSV

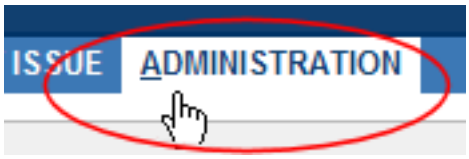
If a row contains more columns than there are header columns, then the excess columns will be added as comments (though see JRA-7672.

1.10.4.3. 2. Running the CSV Import Configuration Wizard

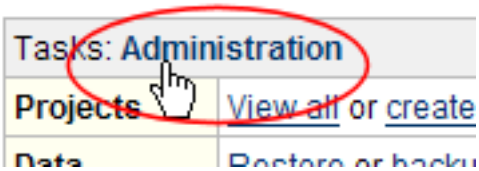
The next step of the import process is to run the import configuration wizard to determine how the

CSV data can be mapped to JIRA fields.

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

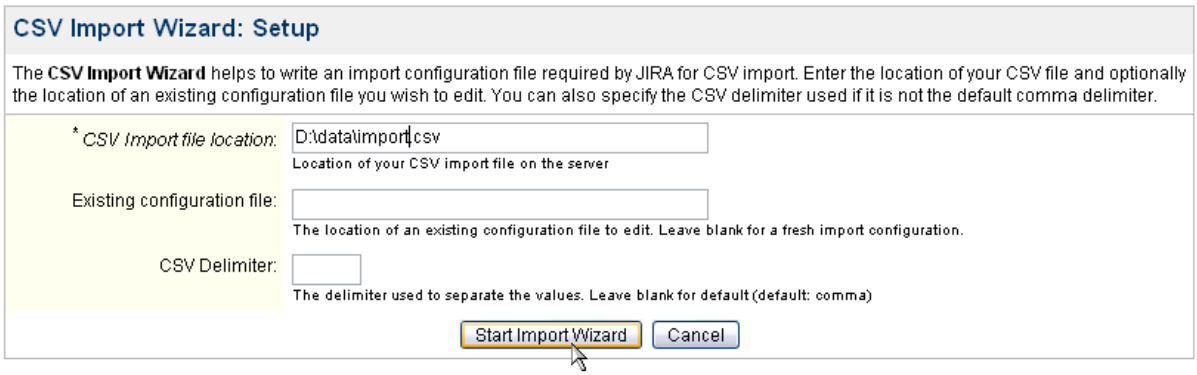


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title '**Import & Export**', click '**External System Import**.'
4. The '**Import Data**' page will be displayed. Select '**Comma-separated values (CSV)**'.
5. The '**Import issues from CSV file**' page will be displayed. Click the '**CSV Import Wizard**' link.
6. The '**CSV Import Wizard: Setup**' page will be displayed:



Import Configuration Wizard

You can optionally specify the delimiter your CSV file used. Leave the field blank if you wish to use the (default) comma delimiter. Please note that the delimiter can only be one character long.

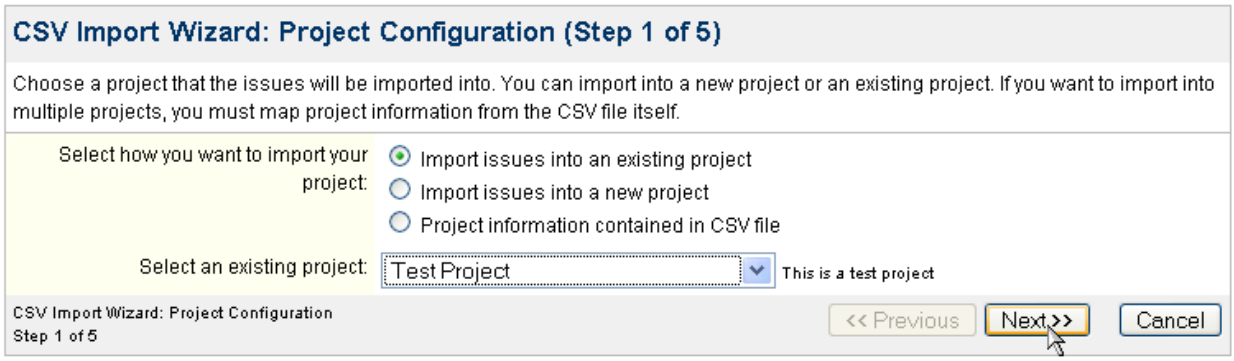
7. Click the link '**Start Import Wizard**.

2.1 Project Configuration

The first step is to choose which project the issues will be imported into. You can import into a new project or an existing project. If certain project details (e.g. **name** and **key**) match an existing project, then the issues will be imported into an existing project. Note that if you are creating a new project, no validations are performed at this time — invalid data will result in a failure later, at import time.

If you want to import into multiple projects, you must map project information from the CSV file itself. That means that all rows must have the project information in them.

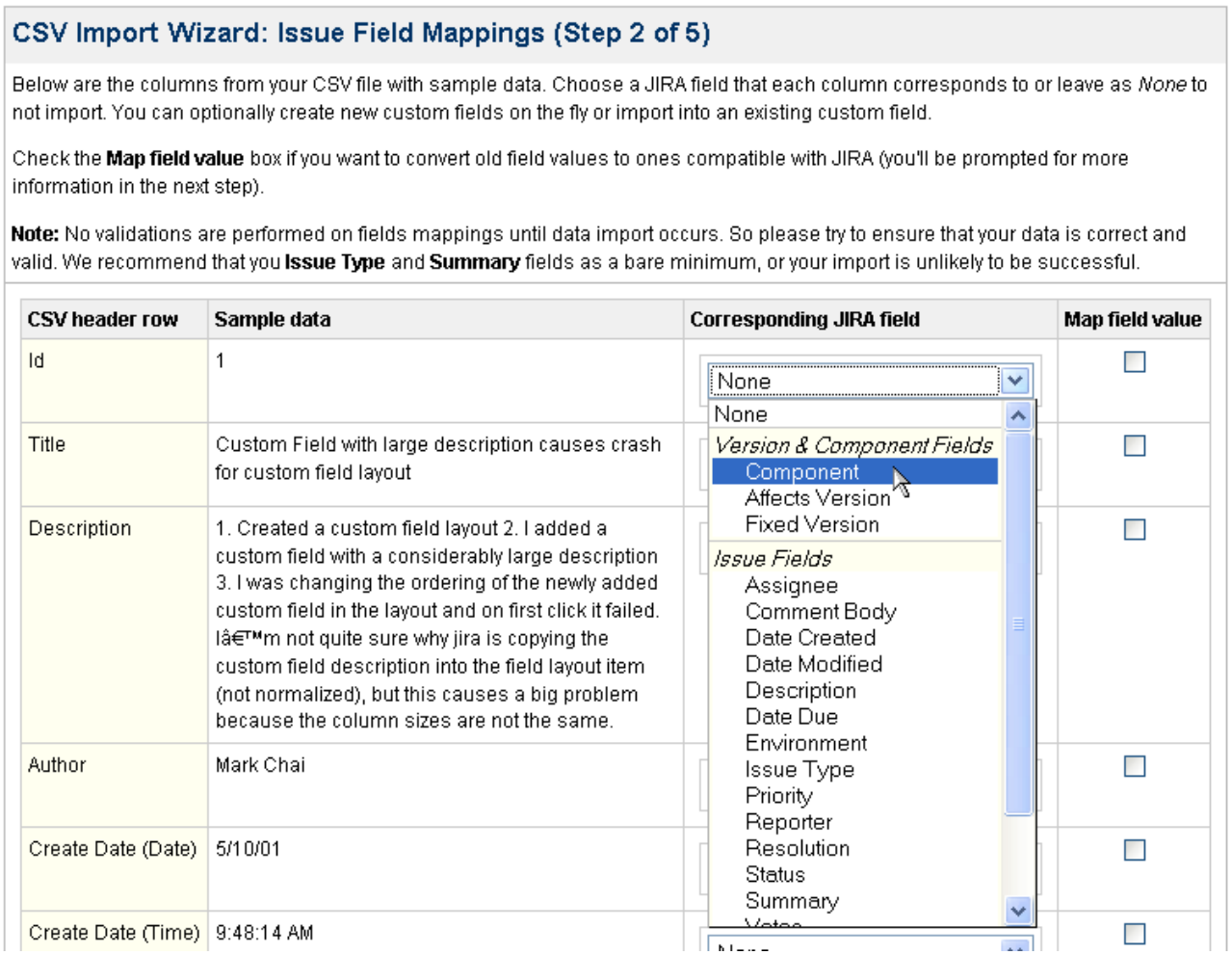
The recommended import method is a *single project* per CSV file, imported into an *existing project*.



Project Configuration

2.2 Issue Field Mappings

The second step is to decide which CSV fields you want to import. The screen shows all the columns that are found in your CSV file, and a sample data row. On this screen you can map each column of your CSV file to system fields in JIRA, or leave as **None** to not import. You can optionally create new custom fields on the fly or import into an existing custom field.



Issue Field mappings

Note:
As no validations are performed on field mappings until data import occurs, please try to ensure that your data is correct and valid.

System Fields

You can select multiple fields to be combined into Version and Component fields. For example, you can import from '**Raised Version**' and '**Found in Version**' into the '**Affects Versions**' field. For Versions and Components, the importer will detect if the version exists in JIRA for the project . If it doesn't exist, then it will automatically created.

User fields (**Assignee** and **Reporter**) are assumed to be in a 'FirstName LastName' format. New users will be created with the username as 'FirstNameLastName'; spaces, apostrophes and brackets are stripped out. If the name only has one word, that one word will be used as the username.

Note:

If you are using External User Management, the import process will not be able to create JIRA users; instead, the importer will give you a list of any new users that need to be created. You will need to create the users in your external user repository before commencing the import.

Note:

If you have a user limited license (e.g. personal license), any users you import over and above your user limit will be created in JIRA without permission to log in to JIRA. You will not be able to select which of your users are assigned login permissions under the user limit, when you perform the import. However, you can change this after the users are imported, by editing user permissions.

In most cases when importing system fields values like **Priority**, **>Issue Type**, **Status** and **Resolution**, you will need to map the field values. The mapping needs to be done even if the imported CSV file has the values set to 'valid' names, e.g. **Issue Type** set to 'Bug' or 'New Feature'. The only alternative to mapping the values is to change the CSV file such that it contains the exact IDs of JIRA's **priorities**, **issue types**, **statuses** and **resolutions** instead of their names. This requires you to determine the correct IDs and then update the whole CSV file, so it is easier to map the values during the import.

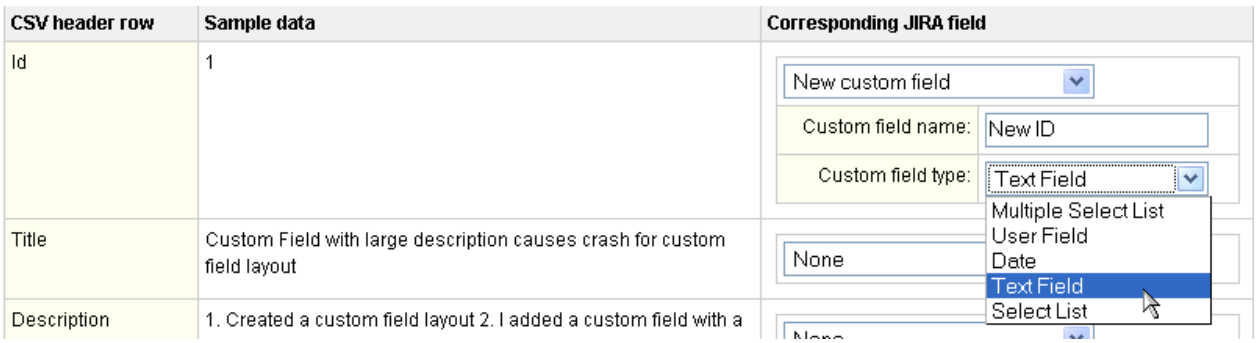
Time Tracking Fields

As of JIRA 3.7 you can now import into the 'Original Estimate', 'Remaining Estimate', and 'Time Spent' fields. For these fields to be available you must have enabled time tracking within JIRA. The importer expects the estimate or time spent to be expressed in seconds. A value that is not able to be converted to a numeric value will be ignored and not imported.

The portion of the importer that converts the raw string to a java.lang.Long which represents number of seconds is customizable. If you are trying to import data that needs to more intelligently process the value (more than just converting the string to a numeric value) you can write your own java class. It needs to implement the com.atlassian.jira.imports.csv.mapper.TimeEstimateConverter interface and you can direct the importer to use your class by specifying in the csv.properties configuration file the 'settings.advanced.mapper.time.estimate.converter' property (i.e. settings.advanced.mapper.time.estimate.converter=com.atlassian.jira.imports.csv.mappers.SimpleTimeEstim com.atlassian.jira.imports.csv.mappers.SimpleTimeEstimateConverter)

Custom Fields and the importer

You can also map a column to an existing custom field or create a new custom field on the fly. Currently you can only create certain custom fields on the fly. All custom fields created this way will be globally scoped. Moreover, if the name matches an existing custom field, that existing custom field will be used instead. If you are worried about how this works exactly, we recommend that you create your custom fields first before importing them.



Creating custom fields on the fly

If you map to a select list custom field, all unique values will be created as options at import time. If you map to a multiple select field, its values should be separated by a comma. If the field values have commas in them, the commas should be escaped with a backslash. Thus the field:

```
"Wally\, I,Wally\, II"
```

would be translated into one field with multiple values:

- Wally I
- Wally II

Once again, no data validations are done at configuration time, so you should ensure that the data you are trying to import is of a compatible type.

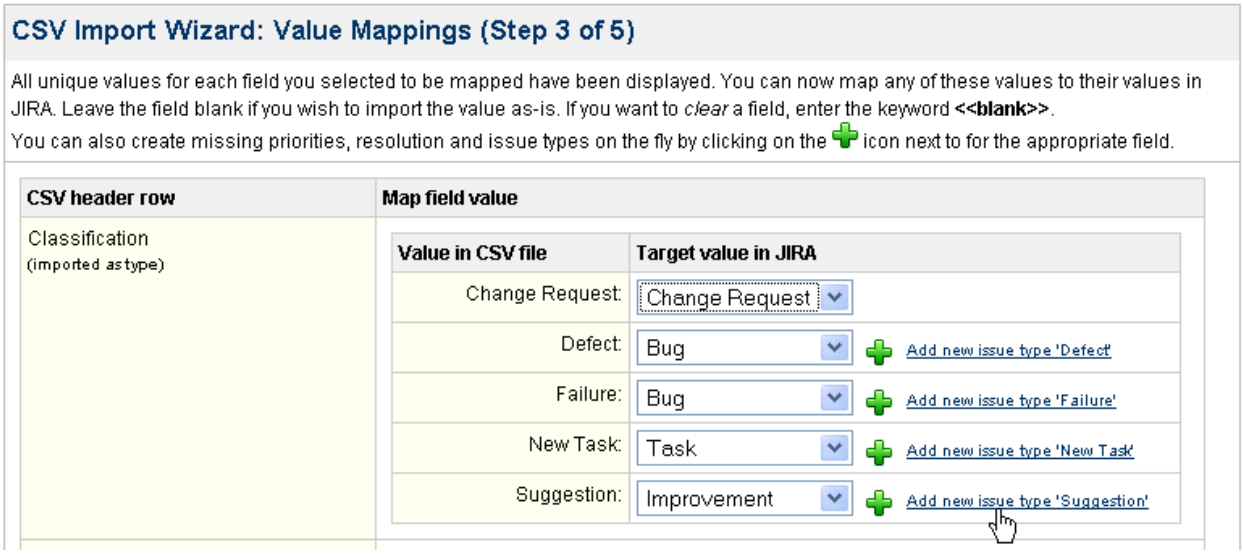
2.3 Map Field Values

You may wish to map certain values in your CSV file to a different value. For example, you might map the field '**Severity**' to JIRA's '**Priority**' field. JIRA expects the ID of priorities that exist in JIRA. Thus for this field, you'll need to check the **Map field value** check-box. This will affect the next screen that you will come to.

Value Mappings

Value mappings determine how values from your CSV importer will be 'translated' to match the values expected by JIRA. This is usually required for fields such as '**Issue Type**', '**Resolution**', '**Priority**' and '**Status**', but can also apply to other fields. On this screen, all unique values for each field you selected to be mapped have been displayed. You can now map any of these values to their values in JIRA. Leave the field blank if you wish to import the value as-is. If you want to clear a field, enter the keyword <<**blank**>>.

On the '**Field Mappings**' screen, each field has a checkbox under the heading '**Map values**'. If you check these boxes you will be able to map the values of these fields when you progress to the next page.



Value mappping

For fields mapping to **Resolution**, **Priority** and **Issue Type**, you will get a select list with the available values in JIRA. In addition you can quickly create values that do not exist in JIRA by clicking the green plus symbols. These will become issue constants that you can change at a later time.

For fields mapping to **Status**, you will get the select list with JIRA's available values, but no plus symbol for creating new status values.

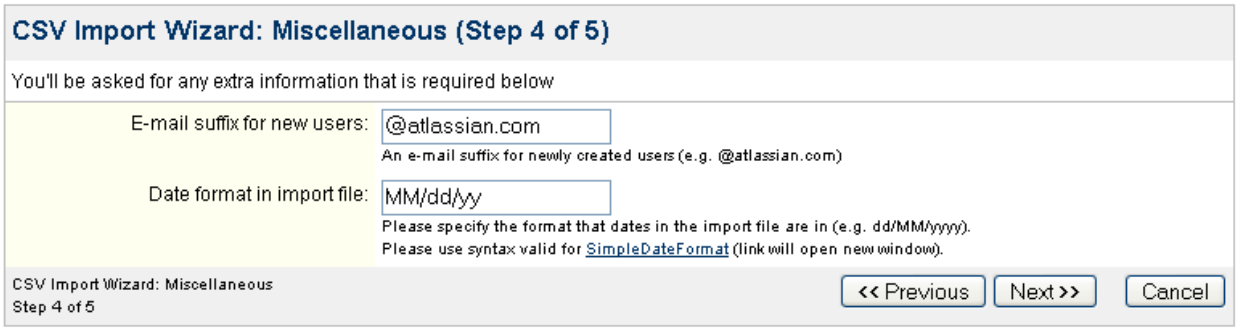
For these fields, there are two special options in the select list in addition to JIRA's available values, '**Import as blank**' and '**No mapping**'. '**Import as blank**' causes the JIRA value to be blank for that field. '**No mapping**' attempts to import the value in the CSV file as-is. Note that using '**No mapping**' for a field value will result in a failed import if the value is not valid for that JIRA field.

For fields mapping to **Status** and **Issue Type**, default values are used when the **blank** option is selected.

2.4 Miscellaneous Information

You will be asked to enter some extra information on this screen, such as:

1. The domain name of the users that will be created in the system.
2. If you are importing date fields, you will also be asked to supply the date format that is used in your CSV file. Note that this could be different from the date format that is used in JIRA. All date fields will be interpreted using the format you supply.

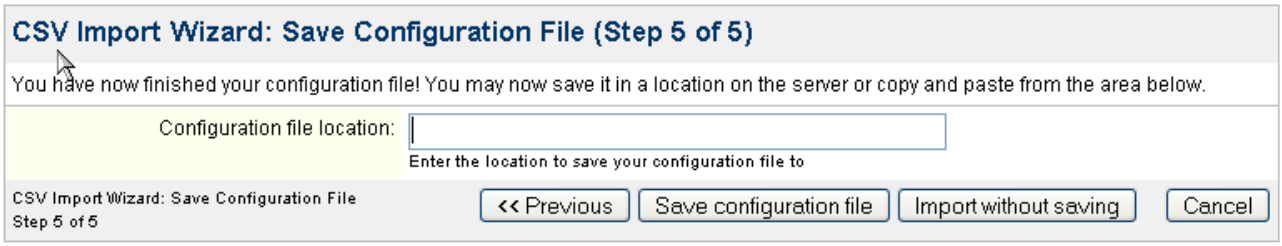


Miscellaneous information

2.5 Saving the Configuration File

The final step of the Wizard allows you to save the configuration file on your server. Saving the configuration file enables you to import more CSV files later without going through the Configuration Wizard again. Please ensure you enter a valid path. Alternatively, you can choose to continue on with the import without saving the configuration in a file.

You can also see a preview of the mapping file that will be saved.

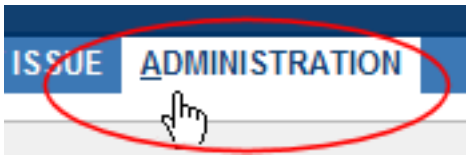


Saving the file

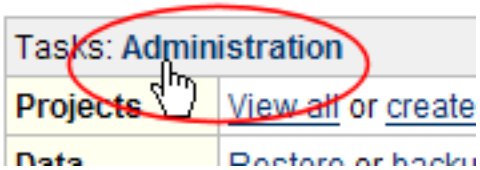
1.10.4.4. 3. Importing the CSV file

Once you have your configuration file, you can then import the CSV file into JIRA.

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

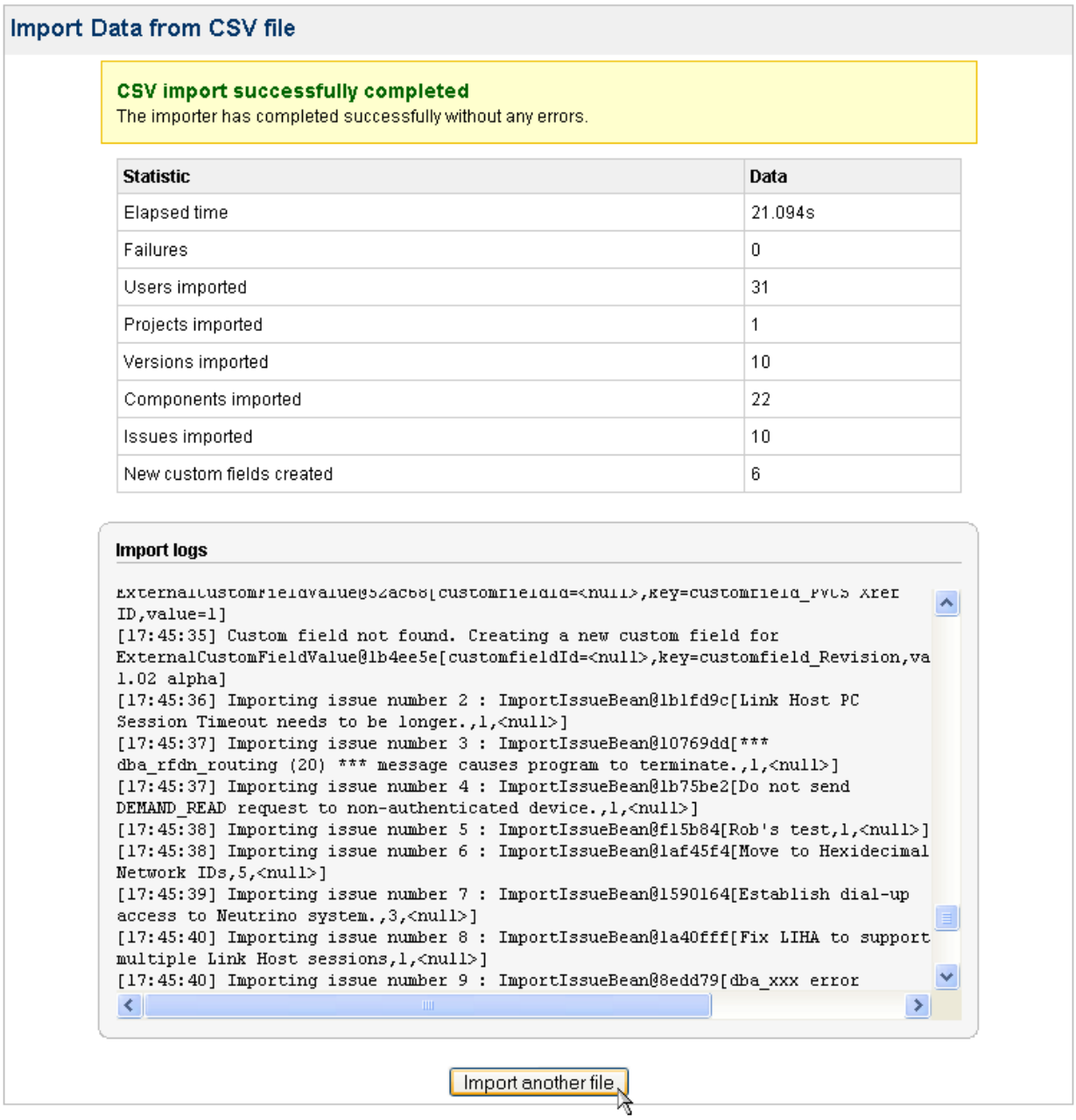


Link to Administration section

3. On the panel on the left, under the title '**Import & Export**', click '**External System Import**.'
4. The '**Import Data**' page will be displayed. Select '**Comma-separated values (CSV)**'.
5. The '**Import issues from CSV file**' page will be displayed.
6. Type the location of your CSV file and your configuration file, and click the '**Import**' button.

The '**Settings**' page gives you precise control over what will be imported on each import run.

Once the import has begun you will be able to follow the progress of the import, with the screen refreshing around every 10 seconds. You can change this rate by updating the field at the bottom of the page. The importer also give you statistics about what objects have been imported and time elapsed so you can have an idea as approximately how long the import will take. You can also choose to '**Abort**' the import, which will cease importing after the current issue is done.



Import completed

1.10.4.5. Known Issues

- The CSV Importer doesn't import all the objects available in JIRA at present. You can find these limitations at issue JRA-5774.
- There is a known problem that prevents the CSV Importer from being used with JIRA instances running on JBoss 4.x. This is due to a compatibility issue between the JBoss 4.x commons-collections.jar and the JIRA commons-collections.jar. The workaround is to replace the commons-collections.jar in JBoss 4.x with the more recent JIRA version. Please see JRA-6473 for further details.

1.11. Moving or Archiving Individual Projects

1.11.1. Moving or Archiving Individual Projects

Over time, your organisation's requirements may change. This can lead to needing to:

1. Archive a completed or obsolete project.
2. Split a large JIRA instance into several JIRA instances, with particular projects in each.
3. Restore a single project from a backup file into a JIRA instance.
4. Restore an entire JIRA instance, from a backup into a new empty JIRA instance.

1.11.2. Archiving a Project

It is sometimes necessary to archive an old project, while retaining the project's data for future auditing purposes. There are a number of ways to achieve this:

1.11.2.1. Online archiving

Archiving a project online means keeping all of the project's issue data in your live JIRA instance. The advantage of archiving a project online is that you can easily make the project accessible again if required.

There are two ways to archive a project online:

'Hiding' a project

A 'hidden' project will still be visible via the 'Administration' menu, but it will no longer appear in the 'Browse Projects' list, and noone will be able to search, view or modify any of the project's issues.

1. Create a new permission scheme. Leave all of the permissions empty.
2. Associate the new permission scheme with the project that you wish to hide (see Assigning a Permission Scheme to a Project).

Making a project 'Read-Only'

If you make a project read-only, the project will be visible via the 'Administration' menu, and will appear in the 'Browse Projects' list. The project's issues will be searchable and viewable, but noone will be able to modify them.

1. Create a new permission scheme. Grant the '**Browse Project**' to everyone who needs to be able to search or browse the project, or view its issues. Leave all of the other permissions empty.
2. Associate the new permission scheme with the project that you wish to hide (see Assigning a Permission Scheme to a Project).

Accessing an archived online project

If you archived a project online, by hiding it or making it read-only, then all of the project's data can be made accessible very easily. Simply associate the project with a permission scheme where the appropriate permissions (e.g. '**Edit Issue**', '**Assign Issue**', '**Resolve Issue**', etc) are assigned to the appropriate people.

1.11.2.2. Offline archiving

Archiving a project offline means creating an XML backup, then deleting the project and all of its issue data from your live JIRA instance. The project will no longer be available via the 'Administration' menu or the 'Browse Projects' list, and its issues will no longer exist in your live JIRA system.

The disadvantage of offline archiving is that there is no easy way to restore a deleted project to your live JIRA instance.

Note:

If there is a possibility that you will need to restore the project into your live JIRA instance at some point in the future, then online

archiving is recommended. Offline archiving should only be done if you are certain you will never need to restore this project to a live JIRA instance (i.e. you will only ever restore the data to a non-production instance).

Archiving a project offline

1. Create a global XML backup of your entire live JIRA instance.
2. Import the XML backup into a test JIRA instance. **Make sure that the test JIRA instance uses a separate database from your live JIRA instance, as the import will overwrite all data in the database.**
3. In your test JIRA instance, verify that you can view the issues of the project that you are archiving.
4. In your live JIRA instance, select **Projects** from the **Administration** menu, then click the **Delete** link to delete the project and all of its issues.

Accessing an archived offline project

1. Import the XML backup into a test JIRA instance. **Make sure that the test JIRA instance uses a separate database from your live JIRA instance, as the import will overwrite all data in the database.**

Restoring a deleted project

If you wish to restore a project from a backup file, please refer to the instructions in the Restoring a Project from Backup documentation.

1.11.3. Splitting a JIRA instance

Occasionally an organisation may need to split its existing JIRA instance into two separate instances. For example, there might be a requirement to have some particular projects in one JIRA instance, and other projects in a second instance.

Note:

This requires two separate server licenses.

To split a JIRA instance,

1. Backup your database, using your database backup procedures, and verify the backup.
2. Backup your attachments directory and verify the backup.
3. Install JIRA on your new server. NOTE:
   - The JIRA version number on your new server must be the same as (or higher than) the version number on your existing server.
   - The JIRA edition on your new server must be the same as (or higher than) the edition on your existing server. E.g. If your existing server is running JIRA Professional edition, the new server must be running either JIRA Professional or JIRA Enterprise edition (not JIRA Standard edition).
   - Do not connect two JIRA instances to the same external database instance.
4. Create an XML file from your existing JIRA server, as described in Backing up data.
5. Import the XML file into your new server, as described in Restoring data.
6. Copy the attachments directory from your existing server to your new server, and configure your new server to use its own directory (for details please see Enabling File Attachments).
7. At this point you should have two JIRA instances with the same users, projects, issues and attachments. Login to both instances and perform some random searches to verify that the data is identical in both instances.
8. Delete the non-required projects from each JIRA instance.

1.12. Integrating with Source Control

1.12.1. Integrating JIRA with CVS and ViewCVS

JIRA's CVS integration shows the related CVS commit information for an issue. When a CVS commit message mentions an issue, JIRA picks this up and displays the commit log in a tab in the mentioned issue.

JIRA's CVS integration features include:

- Ability to interact with a CVS server log directly via local access, pserver or external (ssh) protocols, or to parse a CVS log file generated by an external process.
- Access to the version control information in JIRA can be easily controlled using flexible permissions. If you are running a public instance of JIRA, and do not want the rest of the world to see the version control information, JIRA can be configured to restrict access to that information to the chosen users.
- ViewCVS or FishEye are supported out-of-the-box; and Subversion is available as a plugin (drop-in extensions to JIRA).
- If CVS integration is configured, the files and revisions in JIRA are linked to the relevant pages. E.g. the following screenshot shows a JIRA project:

The Version Control tab

  Because ViewCVS is configured, JIRA has turned the displayed commit information into ViewCVS links. Clicking the name of the file will take the user to the ViewCVS file summary page. Clicking the revision will take the user to the page that shows the contents of the file as it was at that revision. Clicking the 'diff' summary will show the ViewCVS 'diff' page between the shown revision of the file and its previous revision.
- Each project in JIRA can be associated with a CVS module. In JIRA Standard, a project can be associated with only one module. In JIRA Professional and Enterprise editions, a project can have multiple modules.

There are 3 steps to configure CVS integration in JIRA:

1. Create a CVS module
2. Associate project(s) with CVS module(s)
3. Grant permission to view CVS information

1.12.1.1. How JIRA's CVS integration works

JIRA retrieves the CVS commit information for an issue by parsing the output of the 'cvs rlog' (or cvs log) command of each associated CVS module and scanning it for the issue's key. If an issue key is found in the commit message, the commit message is displayed on the Version Control tab for the issue.

If you have allowed JIRA to automatically synchronise with the CVS repository, JIRA will periodically run the 'cvs rlog' command for the module and store the results in a file which path is specified by the module's *Log File Path* attribute. The file is then parsed for commit information.

> **Note:**
> Even if you are using local repository access JIRA will obtain the CVS log for the module and then parse it. JIRA does not access the CVS repository directly.

If you have chosen to update the log manually, JIRA will only periodically parse the CVS log specified by the module's *Log File Path* attribute.

As JIRA parses the module's CVS log and keeps relevant commits in memory, the required memory for JIRA is relative to the size of the CVS module.

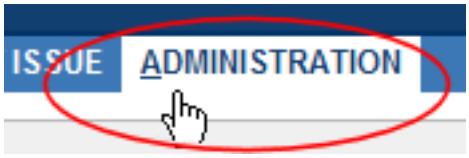
Please note:

- Currently, JIRA is able to retrieve CVS log data via local access, pserver protocol or ssh (ext method). If your CVS is not reachable by these methods you can disable automatic log retrieval (see below).
- If you would like JIRA to automatically keep synchronised with your CVS repository, the communication between JIRA and the CVS server might be fairly bandwidth intensive as JIRA will periodically retrieve the CVS module's log data from the CVS repository. If this is causing problems, consider adjusting the frequency (see below) or disabling CVS log retrieval.
- JIRA loads and parses the output of the 'cvs log' command for each CVS module and keeps 'relevant' commits in memory. Therefore JIRA's memory requirements depend on the number of relevant commits found in the CVS module. Relevant commits are CVS commits which have at least one potential JIRA key in their commit messages.
- Only commit messages which contain a possible JIRA issue key are linked to an issue.
- JIRA's 'System encoding' is used when parsing the CVS logs, so it needs to match that of the CVS log. The system encoding can be seen at Admin -> System -> System Info. See also how to set the system encoding.

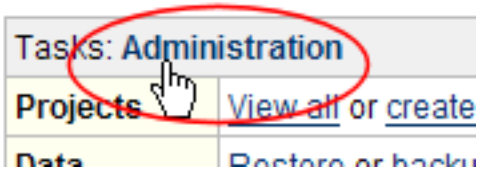
1.12.1.2. Creating CVS Modules in JIRA

A CVS 'module' refers to a top-level directory in a CVS repository. To create a CVS module:

1. Create or decide which existing directory will be used to store CVS module's log data (The file with the output of the 'cvs log' command). JIRA must have read and write access to the directory. The write access is required even if you choose to update the CVS log manually as JIRA needs to use this directory to create a lock file in order to synchronise access to the CVS module's log.
2. Log in as a user with the '**JIRA System Administrators**' global permission.
3. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

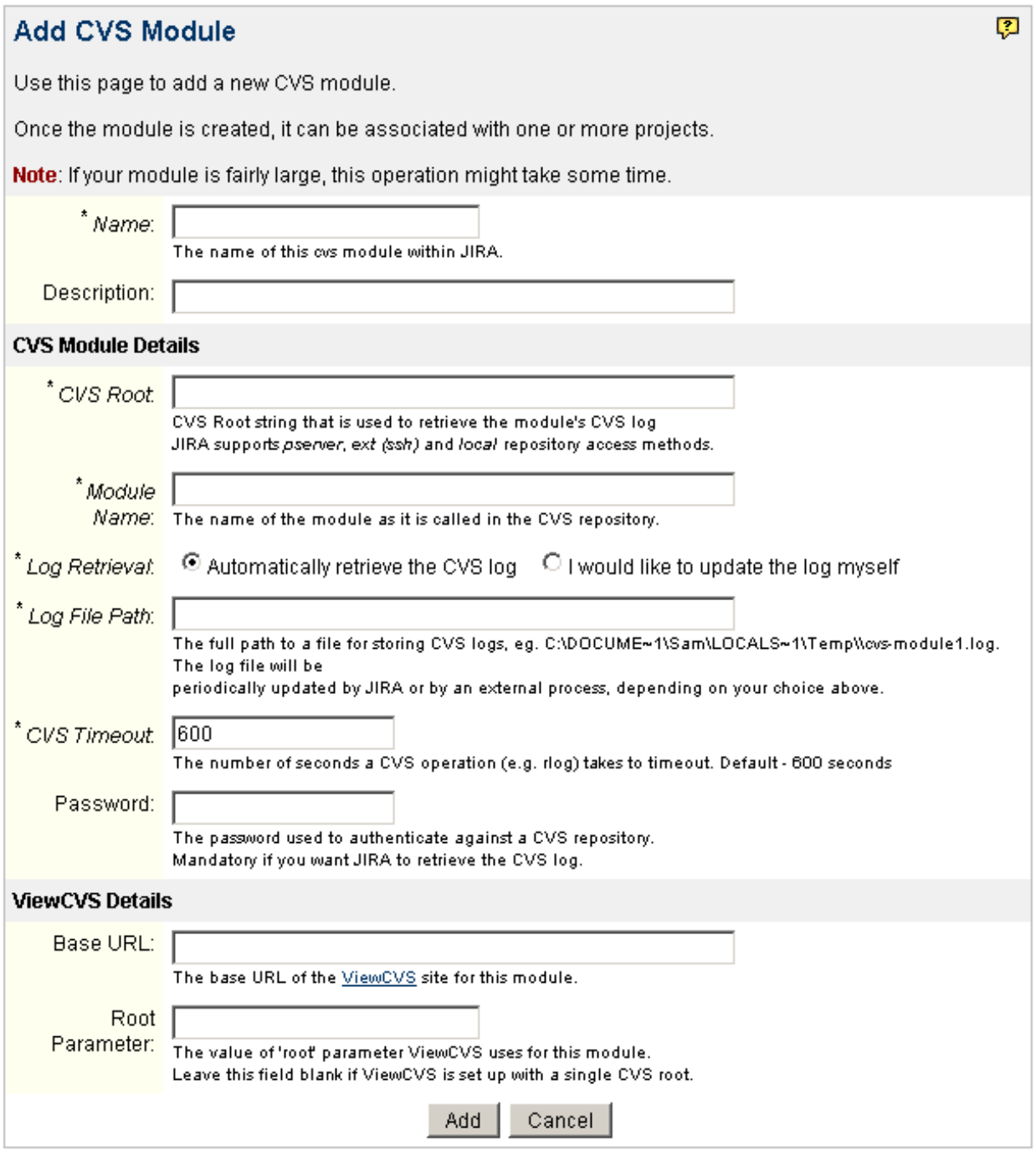


Link to Administration section



Link to Administration section

4. On the panel on the left, expand the sub-menu titled '**Global Settings**' if it is not open already. Click on the link labelled '**CVS Modules**'. This should bring up the '**CVS Modules**' page.
5. Click on the '**Add new CVS module**' link on this page.
6. This will bring up the '**Add CVS Module**' page.



Adding a CVS module

Fill in as follows:

1. For **Name** put a short descriptive name, possibly just the name of the CVS module as it appears in your CVS repository.
2. (Optional) For **Description** put a short phrase that describes this CVS module.
3. Specify **CVS Root** that will be used to retrieve the CVS module's log or was used to retrieve the log. The CVS Root is needed while parsing the log data so it is required even if you choose to retrieve CVS log manually. Please provide 'full' CVS Root details. For example:
   - /some/local/path (for local repository access)
   - :pserver:username@hostname:port/some/path (for pserver access)
   - :ext:username@hostname:/some/path (for ssh access)

   If JIRA finds trouble understanding your local CVS Root (e.g. on Windows systems) please prefix the path with *:local:*. For example, :local:d:\some\path.
4. For **Module Name** specify the name of the module as it is called in the CVS repository. This will usually be the top-level directory (eg. `myproject`), but can also include subdirectories (`myproject/subproject/src/java`) - basically anything that can be parsed to a `cvs checkout` command.

   This information is required to retrieve the CVS log as well as to parse it, so you will need to provide the module's CVS name even if you choose to retrieve the CVS log manually.
5. For **Log Retrieval** choose whether you would like JIRA to automatically synchonise with the CVS repository. If you choose '**Automatically retrieve the CVS log**', JIRA will periodically retrieve the CVS log for the module automatically and then parse it for commit information. If you choose '**I would like to update the log myself**', JIRA will not retrieve the log, but will periodically just parse it. If you choose this option you will need to update the CVS log by other means (e.g. manually or using a scheduled script) to keep the CVS information in JIRA current.
6. For **Log File Path** specify the full path to the **file** that will contain the CVS log data. This file should be located in a directory mentioned in step 1. If you would like JIRA to periodically update the contents of the log this file does not need to exist at the moment, as JIRA will automatically create it. If you choose to manually update the file please ensure that the log file already exists at the specified path and is readable by JIRA.
7. For **CVS Timeout**, specify how many seconds it takes the CVS operation (e.g. rlog) to timeout
8. The **Password** needs to be provided only if you let JIRA automatically retrieve the module's CVS log. Please specify the password that is needed to retrieve the log using the method specified in the CVS Root. If no password is required, leave the field empty.
9. (Optional) For **Base URL** in the '**ViewCVS Details**' section of the page, enter the fully qualified URL (i.e. include 'http://' or 'https://' at the beginning) to the ViewCVS site of the CVS module. The URL needs to point to the root of the module on the ViewCVS site.

Note:

If you are integrating with FishEye you do not need to perform any special steps. FishEye can resolve all the URLs that ViewCVS expects. You just need to enter the fully qualified URL to your FishEye installation and the specific repository you wish to view. This is the same URL you would get if you were to browse to the project within FishEye.

10. (Optional) For **Root Parameter** in the '**ViewCVS Details**' section of the page, enter the name of the Project Root that is used in ViewCVS to navigate the CVS module. This parameter is required only if ViewCVS has been set up to work with multiple CVS modules, and this module is not the default module on the ViewCVS server. The value that should be placed in this field is the same as the value of the 'root' URL parameter that appears on every ViewCVS URL (e.g. when viewing a file). If the URL that appears in

your browser when viewing a file from this CVS module on ViewCVS does not have the 'root' parameter, leave this field blank.

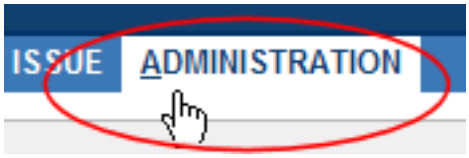
7. Click the '**Add**' button.
8. This should bring you back to the '**CVS Modules**' page, where you should see the new CVS module listed. You can edit and delete this module here.

Note:

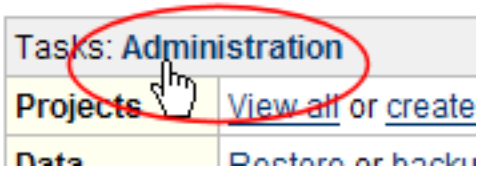
If JIRA has trouble understanding your local CVS Root (e.g. on Windows systems) please prefix the path with :local:. For example, :local:d:\some\path

1.12.1.3. Associating Project(s) with CVS Modules

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. In the left-hand navigation panel, click the '**Projects**' link.
4. This will bring up the '**Projects**' page. It lists all the existing projects. Select a project that you would like to associate with the CVS module.
5. The project's summary page sill be displayed. Next to '**CVS Modules**' click the '**Celect Module**' link. This will display the '**Select Version Control Modules**' page, where you can associate the project with a CVS module (or with multiple CVS modules, in JIRA Professional and Enterprise editions).
6. Select the appropriate module(s), and click the '**Select**' button.

1.12.1.4. Configuring Permissions

The 'View Version Control' permission needs to be given to users/groups/roles that should be allowed to see CVS commit information. Note: by default this permission is given to the 'jira-developers' group. Please read the Project Permissions section, and follow the instructions given there to assign the 'View Version Control' permission.

1.12.1.5. Disabling Automatic CVS Log Retrieval

To disable automatic CVS log retrieval for a CVS module please choose the '**I would like to update the log myself**' option for the module's '**Log Retrieval**' attribute.

If you have disabled automatic CVS log retrieval for the CVS module, JIRA will only *parse* the CVS log periodically. Therefore, for the new commit information to appear in JIRA, the log needs to be updated by other means. This can either be done manually, or a scheduled CVS update script can be used.

Note:

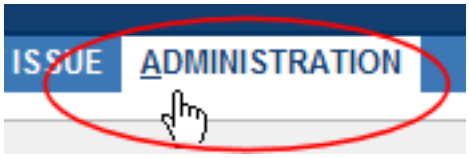
Before updating the module's CVS log, please check for the existence of a lock file with name '**cvslog.write.lock**' in the *same* directory as the CVS log file. If the lock file exists, please wait until it is removed before updating the log.
When updating the CVS log for a module, please create a lock file with the name **cvslog.write.lock** in the *same* directory as the CVS log file to ensure that JIRA does not start parsing the log while it is still being updated. Please do not forget to remove the lock file after the update has finished.

1.12.1.6. Adjusting the Frequency of Module Updates

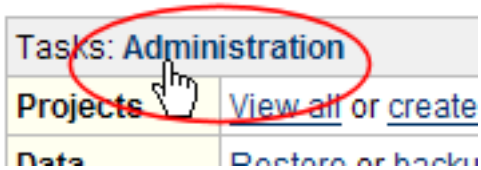
To minimise the network traffic between JIRA and the CVS server, JIRA updates and re-parses the commit information of the associated CVS modules only once during the specified period of time. By default, this period of time is 1 hour, but it can be adjusted if required.

When the first CVS module is created in JIRA, a background service is automatically started. The service is called '**VCS Update Service**'. To change the frequency of the module updates, follow these steps:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Open the '**System**' tab of the left-hand menu, if it is not already open.
4. Select '**Services**' from the '**System**' tab. A page showing all the configured services will appear. If at least one CVS module has been configured, the '**VCS Update Service**' should be present in the list.
5. Click the '**Edit**' link in the right-most column of the '**VCS Update Service**'. This will display a page where you can set the delay for the service.
6. Change the value as required. Remember that the delay is specified in minutes.
7. Click the '**Update**' button to make the changes take effect.

Please keep in mind:

- The CVS modules are updated one after another every specified period of time. That is, it is not possible to specify a different update delay for each configured CVS module.
- If you are using automatic log retrieval for your CVS modules and you set the delay to a very low value, the bandwidth consumption between JIRA and the CVS server might be very high.
- If the delay is set to a very large value, the 'new' cvs commit messages will not appear in JIRA for some time.

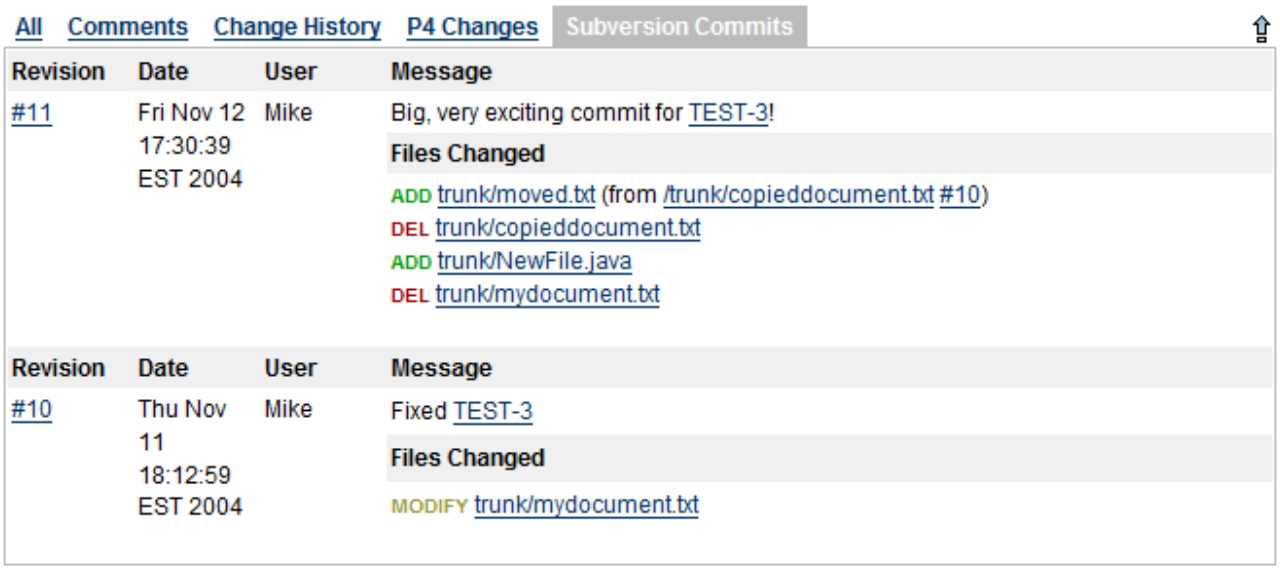
1.12.1.7. CVS Aliases

JIRA does not currently support CVS aliases. If you have a CVS alias that references more than one module, please create each CVS module in JIRA and then associate each module with the relevant JIRA project(s) (assuming you are using JIRA Professional or Enterprise edition, as Standard edition only supports one CVS module per project). The feature request for adding CVS

alias module support to JIRA is JRA-4586. Please vote for the issue to increase its popularity. Please refer to this document, which describes the way Atlassian implements new features and improvements.

1.12.2. Subversion integration

JIRA's Subversion integration lets one see Subversion commit information relevant to each issue. Subversion integration is implemented as a plugin (drop-in extension) to JIRA.



Subversion tab screenshot

Commits will appear in this tab if the commit log mentions the issue key ('TEST-3' above).

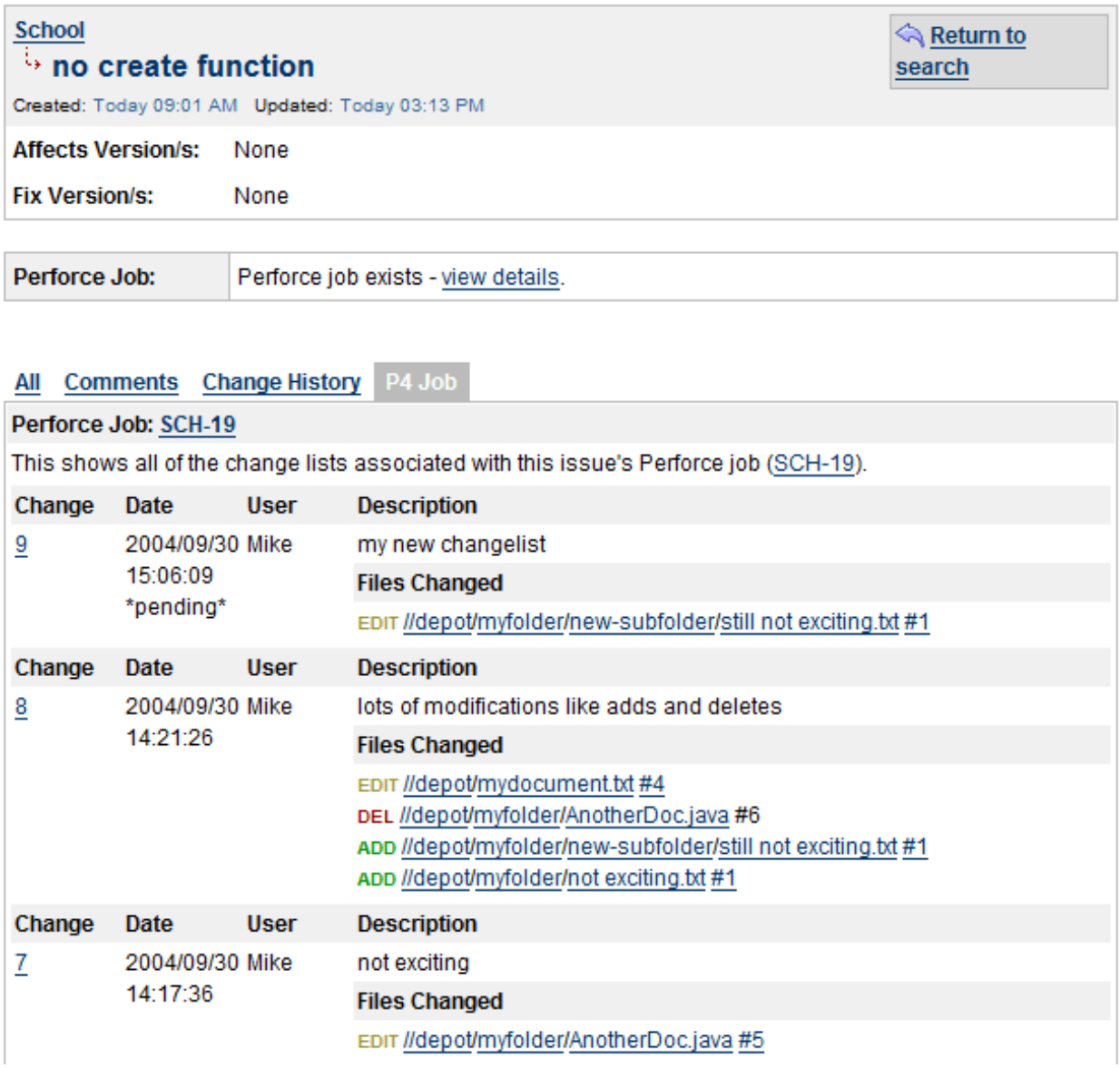
For more information, see the Subversion plugin page online.

1.12.3. Perforce integration

Note:

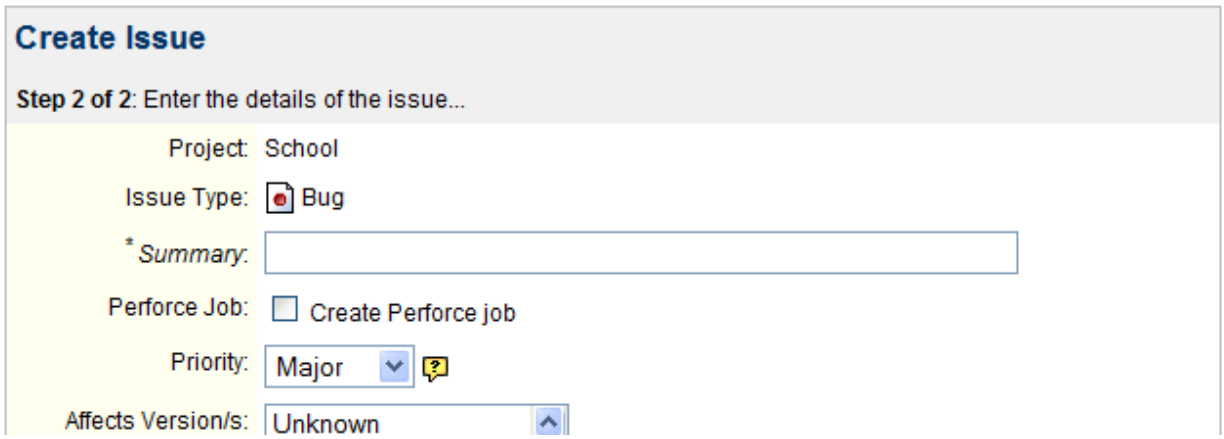
The Perforce Plugin for JIRA is now **deprecated**, as it has been superseded by the FishEye plugin for JIRA. Please ensure that you read the the Perforce plugin page for further information, including details on support and our recommended FishEye solution.

JIRA's Perforce integration lets you see Perforce commit information relevant to each issue:



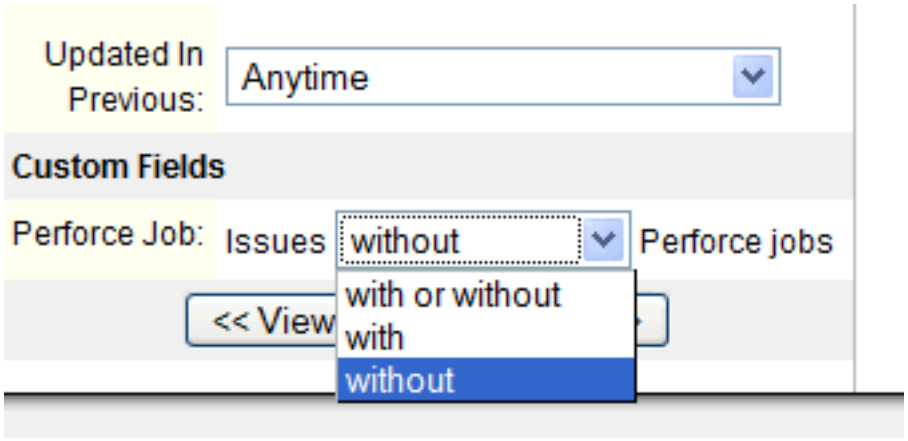
Viewing an issue with its associated Perforce job and linked changelists.

This includes two-way integration with Perforce Jobs, for example creating a job when an issue is created:



Issue create screen showing optional creation of P4 job

and searching for JIRA issues with associated perforce jobs:

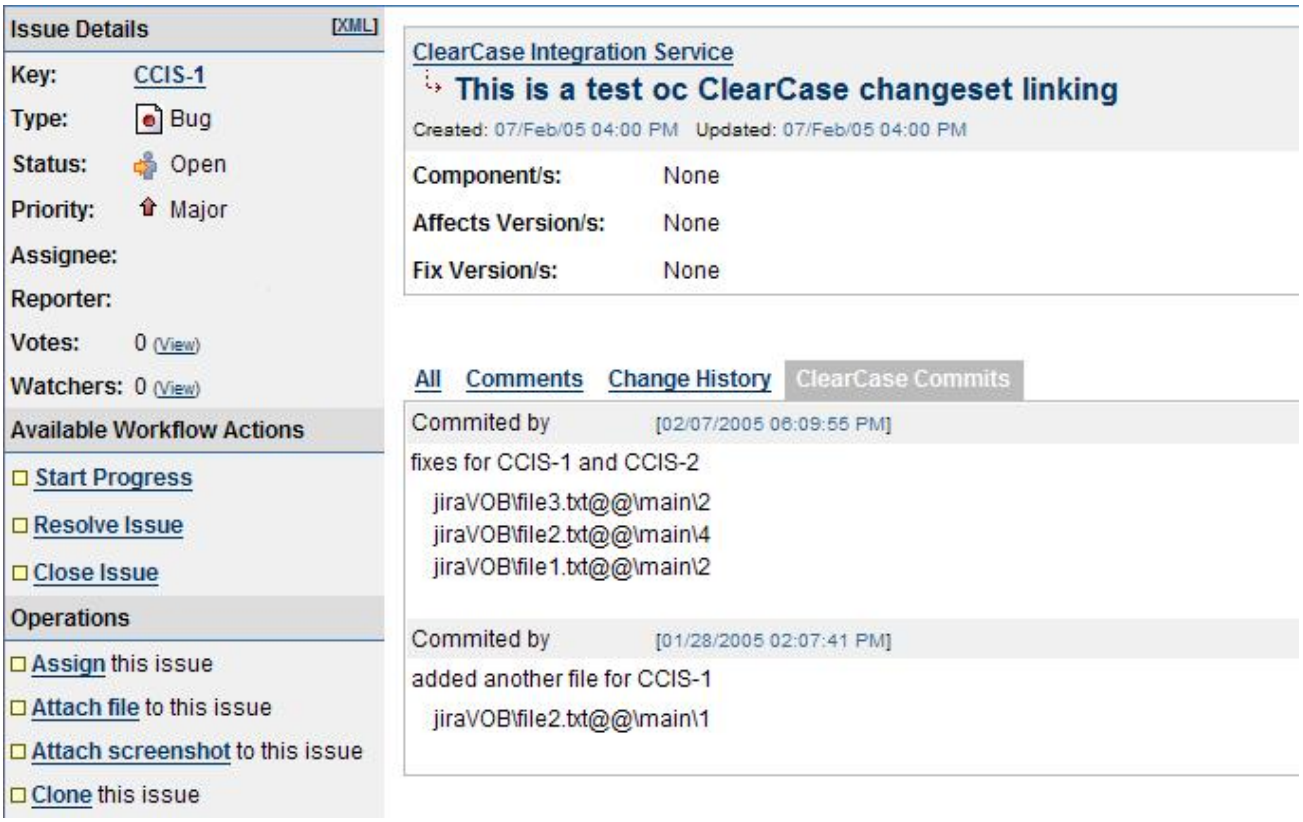


Issue navigator interface to query by perforce job.

Perforce integration is implemented as a plugin (drop-in extension) to JIRA, which is licensed separately to JIRA.

1.12.4. ClearCase integration

Although not developed or supported by Atlassian, there is a JIRA ClearCase plugin available which you may find useful. It shows ClearCase checkins associated with JIRA issues:



ClearCase checkins associated with an issue

This can be combined with the Commit Acceptance Plugin to get further towards the level of integration ClearQuest offers.

1.12.5. FishEye integration

JIRA's FishEye integration allows you to browse your source-control repository from inside JIRA, provided you are using Atlassian FishEye with your source-control repository. FishEye integration is implemented as a plugin (drop-in extension) to JIRA.

FishEye integration will allow you to:

- View an Issue's FishEye Changesets
- Browse a Project's FishEye Changesets

For more information, see the FishEye plugin page.

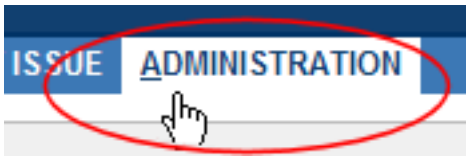
1.13. Configuration Options & Settings

1.13.1. Configuring JIRA

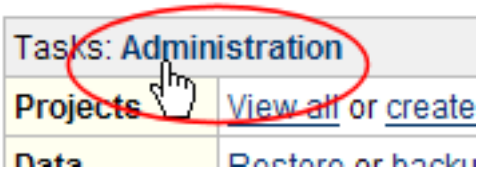
JIRA has a number of configuration options that allow the system to be customised for use within your organisation.

To manage your JIRA configuration:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

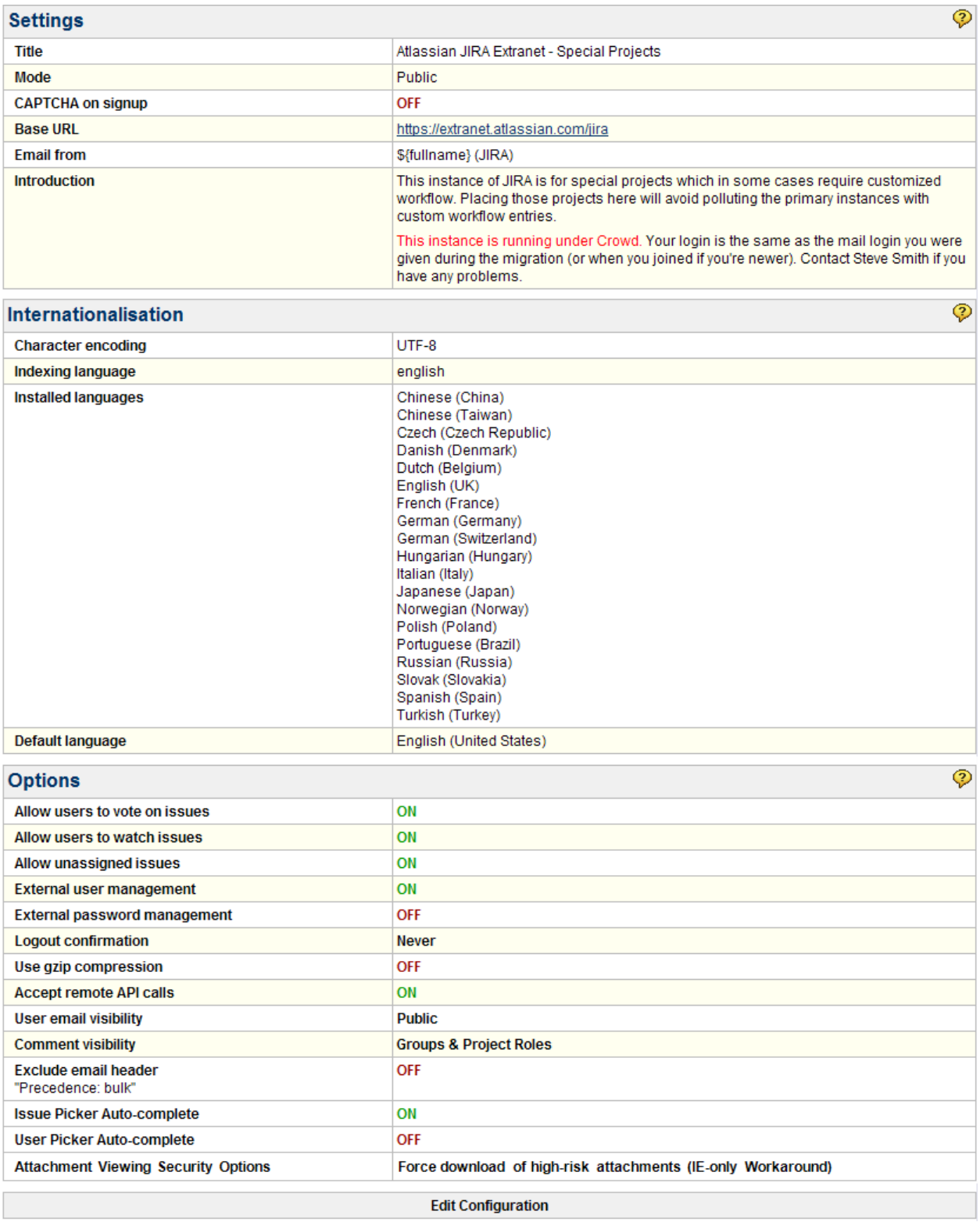


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title 'Global Settings', click on the link labelled 'General Configuration'.
4. The following screen will be displayed. Click 'Edit Configuration' to edit the three sections as described below.



Settings	
Title	Atlassian JIRA Extranet - Special Projects
Mode	Public
CAPTCHA on signup	OFF
Base URL	https://extranet.atlassian.com/jira
Email from	${fullname} (JIRA)
Introduction	This instance of JIRA is for special projects which in some cases require customized workflow. Placing those projects here will avoid polluting the primary instances with custom workflow entries. This instance is running under Crowd. Your login is the same as the mail login you were given during the migration (or when you joined if you're newer). Contact Steve Smith if you have any problems.

Internationalisation	
Character encoding	UTF-8
Indexing language	english
Installed languages	Chinese (China) Chinese (Taiwan) Czech (Czech Republic) Danish (Denmark) Dutch (Belgium) English (UK) French (France) German (Germany) German (Switzerland) Hungarian (Hungary) Italian (Italy) Japanese (Japan) Norwegian (Norway) Polish (Poland) Portuguese (Brazil) Russian (Russia) Slovak (Slovakia) Spanish (Spain) Turkish (Turkey)
Default language	English (United States)

Options	
Allow users to vote on issues	ON
Allow users to watch issues	ON
Allow unassigned issues	ON
External user management	ON
External password management	OFF
Logout confirmation	Never
Use gzip compression	OFF
Accept remote API calls	ON
User email visibility	Public
Comment visibility	Groups & Project Roles
Exclude email header "Precedence: bulk"	OFF
Issue Picker Auto-complete	ON
User Picker Auto-complete	OFF
Attachment Viewing Security Options	Force download of high-risk attachments (IE-only Workaround)

Edit Configuration

Screenshot of JIRA configuration settings and options

1.13.1.1. Settings

Setting	Description
Title	This is the title that will be displayed on the JIRA login page and the dashboard. It helps identify your installation and its purpose. (Note: also see logo, which is displayed on every JIRA page.)

Mode	JIRA can operate in two modes: **Public** — Anyone can issues (within permissions). **Private** — Useful for inte you do not wa disabled; only users. Default: **Public**
CAPTCHA on signup	If you are running JIRA in Public mode (see above), it is strongly recommended that you enable Captcha. Default: **ON**
Base URL	This is the base URL of this JIRA installation. It is used in outgoing email notifications to construct valid links to JIRA issues.
Email From Header	Specifies the **From:** header format in notification emails. Default is of the form "John Doe (JIRA) <jira@company.com>". Available variables are $'{fullname'}, $'{email'} and $'{email.hostname'}.
Introduction	A short introduction message displayed on the dashboard. (Note: also see the announcement banner, which is displayed on *every* JIRA page.) It is possible to include HTML, but ensure all tags are correctly closed and that the HTML is well formed.

1.13.1.2. Internationalisation

Setting	Description
Character encoding	The character encoding for input and viewing of information within JIRA. For most western languages, the default ("UTF-8") is suitable. If you change this setting, ensure that you also change your database's encoding. View a list of supported encodings. However, please use the IANA preferred MIME name (such as 'iso-8859-1' instead of 'ISO8859_1') to ensure XML backups have the correct encoding string.
Indexing language	JIRA uses Lucene, a high-performance text search engine library, in full-text searches over the information stored in JIRA. This option should be set to the language in which information is entered into JIRA. It is necessary to re-index JIRA if this value is changed.
Installed languages	This section lists all language packs available within the JIRA system. (Note: to install

	additional languages, see Internationalisation.)
Default language	This is the language used throughout the JIRA interface (as selected from the list displayed in **Installed Languages** above). Users override this language by using the **Language** setting in their user profile.

1.13.1.3. Options

Setting	Description
Allow users to vote on issues	Controls whether voting is enabled in JIRA. Voting allows users to indicate a preference for issues they would like to be completed or resolved. See also the 'View Voters and Watchers' permission. Default: **ON**
Allow users to watch issues	Controls whether watching is enabled in JIRA. Users can 'watch' issues which they are interested in. Users watching an issue will be notified of all changes to it. See also the 'View Voters and Watchers' and 'Manage Watcher List' permissions. Default: **ON**
Allow unassigned issues	When turned **ON**, JIRA will allow issues to be unassigned or assigned to 'no-one'. When turned **OFF**, issues must always be assigned to someone - by default, the assignee will be the **Project Lead** as defined for each project. Default: **OFF**
External user management	When turned **ON**, JIRA will assume that you are managing users from outside JIRA (e.g. using Crowd). This means that you will no longer be able to create, edit or delete users/groups from within JIRA (or via email or import); but you can still assign users/groups to project roles, and create/edit/delete user properties. Additionally, JIRA will not display options for users to change their password, or edit their profile. **Note:** JIRA's LDAP integration is currently limited to external *password* management, so this option should be left **OFF** when using LDAP. Default: **OFF**
External password management	When turned **ON**, JIRA will assume that you are managing passwords from outside JIRA. JIRA will not display options for users to change their password, or display the 'Forgot Password' link on the login screen. **Note:** With the default osuser LDAP provider, this option should be turned **ON**, as accounts are not yet stored in LDAP and this option only hides the

	password features within JIRA. Default: **OFF**
Logout confirmation	Controls whether to obtain user's confirmation when logging out. **cookie** means prompt for confirmation if the user was automatically logged in (via a cookie). Default: **NEVER**
Use GZip compression	Controls whether to compress the web pages that JIRA sends to the browser. It is recommended that this be turned ON, unless you are using mod_proxy. Default: **OFF**
Allow remote API access	Controls whether to allow remote client access (via XML-RPC or SOAP) to this JIRA installation for authenticated users. Default: **OFF**
User email visibility	Controls how user email addresses are displayed in the user profile page. **PUBLIC** — email addresses are visible to all. **HIDDEN** — email addresses are hidden from all users. **MASKED** — the email address is masked (e.g. 'user@example.com' is displayed as 'user at example dot com'). **LOGGED IN USERS ONLY** — only users logged in to JIRA can view the email addresses Default: **PUBLIC**
Comment visibility	Determines what will be contained in the list that is presented to users when specifying comment visibility and worklog visibility. **Groups & Project Roles** — the list will contain groups and project roles. **Project Roles only** — the list will only contain project roles. Default: **Project Roles only**
Exclude email header 'Precedence: bulk'	Controls whether to prevent the **Precedence: Bulk** header on JIRA notification emails. This option should only be enabled when notifications go to a mailing list which rejects 'bulk' emails. In normal circumstances, this header prevents auto-replies (and hence potential mail loops). Default: **OFF**
Issue Picker Auto-complete	Provides auto-completion of issue keys in the 'Issue Picker' popup screen. Turn OFF if your users'

	browsers are incompatible with AJAX. Default: **ON**
User Picker Auto-complete	Provides auto-completion of user names in the 'User Picker' popup screen. Turn OFF if you have a very large number of users, or if your users' browsers are incompatible with AJAX. Note: If '**User email visibility**' (see above) is set to HIDDEN, the users' email addresses will not be searched or shown in the auto-complete results. Default: **ON** if you have less than 5,000 users (or **OFF** if you have more than 5,000 users).
Attachment Viewing Security Options	Changes default browser action for attachments in JIRA. Options are; • Force download of all attachments (Secure) — force the download of all attachments. This is the most secure option, but is less convenient for users. • Force download of high-risk attachments (IE-only Workaround) — force the download of attachments in IE that the IE would mistakenly detect as an HTML file. Declared HTML attachments are also never displayed inline. • Allow inline display of attachments (Insecure) — allows all attachments to be displayed inline. Only select this option if you fully understand the security risks. See JIRA Security Advisory 2008-08-26 for further details. Default: **Force download of high-risk attachments (IE-only Workaround)**

1.13.1.4. Advanced properties

There are a handful of other properties (usually storing defaults of little interest to most JIRA users) in the **WEB-INF/classes/jira-application.properties** file, which you may want to edit.

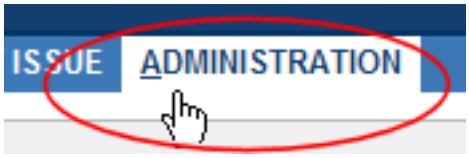
1.13.2. Configuring Issue Linking

Issue Linking allows you to create an association between issues. For instance, an issue may *duplicate* another, or its resolution may *depend* on another's.

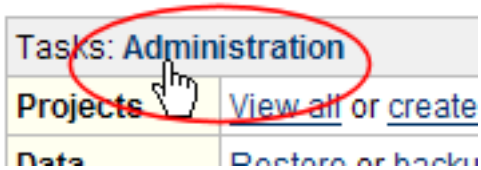
There are two steps involved in configuring 'Issue Linking'. The first step is configuring JIRA to allow issue links. This is a global setting. The second step is to the create the issue link types required. These are also global.

1.13.2.1. Step 1: Turning on Issue Linking.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

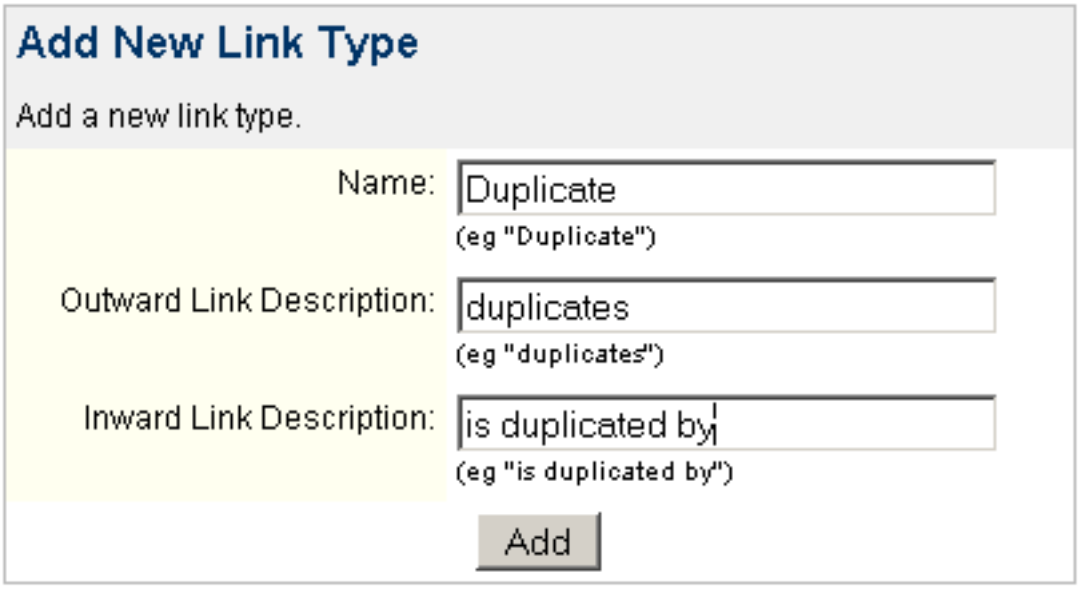


Link to Administration section

3. On the panel on the left, under the title "Global Settings", click on the link labelled "Issue Linking".
4. You will be shown a status page stating whether linking is enabled. If it is not, click the "Activate" button.
5. You will be shown the status page again, this time stating that Issue Linking is enabled.

1.13.2.2. Step 2: Creating Issue Link Types.

1. On the Issue Linking page there will now be a form titled "Add New Link Type".



Form for adding a new link type

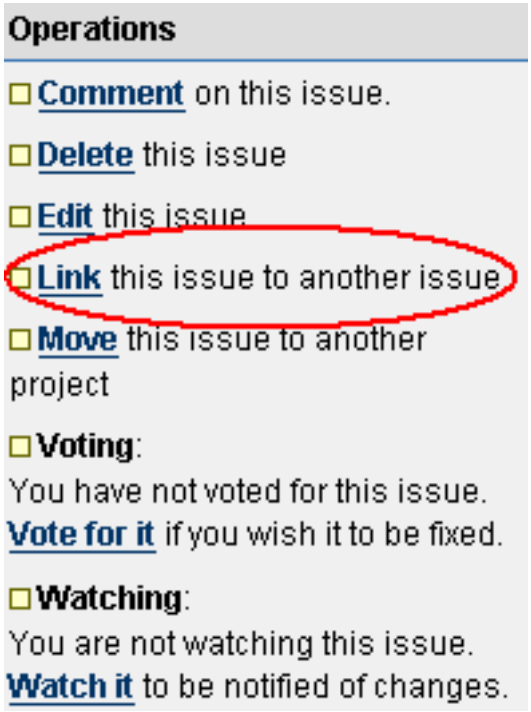
2. To create a new link type, say Duplicate, proceed as follows:
   - enter "Duplicate" in the "Name" text field
   - enter "duplicates" in the "Outward Link Description" text field
   - enter "is duplicated by" in the "Inward Link Description" text field
3. Click the add button.
4. This returns to the link type management page, with a new section listing the "Duplicate" issue linking type. Here you can edit or delete the relationship, as required.

Name	Outward Desc.	Inward Desc.	
Duplicate	duplicates	is duplicated by	Edit | Del

List of declared link types

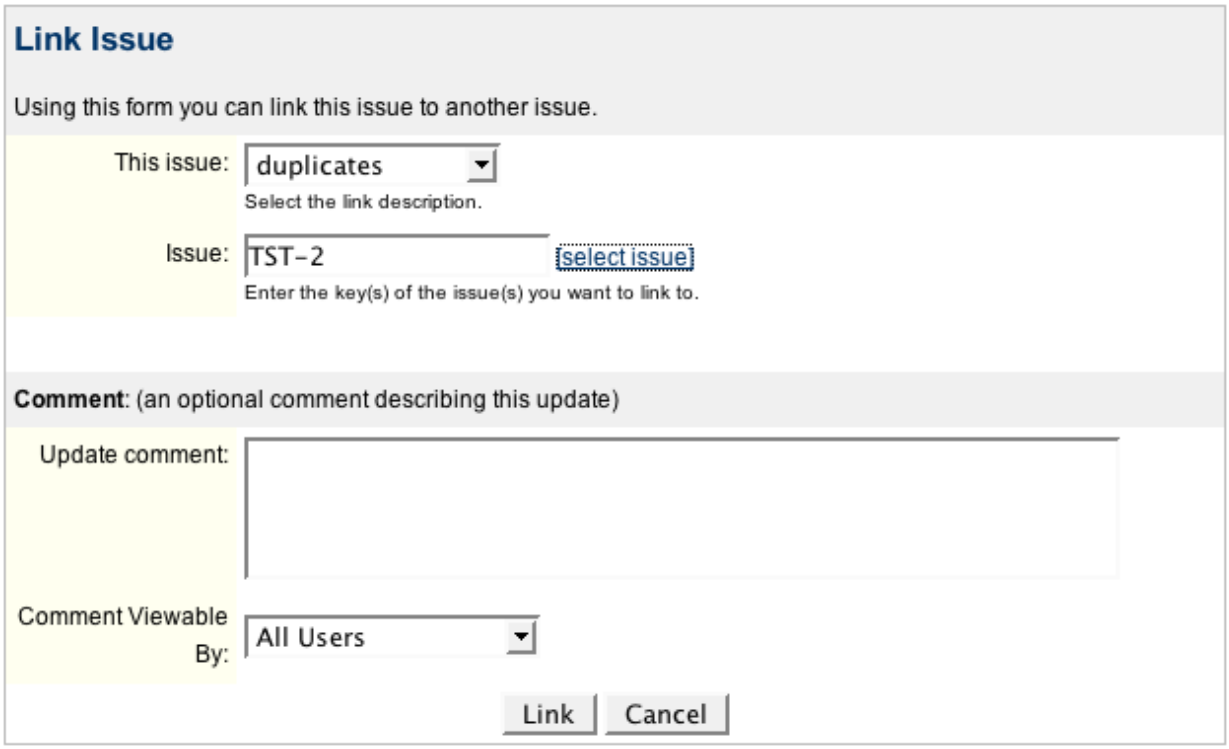
1.13.2.3. Step 3: Linking Issues.

1. To link an issue to another issue, click on the link 'Link this Issue to Another Issue' on the 'View' Issue page.



Link Issue

2. This will bring up the 'Link Issue' form.



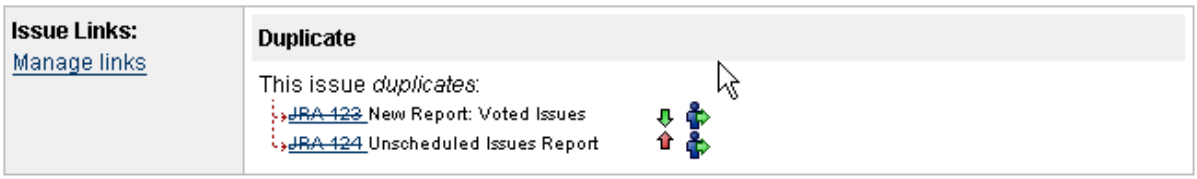
Link Issue

3. Select the link type and enter the key of the issue that you want to link to. It is also possible to link to multiple issues.

   You can optionally add a comment.

Click on the "Link" button.

4. You will see the issue page again, with a new section listing the issues that are linked to this issue



List of linked issues

1.13.3. Enabling and Configuring Sub-tasks

Sub-task issues are generally used to split up a parent issue into a number of tasks which can be assigned and tracked separately.

Sub-tasks have all the same fields as standard issues, although note that their 'issue type' must be one of the *sub-task issue types* (see below) rather than one of the standard issue types.

Once you have enabled sub-tasks and defined at least one sub-task issue type, users will be able to:

- create sub-tasks.
- convert issues to sub-tasks (and vice versa).

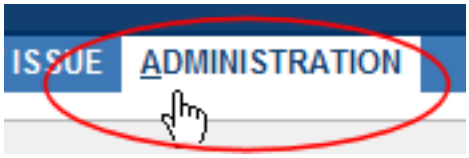
Note:

Sub-tasks are only supported in the Professional and Enterprise editions of JIRA.

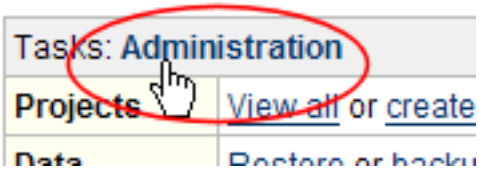
1.13.3.1. Enabling sub-tasks

Sub-tasks are disabled by default. To enable sub-tasks:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Locate the 'Global Settings' sub-menu on the left hand side of the screen, and choose *Sub-Tasks* from the list.
4. The 'Sub-Tasks' administration screen will be displayed. Click the 'Enable' link.
5. The page will reload and inform you that the sub-tasks are now enabled. A default sub-task issue type has also been automatically created. You can edit it by selecting the *Edit* link in the *Operations* column.

1.13.3.2. Defining sub-task issue types

Sub-tasks must be assigned one of the *sub-task issue types*, which are different to standard issue types. Please note that you must define at least one sub-task issue type before users can create sub-tasks.

Sub-task issue types can be customised on the 'Sub-Tasks' administration screen (which is described above). The 'Sub-Tasks' administration screen allows you to create, delete, edit, translate and choose icons for your sub-task issue types. For details, please see 'Defining Issue Types'.

1.13.3.3. Configuring sub-tasks and workflow

It is possible to restrict the progression of an issue through workflow depending on the state of the issue's sub-tasks. For example, you might need to restrict an issue from being resolved until all of its sub-tasks are resolved. To achieve this, you would create a custom workflow and use the *Sub-Task Blocking Condition* on the workflow transitions that are to be restricted by the sub-tasks' state.

1.13.3.4. Configuring sub-task fields

You can choose which subtask fields are displayed in the 'Sub-Tasks' section of an issue (see Working with sub-tasks'):

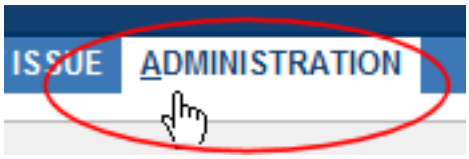
- the parent's 'View Issue' screen.
- the sub-task 'Quick Create' form.

This is done via the jira-application.properties file.

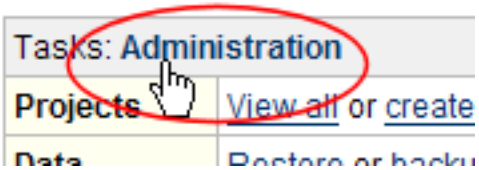
1.13.3.5. Disabling sub-tasks

Sub-tasks are disabled from the sub-task administration screen. To disable sub-tasks please follow the following steps:

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Open the *Global Settings* sub-menu on the left hand side if it is not open already, and choose *Sub-Tasks* from the list.
4. After the *Sub-Tasks* administration screen loads please click the "Disable" link. The page should reload and inform you that the sub-tasks are now disabled.

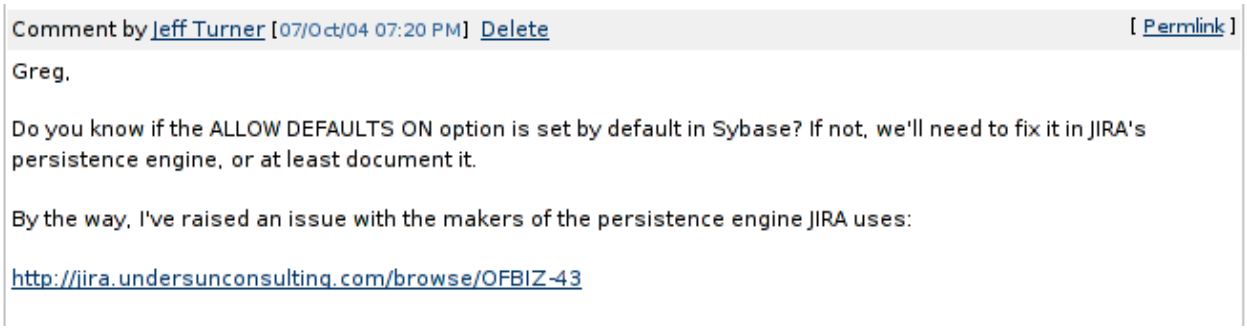
Note:

Sub-tasks cannot be disabled if one or more sub-tasks exist in the system. You will need to remove the existing sub-tasks (or convert them to standard issues) before disabling this feature.

1.13.4. Trackback linking

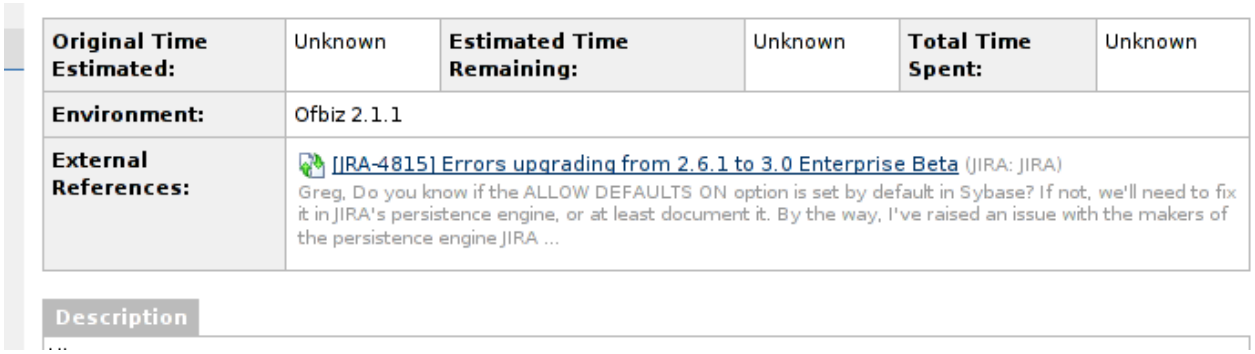
Trackback linking is a means by which a page can *tell another page* that it has been linked to. (To learn more about how trackback works, please have a look at the Trackback specification).

For instance, say that a user writes a URL in a JIRA comment:



Comment containing URL to trackback-enabled site

If the URL is to a trackback-enabled web application like a weblog, Confluence page or another JIRA site, the linked-to page will be told that it was linked to, and can automatically create a link back to the linker:



Linked-to page linking back to the linker

Note:

When linking *to* a JIRA instance, make sure its **base URL is correct.** (To check a JIRA instance's 'Base URL', go to '**General Configuration**' under '**Global Settings**' in the '**Administration**' menu.)

1.13.4.1. Configuring trackbacks

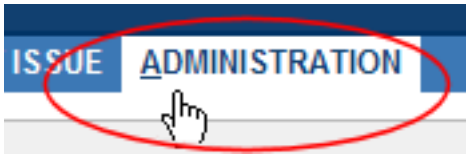
In JIRA, you can configure whether to:

- display links to external pages that link to your pages (accept incoming trackback pings)
- notify external pages that they have been linked to (send outgoing pings)

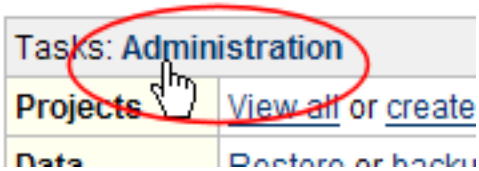
The default configuration is to display incoming links, but not notify external pages.

To configure trackbacks:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

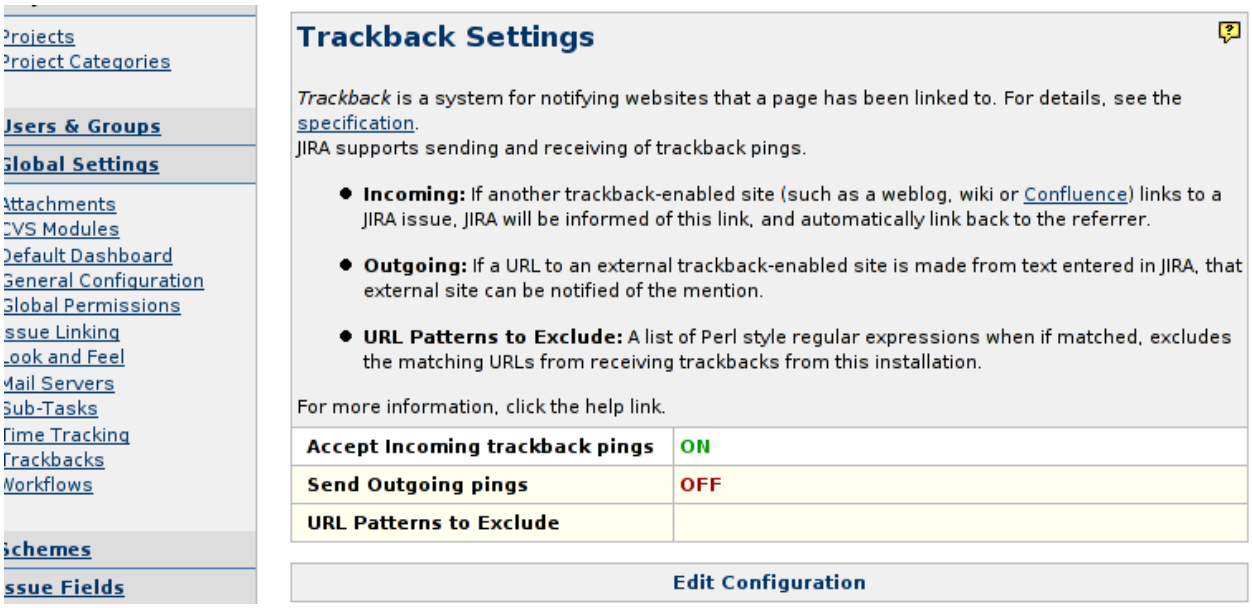


Link to Administration section



Link to Administration section

3. Under the '**Global Settings**' sub-menu in the left-hand navigation column, click the '**Trackbacks**' link. This will display the '**Trackback Settings**' page:



Default trackback configuration

4. Click the '**Edit Configuration**' link.
5. In the '**Accept Incoming Trackback Pings**' field, select '**ON**' to enable trackbacks.
6. In the '**Send Outgoing Trackback Pings**' field, either:
   - select '**On for all issues**' to always notify external sites that they have been linked to.
   - select '**On for public issues only**' to only notify external sites that they have been linked to if the issue is publicly visible.
   - select '**Off**' to never notify external sites that they have been linked to.
7. *(Optional)*In the '**URL Patterns to Exclude**' field, specify the URLs of any sites which you always want to exclude from being notified that they have been linked to. Use regular expressions (one per line), e.g.:
   ```
   .*server.domain.com.*
   .*server2.domain.com.*
   .*domain2.com.*
   ```
8. Click the '**Update**' button.

1.13.4.2. Temporarily disabling trackbacks

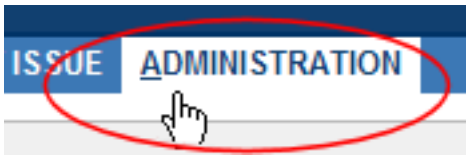
Trackback pings can be disabled (e.g. during a data import) by setting the **jira.trackback.senddisabled=true** flag on startup.

1.13.5. Configuring Time Tracking

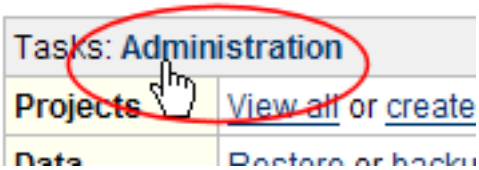
To enable time tracking in JIRA, you must first activate it and then assign permissions for users to log work on issues.

1.13.5.1. Step 1: Activating Time Tracking.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

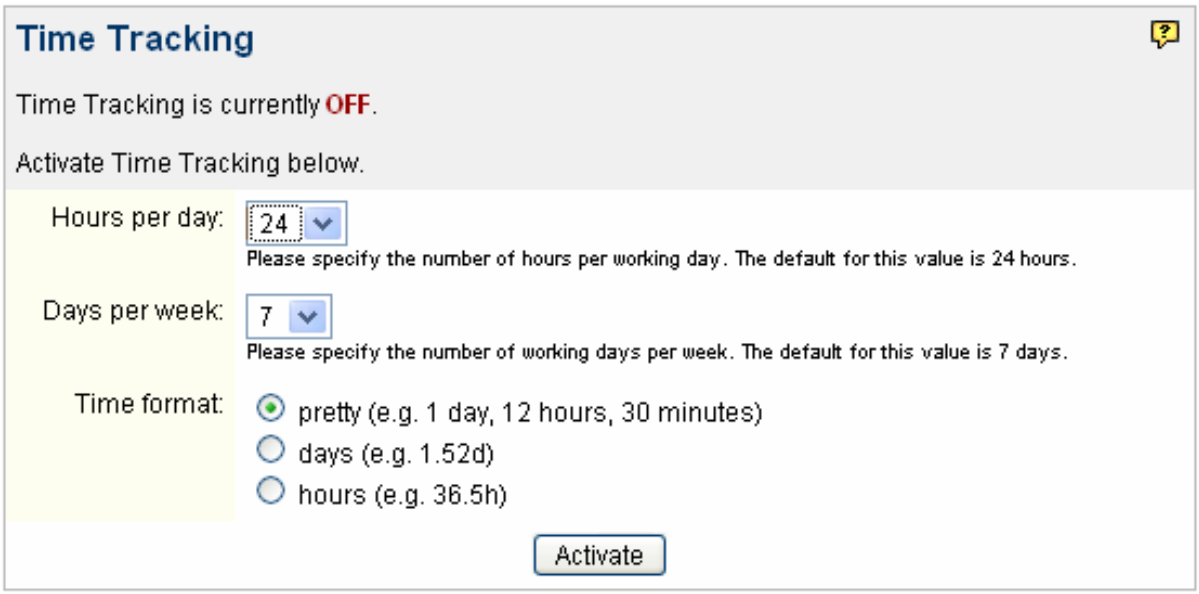


Link to Administration section



Link to Administration section

3. On the panel on the left, under the title "Global Settings", click on the link labelled "Time Tracking". By default, time tracking is OFF:



Activate Time Tracking

4. Select a suitable value for **Hours Per Day** (e.g. 8)
5. Select a suitable value for **Days Per Week** (e.g. 5)
6. Select your preferred **Time Format**. This will determine the format of the 'Time Spent' field when an issue is displayed.
7. Click **Activate** to turn time tracking ON.

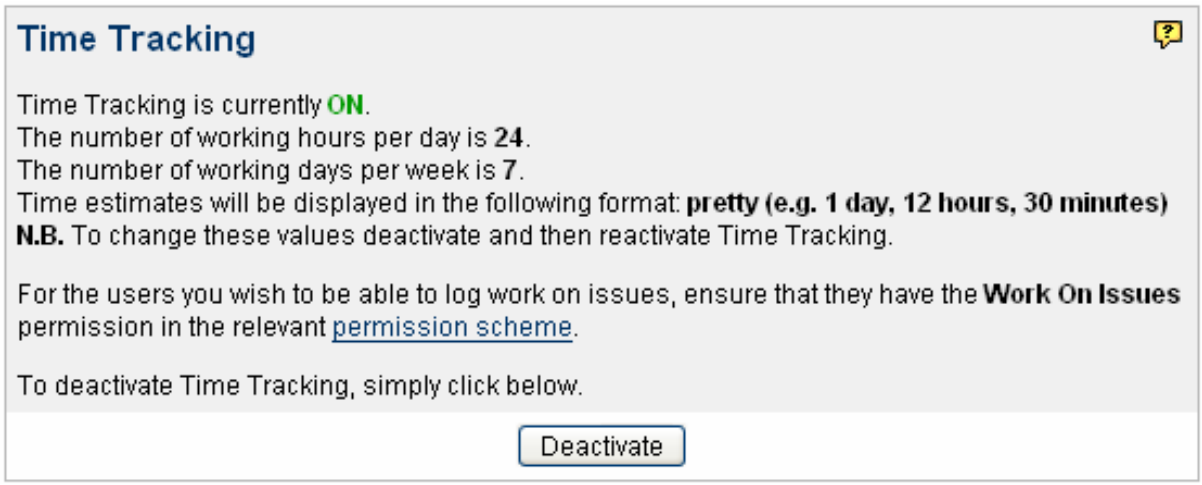
Note:

To change the **Hours Per Day** and **Days Per Week** once Time Tracking is activated, you will need deactivate and then reactivate Time Tracking with the new values.

1.13.5.2. Step 2: Allowing users to log work.

To be able to log work on issues, users, groups or project roles must first be assigned permissions to the appropriate project(s) as follows:

1. Once you have activated time tracking (see above) you will see the following screen:



Display once time tracking is enabled

2. Click the **Permission Schemes** link.
3. Select the permission scheme associated with the project(s) for which you wish to allow work to be logged.
4. Check whether the row labelled 'Work On Issues' contains the appropriate users, groups or project roles. If it doesn't, click the **Add** link:

Adding the Work on Issues permission

5. Then select the user, group or project role you wish to be able to log work on issues. Then click the "Add" button.

1.13.6. Configuring File Attachments

When you enable file attachments, you allow users to attach files and screenshots to JIRA issues. This requires space on the server to save the attachments into. Note that attachments are not stored in JIRA's database and so will need to be backed up separately.

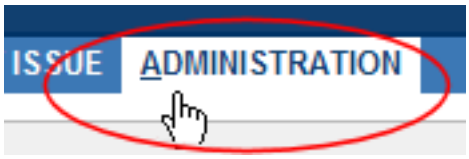
Configuring file attachments takes two steps:

1. Enabling attachments.
2. Granting the '**Create Attachments**' permission to appropriate users.

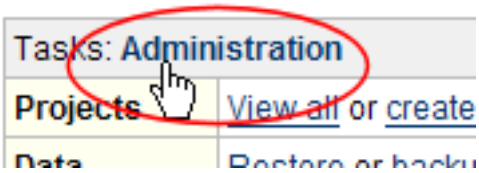
Additionally, if you wish to allow users to attach a file *when creating a new issue*, you need to ensure that the '**Attachment**' field is not hidden within the field configuration(s) associated with the specific issue type(s).

1.13.6.1. Step 1: Enabling attachments

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

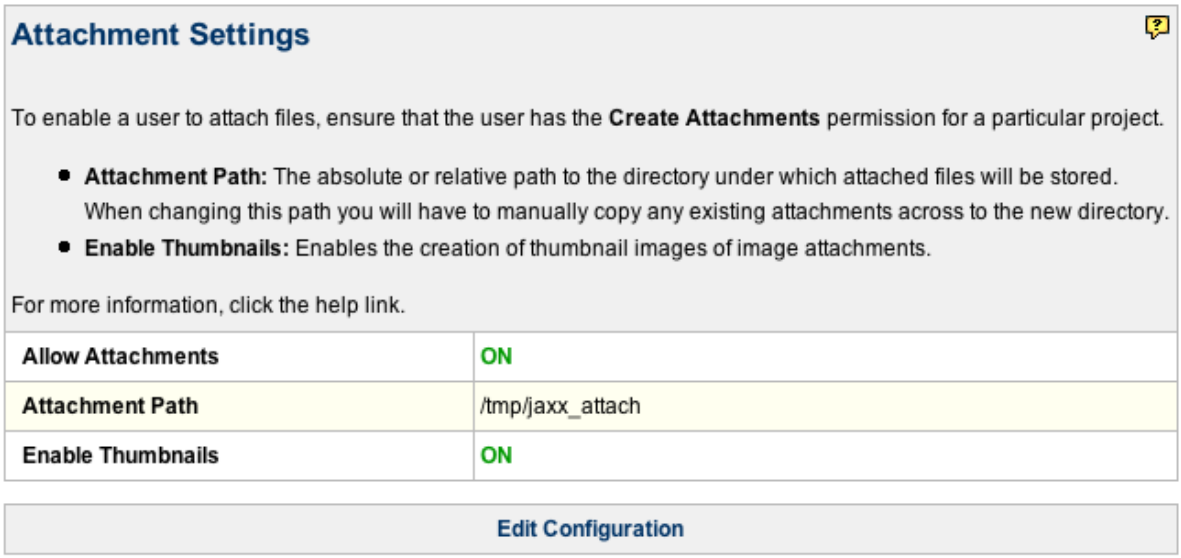


Link to Administration section



Link to Administration section

3. Under the '**Global Settings**' sub-menu in the left-hand navigation column, click the '**Attachments**' link. This will display the '**Attachment Settings**' page, which states whether attachments are on or off:



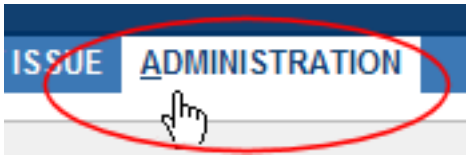
Page for enabling attachments

4. Click the '**Edit Configuration**' link.
5. In the '**Allow Attachments**' field, select '**ON**'.
6. In the '**Attachment Path**' field, type the absolute or relative path to the directory where attachments will be stored. Note that this directory should be given appropriate security as described in Security Overview.
7. *(Optional)* In the '**Enable Thumbnails**' field, select '**ON**' if you wish to enable image attachments to be displayed as thumbnails. For details please see *'Image attachment thumbnails'*.
8. Click the '**Update**' button.

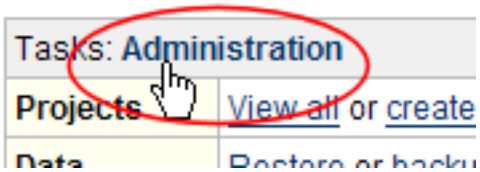
1.13.6.2. Step 2: Granting the 'Create Attachments' permission to users

You now need to grant the '**Create Attachments**' permission to appropriate users in the permission scheme(s) of project(s) for which you wish to allow attachments.

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

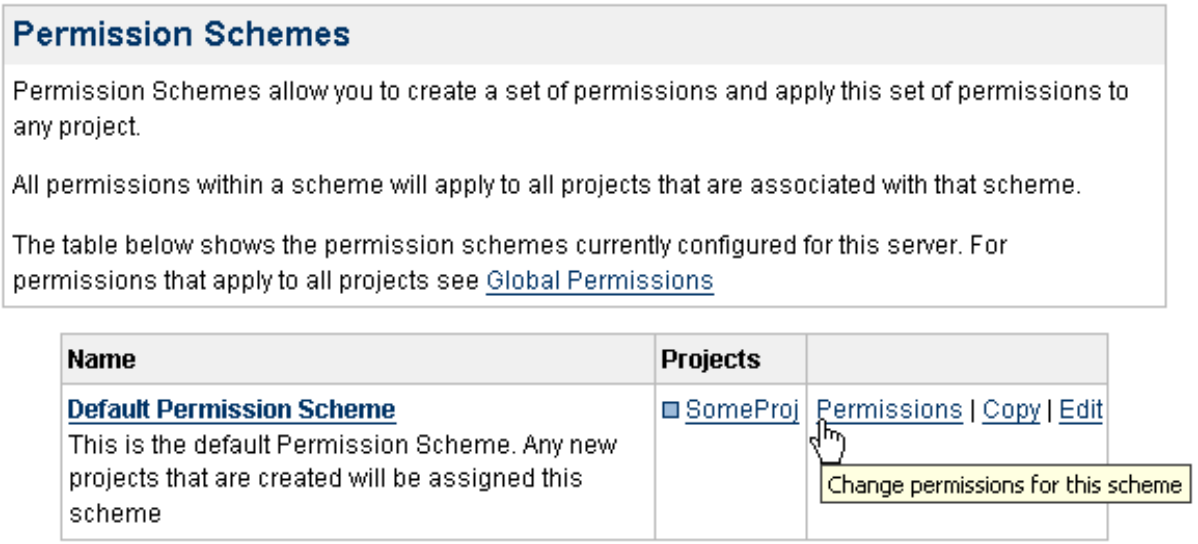


Link to Administration section



Link to Administration section

3. Under the '**Schemes**' sub-menu in the left-hand navigation column, click the '**Permission Schemes**' link. This will display a list of all permision schemes in your JIRA system, and the projects which use each scheme.
4. For each relevant permission scheme,
    1. Click the '**Permissions**' link to edit the scheme.

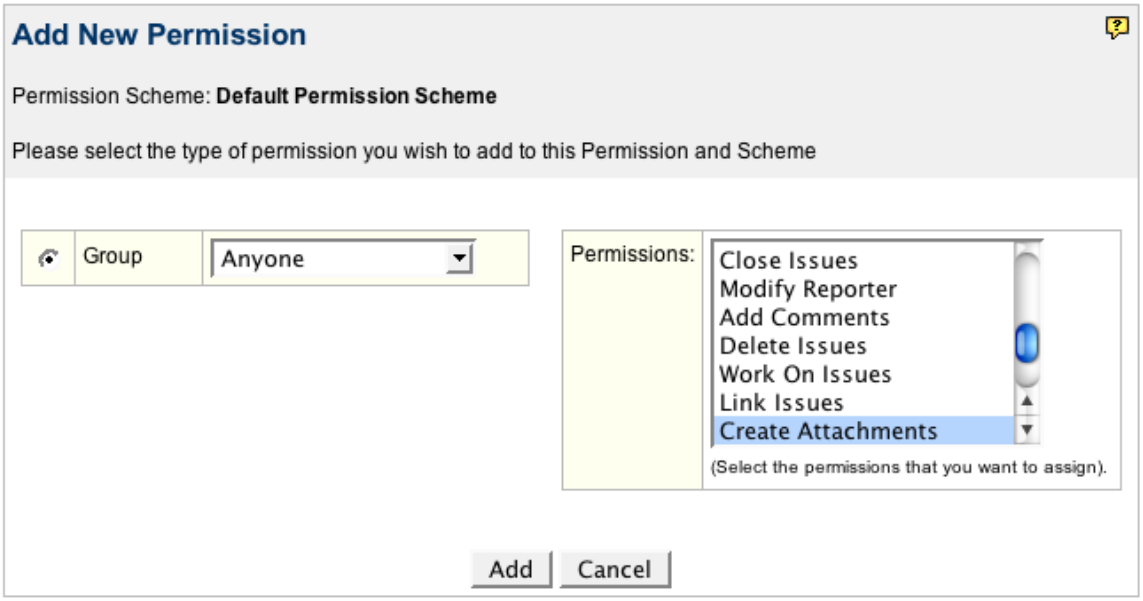


Permission Schemes

    2. In the '**Permissions**' drop-down list, find '**Create Attachments**', and click the '**Add**' link.

Adding a permission to a scheme

    3. In the '**User-Group**' drop-down list, select the relevant group. Then click the '**Add**' button.



Adding a permission to a scheme

1.13.6.3. Changing the attachment size limit

The default maximum attachment size is 10Mb. This can be changed, but not yet through the web interface (see JRA-2994).

To change the maximum attachment size, edit the `atlassian-jira/WEB-INF/classes/webwork.properties` file and change the `webwork.multipart.maxSize` parameter.

1.13.6.4. Specifying the maximum attachments per issue

JIRA allows multiple files to be attached to an issue in one operation. From the '**Attach File**' page, the user can toggle between multiple and single attachment screens by selecting the '**Attach multiple**' files link. The attachment form will retain the multiple/single attachment preference for that specific user for the duration of the user's session or until manually changed.

The number of attachment 'boxes' to be displayed on the multiple attachment screen is set to 3 by default. To change this, edit `jira.attachment.number` in the `jira-application.properties` file, then restart JIRA.

1.13.7. Configuring Image Attachment Thumbnails

'Thumbnails' allow an image to be previewed in miniature, without having to download the original full-size image. JIRA can automatically create thumbnails for file attachments of the following types:

- GIF
- JPEG
- PNG

Once thumbnail functionality is enabled, thumbnails are displayed to users in the '**Image Attachments**' section when viewing an issue. (All other file attachments are displayed in the **File Attachments** section.) You can also configure the Issue Navigator column layout to display the thumbnails in an `Images` column.

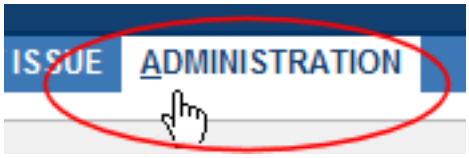
All thumbnail images are stored in JPEG format in the `attachments` directory (see '*Configuring File Attachments*'), together with the original attachments. The thumbnail images

are denoted by '`_thumb_`' in the filename.

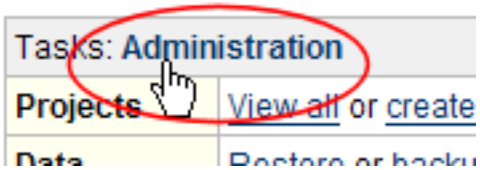
Note:
Thumbnail image generation requires the following:
1. The system must have X11 support. This web page details the minimum set of libraries needed to use JDK 1.4.2 under RedHat Linux 9.0.
2. The following java system property must be set: `-Djava.awt.headless=true`.

1.13.7.1. Enabling thumbnails

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Under the '**Global Settings**' sub-menu in the left-hand navigation column, click the '**Attachments**' link. This will display the '**Attachment Settings**' page, which states whether attachments are currently on or off.
   - **Note:** attachments must be enabled in order to enable thumbnails. Attachments can only be enabled by people who have the '**JIRA System Administrators**' global permission.
4. Click the '**Edit Configuration**' link.
5. In the '**Enable Thumbnails**' field, select '**ON**'.
6. Click the '**Update**' button.

1.13.7.2. Configuring thumbnail size

By default, thumbnails are 200 pixels wide and 200 pixels high. To change the dimensions of thumbnail images:

1. Stop JIRA.
2. Edit the following values found in the file `jira-application.properties`:
   - `jira.thumbnail.maxwidth`
   - `jira.thumbnail.maxheight`
3. Delete all existing thumbnail images within the `attachments` directory (denoted by `*_thumb_*` in the filename).
4. Restart JIRA. All thumbnails will be automatically recreated using the new dimensions.

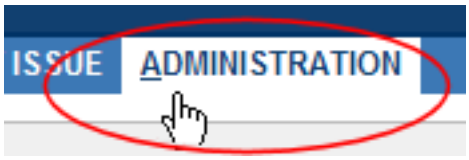
1.14. Server Administration

1.14.1. Search Indexing

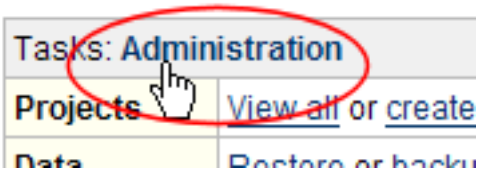
In order to provide fast searching, JIRA creates an index of the text entered into issue fields. This index is stored on the file system, and updated whenever issue text is added or modified. It is sometimes necessary to regenerate this index manually; for instance if issues have been manually entered into the database, or the index has been lost or corrupted.

1.14.1.1. Indexing Administration

1. Log in as a user with the '**JIRA Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Under the '**System**' sub-menu on the left, click the '**Indexing**' link.
4. This page allows you to:
   - reindex your data
   - optimise your index
   - move your index to another directory*. Note that your index directory should be a fast, local disk on the machine on which your JIRA instance is installed. Also note that your index directory should be given appropriate security, as described in Security Overview.

   *Note that only people with the '**JIRA System Administrators**' global permission can move the index.

Note:

Whenever you reindex data, JIRA will clear any existing indexes and re-index all the current data from scratch. This may take a few minutes, depending on how many issues you have, and users will be unable to access JIRA during this time.

1.14.2. Backing up data

This page describes how to back up your JIRA data, and establish processes for maintaining continual backups. Backing up your JIRA data is the first step in upgrading your server to a new JIRA revision, or splitting your JIRA instance across multiple servers. See also *Restoring JIRA data* and *Restoring a Project from Backup*.

Creating a complete backup of JIRA consists of two stages:

1. Backing up the data in the database
2. If attachments are enabled, backing up the attachments directory

1.14.2.1. 1. Backing up database contents

There are two possibilities: native database-specific tools, or JIRA's XML backup utility.

For production use, it is **strongly recommended to use native database-specific tools**. The XML backup is not guaranteed to be consistent, as the database is not locked during the backup process.

Using native database tools

All serious databases come with tools to back up and restore databases (the 'MS' in RDBMS). We

strongly recommend these tools in preference to the XML backup option described below, as they:

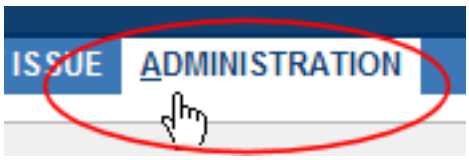
- ensure integrity of the database by taking the backup at a single point in time
- are much faster and less resource-intensive than JIRA's XML backup.
- integrate with existing backup strategies (e.g. allowing one backup run for all database-using apps).
- may allow for incremental (as opposed to 'full') backups, saving disk space.
- avoid character encoding and format issues relating to JIRA's use of XML as a backup format.

See the documentation for your database on how to set up periodic backups. This typically involves a cron job or Windows scheduled task invoking a command-line tool like mysqldump or pg_dump,

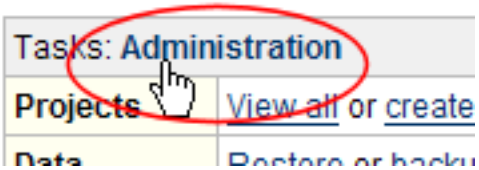
Using JIRA's XML backup utility

To perform a once-off backup, e.g. before an upgrade, follow the steps below. (Note that you can also configure scheduled XML backups, as described in *Automating JIRA Backups*.)

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

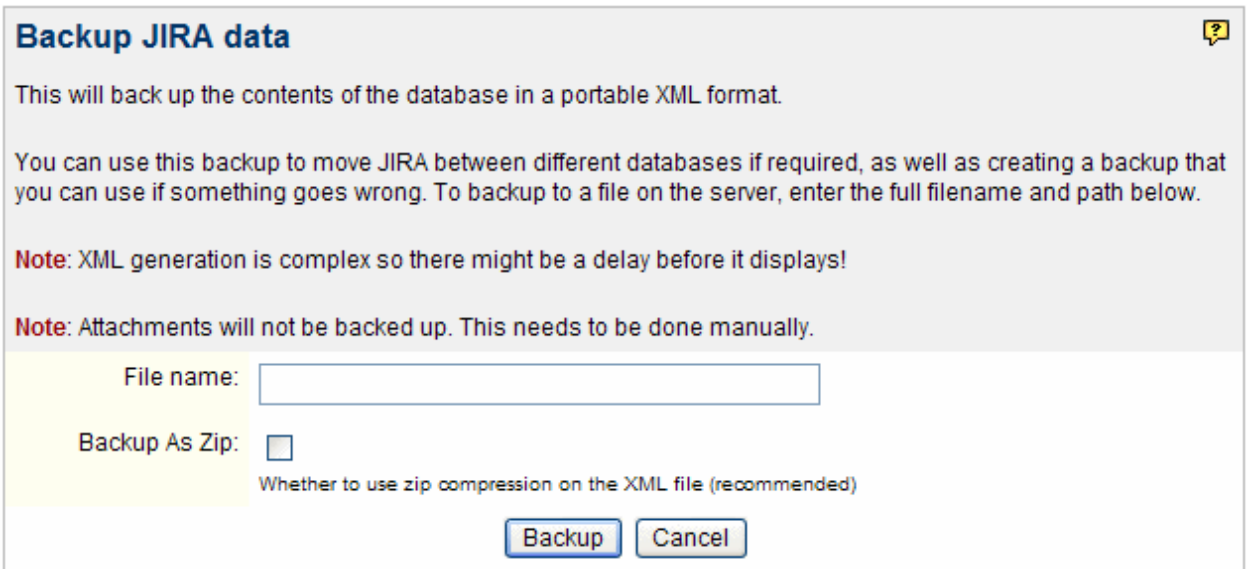


Link to Administration section



Link to Administration section

3. Under the '**Import & Export**' sub-menu in the left-hand navigation column, click the '**Backup Data to XML**' link. This will display the '**Backup JIRA data** page:



Backup data from XML

4. In '**File name**' field, type the full path, including filename, of the location to which you want JIRA to write the backup file. **Note: Ensure that JIRA has rights to write to this location**.
5. *(Optional but recommended)* Select the '**Backup as Zip**' checkbox.

6. Click the '**Backup**' button, and wait while your JIRA data is backed up.
7. When the backup is complete, a message will be displayed, confirming that JIRA has written the data to the file you specified.

1.14.2.2. 2. Backing up attachments

If you have attachments enabled you also need to create a backup of the attachments directory, as the attachments do not get stored in the database.

To back up attachments, you need to create a snapshot of the attachment directory (including all files and subdirectories). Note that the directory structure under the attachments directory **must** be preserved in the snapshot.

Creating this snapshot is an operating system-specific task, e.g.:

- On MS Windows, a batch script copying the directory can be written and scheduled periodically (Programs > Accessories > System Tools > Scheduled Tasks). There are also various utilities available to simplify this (eg. http://www.picozip.com).
- On Linux/Solaris, it is best to write a small shell script, placed in `/etc/cron.daily`, backing up files to a directory like `/var/backup/jira`. It is best to copy an existing script in `/etc/cron.daily` to ensure local conventions (file locations, lockfiles, permissions) are adhered to.

1.14.3. Automating JIRA backups

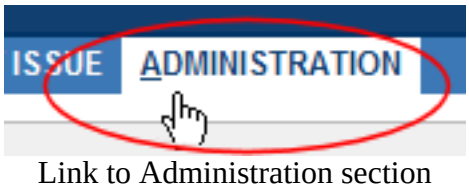
JIRA can be configured to automatically create an XML backup of JIRA data on a routine basis.

Note:
The XML backup includes all data in the database. However, it **does not include** attachments, which are on the filesystem.

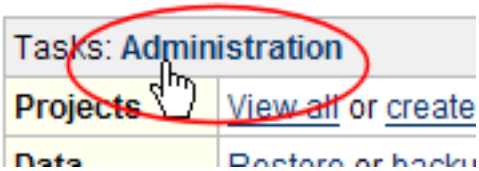
Note:
For production use, it is **strongly recommended** to use native database-specific tools instead of the XML backup service. The XML backup is not guaranteed to be consistent, as the database is not locked during the backup process.

When JIRA is installed, the first step in the Setup Wizard prompts you for a backup path, and if entered, JIRA will automatically generate XML backups (as ZIP files) every 12 hours. If you did not specify this path, follow the steps below to enable automated backups. (Note that you can also perform XML backups manually — see *Backing up data*.)

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



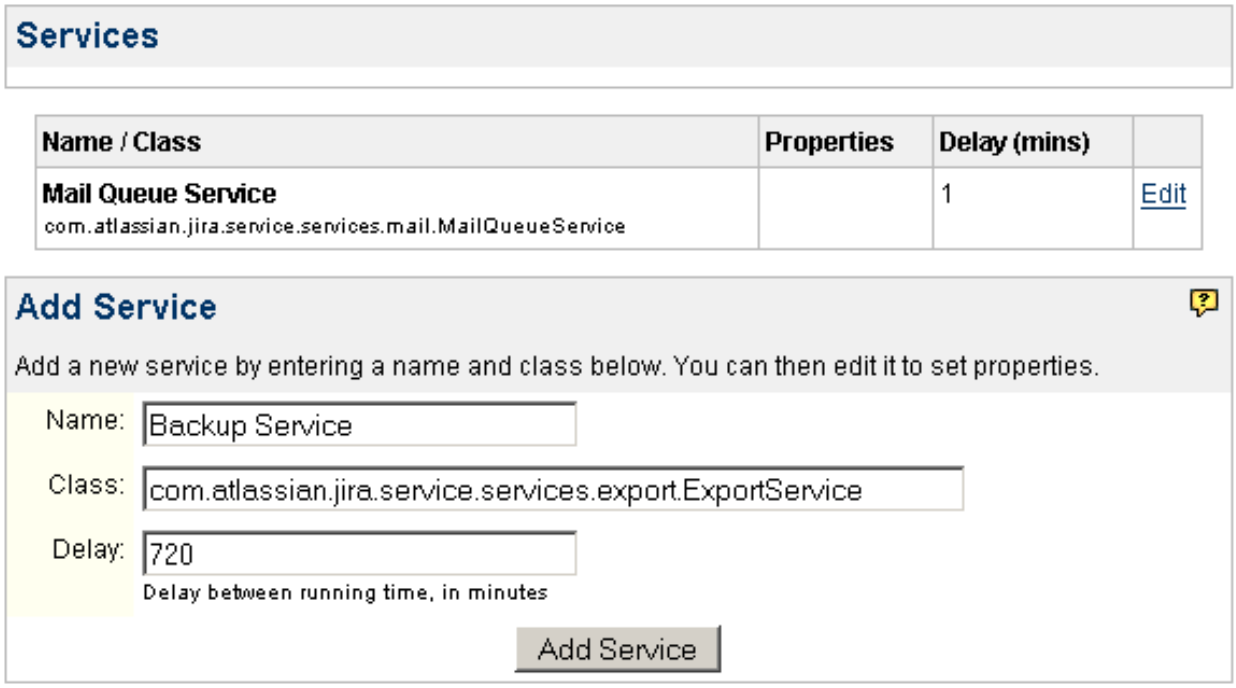
Link to Administration section



Link to Administration section

3. Under the '**System**' sub-menu in the left-hand navigation column, click the '**Services**' link.

4. This will display the '**Services**' page. It lists the current services running on this system. By default there should be one service running: '**Mail Queue Service**'. You cannot delete this service.
5. Fill in the '**Add Service**' form as follows:

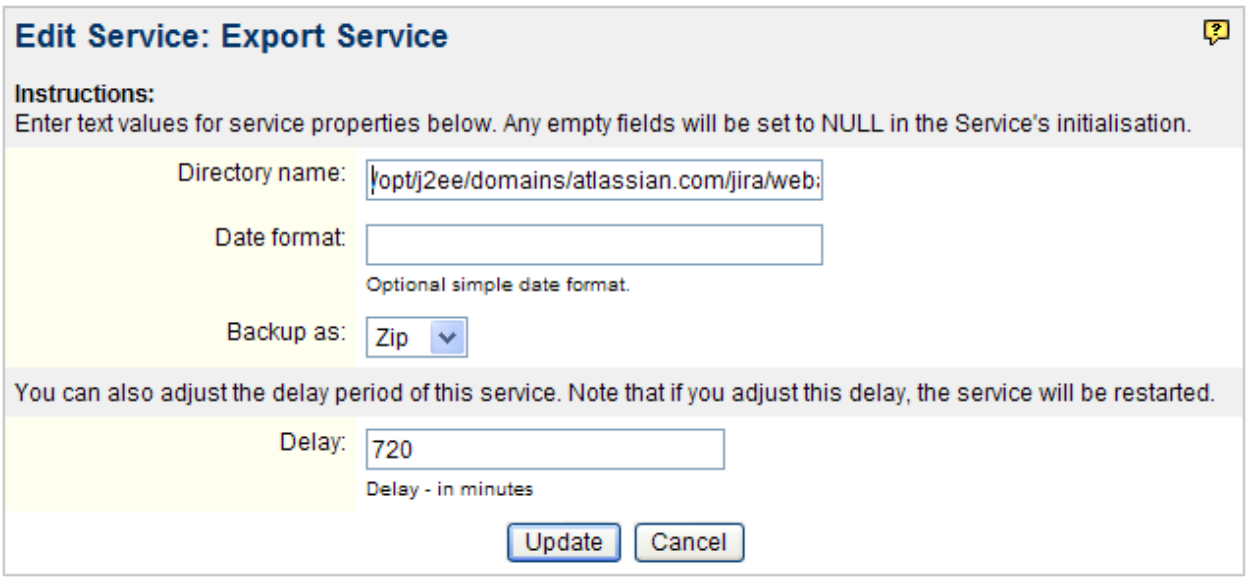


Add Automated Backup Service

- For '**Name**', enter a descriptive name, e.g. **Backup Service**
- For **Class**, enter **com.atlassian.jira.service.services.export.ExportService**
- For '**Delay**', enter the number of minutes between backups. A good default for this would be 720 minutes (12 hours).

Click the '**Add Service**' buttont.

6. This will display the '**Edit Service**' screen. Fill in the following fields:



Edit Automated Backup Service

- For '**Directory Name**', type the full path of the directory to which JIRA will write backup files. **Note: Ensure that JIRA has rights to write to this location**.
- For '**Date format**', specify the format which JIRA will use to name the individual backup

files. This format can be anything that SimpleDateFormat can parse. A good default is 'yyyy-MMM-dd-HHmm', which would generate files named like this: '2007-Mar-05-1322'.
    - For '**Backup as:**', either:
        - Select '**XML**' to have JIRA store your data as an XML file; or
        - Select **Zip**' to have JIRA to compress your backup and store it as a ZIP file.
7. Your backup service is now configured. XML backups will be performed according to the schedule you specified in the **Delay** field (above).
    - For every successful backup, an XML (or ZIP) file will be saved in the backup directory that you specified in the **Directory Name** field (above).
    - If a scheduled backup fails for any reason, the XML (or ZIP) file will be saved into the 'corrupted' directory, which is directly under your nominated backup directory. (NB. JIRA will created the 'corrupted' directory if required — you don't need to create it.) Additionally, a file explaining the reason for the failure will be written to the 'corrupted' directory. This file will have the same name as the backup file, but with the extension '.failure.txt'.

1.14.4. Restoring data

This page describes how to restore JIRA data from a backup. This is the second step in either upgrading your server to a new JIRA revision, or splitting your JIRA instance across multiple servers.

If you wish restore a single project from your backup into an existing JIRA instance, refer to these instructions on restoring a project from backup instead.

Note:

When restoring data, **all data in the existing JIRA database is deleted**, including all user accounts. Before you begin, make sure you have the password to a login in the backup file that has the 'JIRA System Administrator' global permission.

Restoring JIRA from backup is a three stage process:

1. Optionally, disable email sending/receiving.
2. Restore data from XML to the database.
3. Optionally, if attachments were backed up, restore the attachments to the attachments directory.

1.14.4.1. 1. Disabling email sending/receiving

If you are restoring production data into a *test* JIRA instance for experimentation purposes, you probably want to disable JIRA's email interaction features before you begin:

- **Disable email notifications:**
  if JIRA is configured to send emails about changes to issues, and you want to make test modifications to the copy, you should start JIRA with the `-Datlassian.mail.senddisabled=true` flag.
- **Disable POP/IMAP email polling:**
  if JIRA is configured to poll a mailbox (to create issues from mails), you will likely want to disable this polling on your test installation. This is done by setting the `-Datlassian.mail.fetchdisabled=true` flag.

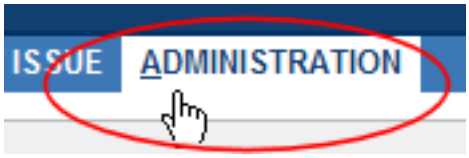
Exactly how to set these flags is dependent on your particular application server, but for JIRA Standalone (i.e. Tomcat), it is done by setting the JAVA_OPTS environment variable before starting JIRA:

```
set JAVA_OPTS="-Datlassian.mail.senddisabled=true -Datlassian.mail.fetchdisabled=true"
cd bin
startup.bat
```

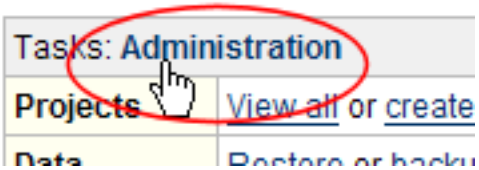
1.14.4.2. 2. Restoring XML data

Note: these instructions assume that you are restoring an XML backup. If you used native database tools to create your backup, the restore process will also be tool-specific so these instructions do not apply to you.

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:

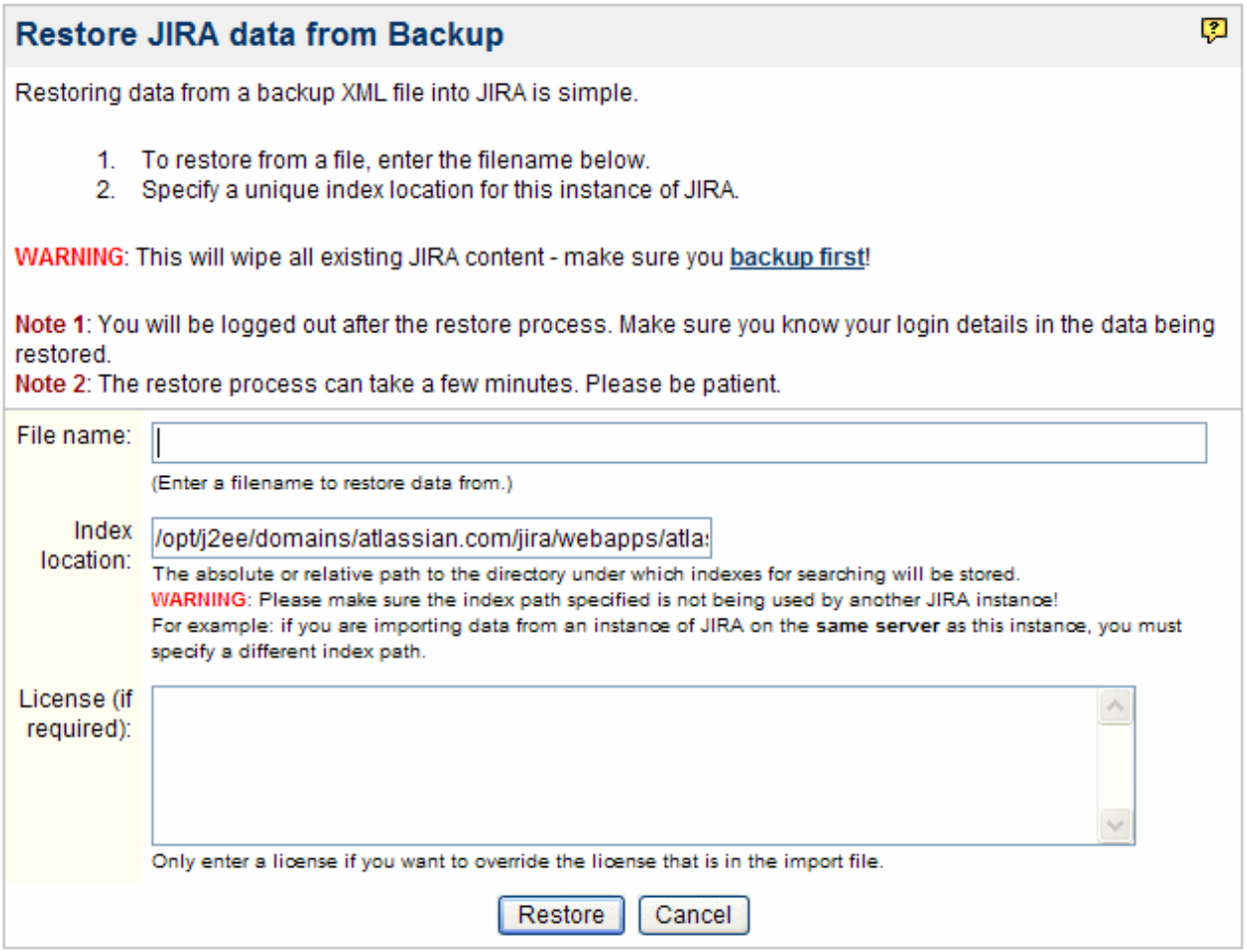


Link to Administration section



Link to Administration section

3. Under the '**Import & Export**' sub-menu in the left-hand navigation column, click the '**Restore Data from XML**' link. This will display the '**Restore data from Backup** page:



Restore data from XML

4. In the '**File name**' field, fill in the full path to the ZIP or XML backup file generated by JIRA.
5. Check or (if in the Setup Wizard) fill in the '**Index location**' path. Ensure this directory contains *only* the indexes, as its contents may be deleted by the restore process.
6. Click the '**Restore**' button, and wait while your JIRA data is restored.
7. JIRA will come back, informing you that you have been logged out. This is done because all the users from the previous JIRA instance have been deleted and replaced with users from the JIRA export file.
8. Log in, and if necessary, correct the search index path. Note: if you are running more than one JIRA instance on the same server, ensure that each instance uses a different index path.

1.14.4.3. 3. Restoring attachments

If you created a backup of the attachments directory, you will need to restore the backup into a directory where JIRA can access it. The process of restoring the attachments backup depends on the way it was created. Usually you can use the same tool to restore the backup as the one that was used to create it (see *Backing up attachments*). **Note: When you restore your attachments, ensure that the file permissions are correct.**

If you are restoring the attachments into a different location (i.e. a different directory path) from where they were previously located (e.g. this will be the case when moving servers), please follow the instructions provided in *Configuring attachments* to change the location of the attachments directory so that JIRA can find the restored attachments.

1.14.5. Restoring a Project from Backup

JIRA's Project Import tool allows you to restore a single project from a backup file into your JIRA instance. This feature is particularly useful if you do not wish to overwrite the existing projects or configuration of your JIRA instance by importing the entire backup. Your backup file must have been created using JIRA's backup tool. You cannot import a project from a backup using your native database tools.

Please note, if you wish to restore a project from a backup file into a **new empty JIRA instance**, we highly recommend that you **do not use the Project Import tool**. Restoring the entire backup file into the new instance and then deleting unwanted projects is much simpler in this scenario, as you will retain the configuration settings from your backup. Instructions on moving a project to a new instance are available on the splitting a JIRA instance page. Projects can be deleted via the 'Projects' page in JIRA, which is accessed from the '*Administration' menu.

1.14.5.1. Before you begin

Restoring a project from a backup is not a trivial task. You may be required to change the configuration of your target JIRA instance to accommodate the project import. Additionally, the Project Import data mapping can be resource intensive on your hardware and may take a long time to complete, if you are importing a large project. Note, the Project Import tool will lock out your instance of JIRA during the actual data import (not during the validations), so please ensure that your instance does not need to be accessible during this time.

Note:

We strongly recommend that you perform a full backup of your target JIRA instance before attempting to restore a project into it.

Project Import Restrictions

The Project Import tool will only import a project between identical instances of JIRA. That is;

- The version of JIRA in which your backup was created must be identical to the version of your target JIRA instance, e.g. if your backup file was created in JIRA 4.0, then your target instance of JIRA must be version 4.0.
- The edition of JIRA in which your backup was created must be identical to the edition of your target JIRA instance, e.g. if your backup file was created in JIRA Standard Edition , then your target instance of JIRA must be JIRA Standard Edition.
- If your instance of JIRA had a custom fields plugin (e.g. JIRA Toolkit) installed when the backup file was created and the custom field was used in your project, then your target instance of JIRA must have the same version of the plugin installed for the Project Import tool to automatically work.

If any of these restrictions apply and you still wish to restore your project from backup, you will need to create a compatible backup file before importing your project by following the appropriate instructions below:

JIRA versions do not match

- If your backup file was created in an earlier version of JIRA than your target instance of JIRA:
  1. Set up a test JIRA instance, which is the same version as your target instance of JIRA. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance.
  2. Import the backup file into a test JIRA instance.
  3. Create a new backup file from your test JIRA instance.
- If your backup file is from a later version of JIRA than your target instance of JIRA:
  1. Upgrade the version of your target instance of JIRA to match the version of JIRA in which the backup was created.

JIRA editions do not match

- If your backup file is from a lower edition of JIRA than your target instance of JIRA:
  1. Set up a test JIRA instance, which is the same edition as your target instance of JIRA. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance.
  2. Import the backup file into a test JIRA instance.
  3. Create a new backup file from your test JIRA instance.
- If your backup file is from a higher edition of JIRA than your target instance of JIRA, you may wish to consider upgrading the edition of your target JIRA instance. Otherwise:
  1. Import the backup file into a test JIRA instance. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance, as the import will overwrite all data in the database.
  2. In your test JIRA instance, downgrade the edition of JIRA to match the edition of your target instance of JIRA. The downgrading JIRA page contains useful information that can help you with this process.
  3. Create a new backup file from your test JIRA instance.

Custom fields plugin versions do not match

- If the custom fields plugin from your backup is an earlier version than the custom fields plugin in your target instance of JIRA:
  1. Import the backup file into a test JIRA instance. Make sure that the test JIRA instance uses a separate database and index from your target JIRA instance, as the import will overwrite all data in the database.
  2. In your test JIRA instance, upgrade your version of your custom fields plugin to match the version of the plugin in your target instance of JIRA.
  3. Create a new backup file from your test JIRA instance.
- If the custom fields plugin from your backup is a later version than the custom fields plugin in

your target instance of JIRA:

1. Upgrade the custom fields plugin version of your target instance of JIRA to match the version of JIRA in which the backup was created.

1.14.5.2. Restoring your project

The Project Import tool will attempt to map the data in your backup file into your target JIRA instance. If the project you are restoring does not exist in your target JIRA instance, it will create and populate the project with data from your backup. If the project already exists and is empty, it will attempt to populate the data from your backup into the project.

Note:

Why should I create an empty project in my target JIRA instance?
It is important to note that the primary task of the Project Import tool is to restore the data from your backup project into your target JIRA instance. While the Project Import tool can create a project if one does not exist in your target JIRA instance, it does not recreate any configuration settings that affect the data (e.g. screen schemes). If you wish to retain any configuration settings from your original project, we recommend that you create an empty project in your target instance with the necessary configuration settings before importing the data from your backup project.

You may wish to carry out the following setup tasks to ensure that your target JIRA instance is prepared to receive a project import beforehand. This can improve the time taken to validate the data mappings to your target JIRA instance.

If you are confident that your JIRA instance is set up appropriately, you can skip straight to the Project Import tool instructions. If there are any problems mapping the data from your backup file to your target JIRA instance, the Project Import tool will present validation errors for you to address.

Preparing your target JIRA instance

The Project Import tool does not automatically add missing project entities (e.g. user groups, issue priorities, custom field types) or fix incorrect associations (e.g. issue types in workflow schemes), so some manual work is required to set up your target JIRA instance so that your project can be restored. If the Project Import wizard cannot find a valid target location for any of the backup project data, it will not be able to restore the project. The instructions below describe the setup activities that address the most common data mapping problems that occur when restoring a project from a backup.

We recommend that you perform as much of the configuration of your target JIRA instance as possible, prior to starting the project import. However, if you do not have the information available to complete these setup activities beforehand, the Project Import wizard will inform you of any problems that need your attention. Alternatively, you can import the backup file into a test JIRA instance to check the configuration.

1. Setting up the project

If you have a project in your target JIRA instance that you wish to restore data into, you will need to ensure that the project is empty, i.e.

- no issues — *read the Quick Search page to find out how to find all issues in a project.*
- no components — *read the Component Management page to find out how to view a summary of a project's components.*
- no versions — *read the Version Management page to find out how to view a summary of a project's versions.*

2. Setting up users and groups

The following types of users are considered mandatory for a project to be imported:

- reporter, assignee, component lead or project lead.

The following users are considered to be optional for a project to be imported:

- comment author/editor, work log author/editor, a user in a custom field (user picker), voter, watcher, change group author (i.e. someone who has changed an issue), attachment author, user in a project role.

The Project Import will attempt to create missing users if they are associated with the project. However, if the Project Import tool cannot create missing mandatory users in your target JIRA instance, then you will not be permitted to import the project. This may occur in the following situations:

- you have External User Management enabled in your target JIRA instance — you will need to disable External User Management or create the missing users manually in your external user repository before commencing the import.
- Atlassian's Crowd was connected to your JIRA instance when the backup was made (hence, the backup file will only contain minimal reference data for each user which is insufficient to create users in JIRA) — you will need to create any missing mandatory users manually. Alternatively, you may wish to connect the current JIRA to the same external user management system as the original, if that is possible. This check does not apply if you connected an LDAP (not using Crowd) to your JIRA instance when the backup was made, as connecting an LDAP to JIRA requires the creation of the users in JIRA (hence, full user data will be in the backup file).

Note:

Please note that if you do not have enough information about the users in your backup file, the Project Import wizard will provide a link to a table of the missing users on a new page as well as a link to an XML file containing the missing users (on the new page). The table of users will display a maximum of 100 users, but the XML file will always be available.

3. Setting up custom fields

As described previously, the versions of your custom field plugins must match between your backup and your target instance of JIRA for your project to be imported. You need to ensure that you have set up your custom fields correctly in your target JIRA instance, as follows:

- **Custom Field Type** — If you do not have a particular custom field type (e.g. cascading select) installed on your target JIRA, then all custom field data in your backup project that uses that custom field type will not be restored. However, your project can still be restored.
  For example, say you have a custom field, 'Title', which is a 'Cascading Select' field type and was used in your backup project (i.e. there is saved data for this field). If you do not have the 'Cascading Select' custom field type installed on your target JIRA, then all data for custom field 'Title' (and all other cascading select custom fields) will not be restored.
- **Custom Field Configuration** — If you do have a particular custom field type (e.g. multi select) installed on your target JIRA, then you must configure all of the custom fields (of that custom type) in your target JIRA to match the equivalent custom fields in your backup project. Additionally, if your custom field has selectable options, then any options used (i.e. there is saved data for these options) in your backup project must exist as options for the custom field in your target JIRA.
  For example, say you have a custom multi select field named, 'Preferred Contact Method', in your backup project with options, 'Phone', 'Email', 'Fax'. Only the 'Phone' and 'Email' were actually used in your backup project. In this scenario, you need to set up your target JIRA instance as follows:
  - There must be a field named, 'Preferred Contact Method', in your target JIRA instance.
  - 'Preferred Contact Method' must be a multi select custom field type.
  - 'Preferred Contact Method' must have the options, 'Phone' and 'Email' at a minimum, since

they were used in your backup project. Please note, 'Preferred Contact Method' in your target JIRA could also have additional options like 'Fax', 'Post', 'Mobile', etc, if you choose.

If you have not configured your existing custom field correctly, you will not be permitted to import your backup project until you correct the configuration errors in your target JIRA.

You may wish to refer to the custom fields documentation for more information on the custom field types and custom field configuration.

- **Compatibility with the Project Import tool** — Custom fields also need to be compatible with the Project Import tool for the custom field data to be imported. Custom fields created prior to JIRA v4.0 cannot be imported by the Project Import tool. The custom field developer will need to make additional code changes to allow the Project Import tool to restore the custom field data. If any of the custom fields used in your backup file are not compatible with the Project Import tool, the Project Import wizard will warn you and the related custom field data will not be imported. All the target JIRA system custom fields and the custom fields included in JIRA plugins supported by Atlassian (e.g. JIRA Toolkit, Charting Plugin, Labels Plugin, Perforce Plugin) are compatible with the Project Import tool.

4. Setting up workflows, issues and groups/roles

In addition to custom fields, you need to correctly configure the project workflow, issue attributes (e.g. issue types) and groups/roles in your target JIRA instance for your project to be restored successfully. Please ensure that you have reviewed the constraints on each of the following:

Workflows and Workflow Schemes (*Professional and Enterprise editions only*):

- The project import process does not import workflows or workflow schemes. If you wish to retain a customised workflow from your backup, you will need to create a new workflow in your target JIRA instance and manually edit the new workflow (e.g. create steps and transitions) to reflect your old workflow (note, the default JIRA workflow is not editable). You will then have to add this workflow to a workflow scheme to activate it. Read more about creating and editing workflows in the JIRA Workflow and Activating Workflows documents. Please note that you may be required to create and edit a new workflow and workflow scheme to satisfy constraints on workflow entities from your backup, as described in the sections below, even if you do not wish to recreate the exact same workflow.

Note:

Do not use the JIRA functionality for exporting and importing workflow XML definitions, to copy your backup workflow to your target JIRA instance. The workflow import/export tools do not include workflow screens in the process. Hence, you will be required to manually edit the workflow definitions post-import to match up new screens to the workflow, which is more work that it is worth.

Issue Types:

- If an issue type has been used in your backup project (i.e. there are issues of this issue type), you must set up the same issue type in your target JIRA project. If you are using JIRA Professional or JIRA Enterprise, you may wish to set up Issue Types for the project instead of globally.
- Workflow schemes — If you have associated an issue type with a particular workflow scheme in your backup project, you must ensure that the same association exists in your target JIRA. See the above section on '**Workflow and Workflow Schemes**' for further information on how to set up a workflow in your target JIRA instance.
- Custom field configuration schemes — custom field configuration schemes can be used to apply a custom field configuration to specific issue types. If you have configured a custom field differently for different issue types in your backup project, you may wish to set up a custom field configuration scheme to apply the same custom field configuration to the same

issue types in your target JIRA instance. This will help ensure that you do not have a custom field for an issue type that is configured incorrectly (e.g. missing an option, if it has multiple selectable options), as described in the 'Setting up custom fields' section above.

Statuses:

- If an issue status has been used in your backup project (i.e. there are issues of this status), you must set up the same status in your target JIRA project.
- Workflow schemes — If you have linked a status into a particular workflow scheme in your backup project, you must ensure that the same association exists in your target JIRA. See the above section on 'Workflow and Workflow Schemes' for further information on how to set up a workflow in your target JIRA instance.

Security Levels (Enterprise Edition only):

- If an issue security level has been used in your backup project (i.e. there are issues with this security level), it must be set up in your target instance of JIRA. If you did not create an existing empty project, we recommend that you do so and set up the appropriate security levels for the project (via an issue security scheme).
- Issue Security schemes — Not applicable. It does not matter which users, groups or project roles are assigned to which security levels, as long as the appropriate security levels exist (please see the constraints on security levels in the 'Setting up entities and types' section).

Priority:

- If an issue priority has been used in your backup project (i.e. there are issues with this priority), it must be set up in your target instance of JIRA.

Resolution:

- If an issue resolution has been used in your backup project (i.e. there are issues with this resolution), it must be set up in your target instance of JIRA.

Issue Link Type:

- If an issue link type has been used in your backup project (i.e. there are issues associated by this link type), it must be set up in your target instance of JIRA.

Project Role:

- If a project role has been used in your backup project (i.e. there are users/groups assigned to this project role), it must be set up in your target instance of JIRA.
  (Note: The Project Import tool will copy across the project role membership from your backup project to your target JIRA instance, if you choose. See the Project Import section for further details.)

Group:

- If a user group has been used in your backup project (i.e. there are users in this group), it must be set up in your target instance of JIRA.

Note:

A note about schemes
The project import process does not directly affect schemes, although entities and types associated with schemes may be affected as described above. Please note that the following schemes are not affected at all by the project import:
* Permission schemes — Not applicable. Permissions schemes do not need to match between the backup and target instance of JIRA.
* Notification schemes — Not applicable. Notification schemes do not need to match between the backup and target instance of JIRA.
* Screen schemes — Not applicable. Screen schemes do not need to match between the backup and target instance of JIRA.
* Issue type screen schemes (Enterprise Edition only) — Not applicable. Issue type screen schemes do not need to match between the backup and target instance of JIRA.
* Field Configuration schemes (Enterprise Edition only) — Not applicable. Please note that if a field was configured as optional in your backup project and is configured as a required field in your target JIRA instance, then the project will still be imported even if the field is empty. However, this field will be enforced as mandatory the next time a user edits an issue containing the field.

5. Setting up links

The Project Import tool will automatically create all issue links between issues within your backed up project. It will also try to create links between the backup project and another project, as long as the other project already exists in your target JIRA instance with the relevant issue keys. If the source/target of a link cannot be found (i.e. the entire project or the particular issue may be missing), the link will not be created although the project will still be imported.

Note that the Project Import tool will create issue links between projects in either direction (source to target, or target to source). This means that if you import two projects from the same backup file, the second project import will create all of the links between the two projects that were missing from the first project import.

Once you have completed as many of the setup tasks as you are able to, run the Project Import tool.

Project Import

Restoring your project is a four step process:

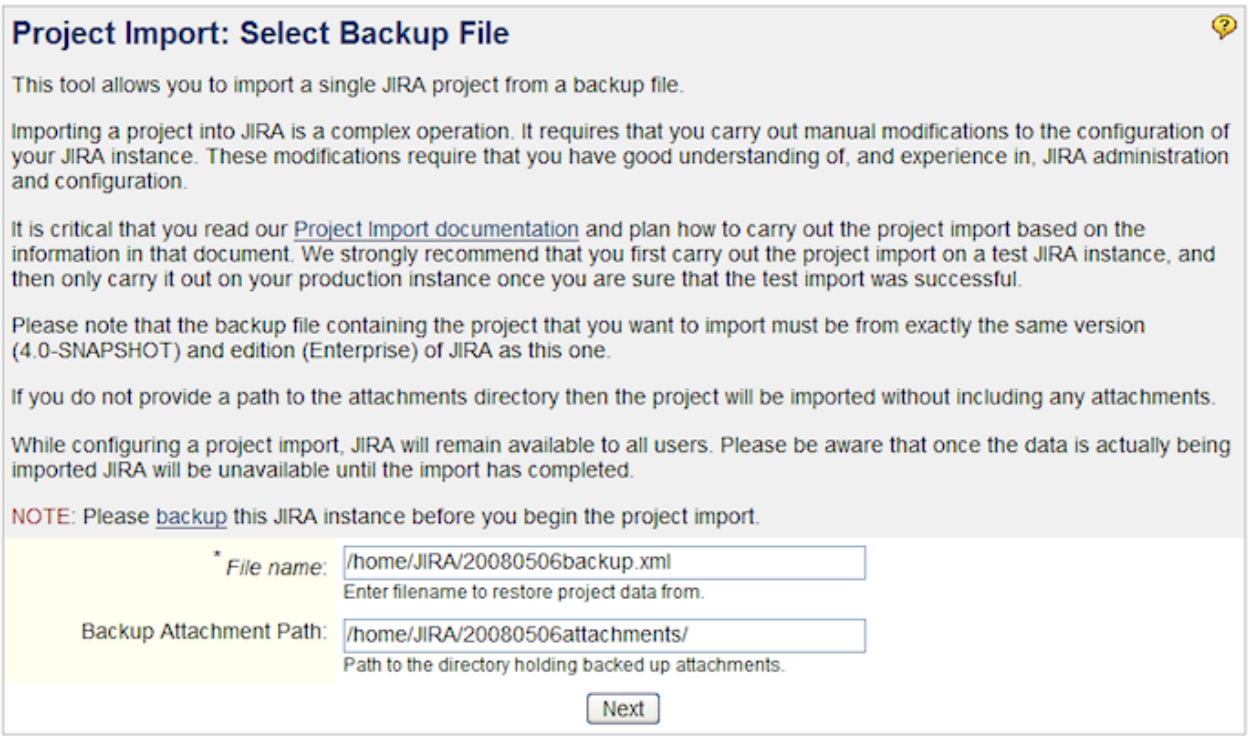
1. Specify the backup file
2. Select a project
3. Review data mapping validations
4. Verify the restored project

If you start the Project Import tool, we strongly recommend that you complete all steps of the wizard before performing any other activities in JIRA. Please be aware that it can take some time to validate the data mappings and then import the project.

You will most likely need to navigate away from the Project Import wizard to correct your JIRA configuration, as advised by validation errors in the wizard. If you have to navigate to other pages in JIRA to correct your JIRA configuration or for other activities, you should:

- *(recommended)* open a separate session of JIRA in a new browser window/tab. When you return to the Project Import wizard in the original browser window/tab, you can use the **'Refresh validations'** button on the validation screen to re-validate the data mappings; or,
- wait until the progress bar completes for the step you are currently in, before navigating elsewhere in JIRA. The state of the Project Import wizard will be saved until you log out of JIRA, your user session expires or you commence a different project import. You can resume your project import by returning to the Project Import page (via the main Administration menu) and selecting the "resume" link on the first page of the wizard.

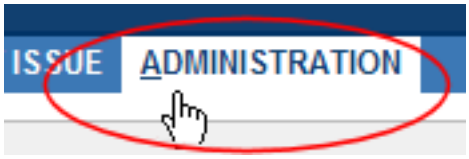
1. Specify the backup file



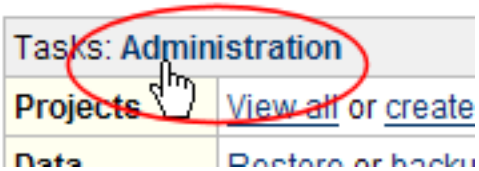
Specify a backup file

To start the Project Import tool,

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

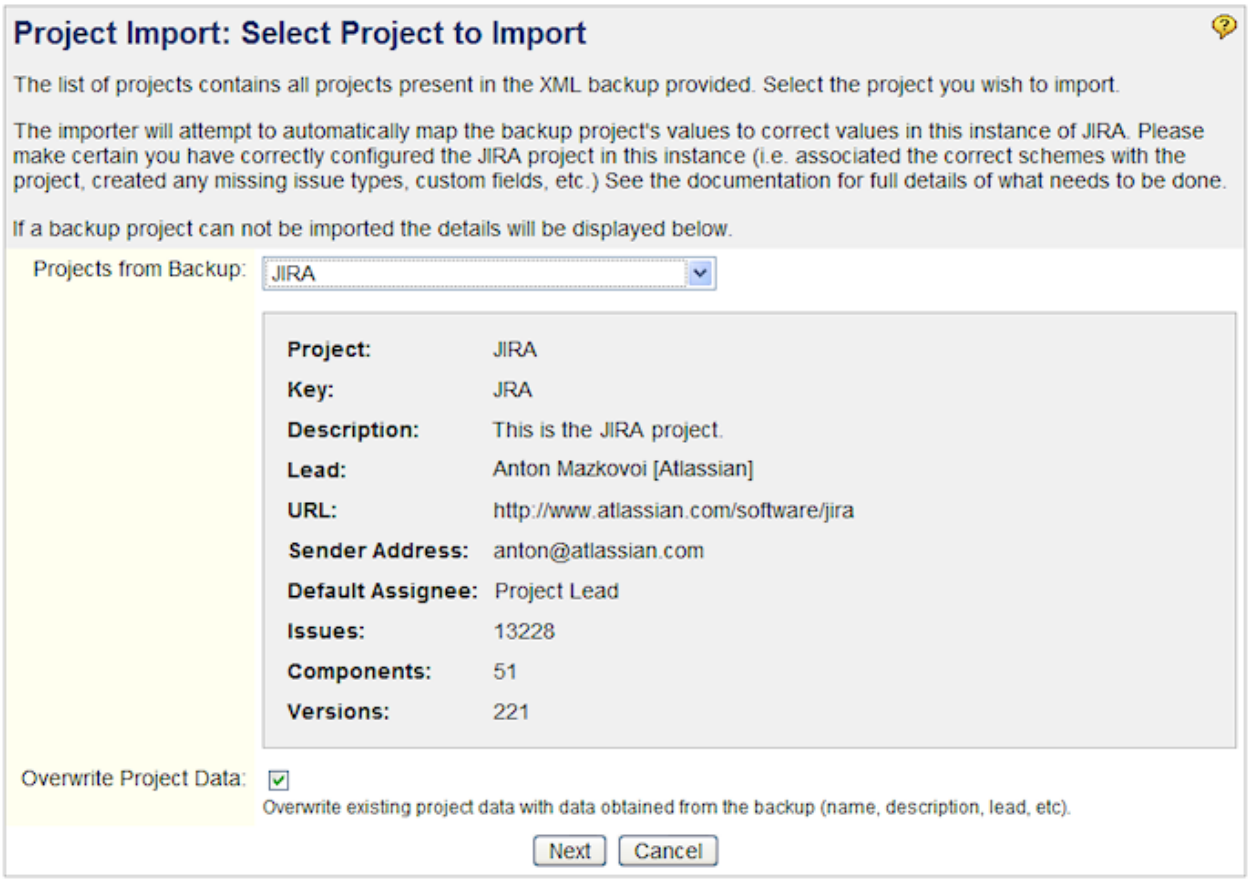


Link to Administration section

3. Click the **'Project Import'** link in the left hand menu. The first step of the Project Import wizard will display, ' Project Import: Select Backup File'.
4. Specify the path and name of your backup file in the 'File name' field. Your backup file must be an XML or ZIP file (as exported by JIRA).
5. Specify the path where you have backed up the attachments (add anchor to backup attachments section) for your project in the **'Backup Attachment Path'** field. Do not specify the attachment path for your target instance of JIRA as the backup attachment path, as the Project Import tool will overwrite attachments in that directory. Please also ensure that you have enabled file attachments in your target JIRA instance. You will not be allowed to proceed with the import if you have specified a backup attachment path and do not enable file attachments in your target JIRA instance.
   Note: You can choose to not specify a backup attachment path. If so, you will be able to restore your project from backup, however it will have no attachments associated with it. Please note,

you cannot restore your attachments separately if you do not restore them as part of the project import, as the database entries for the attachments will be missing.

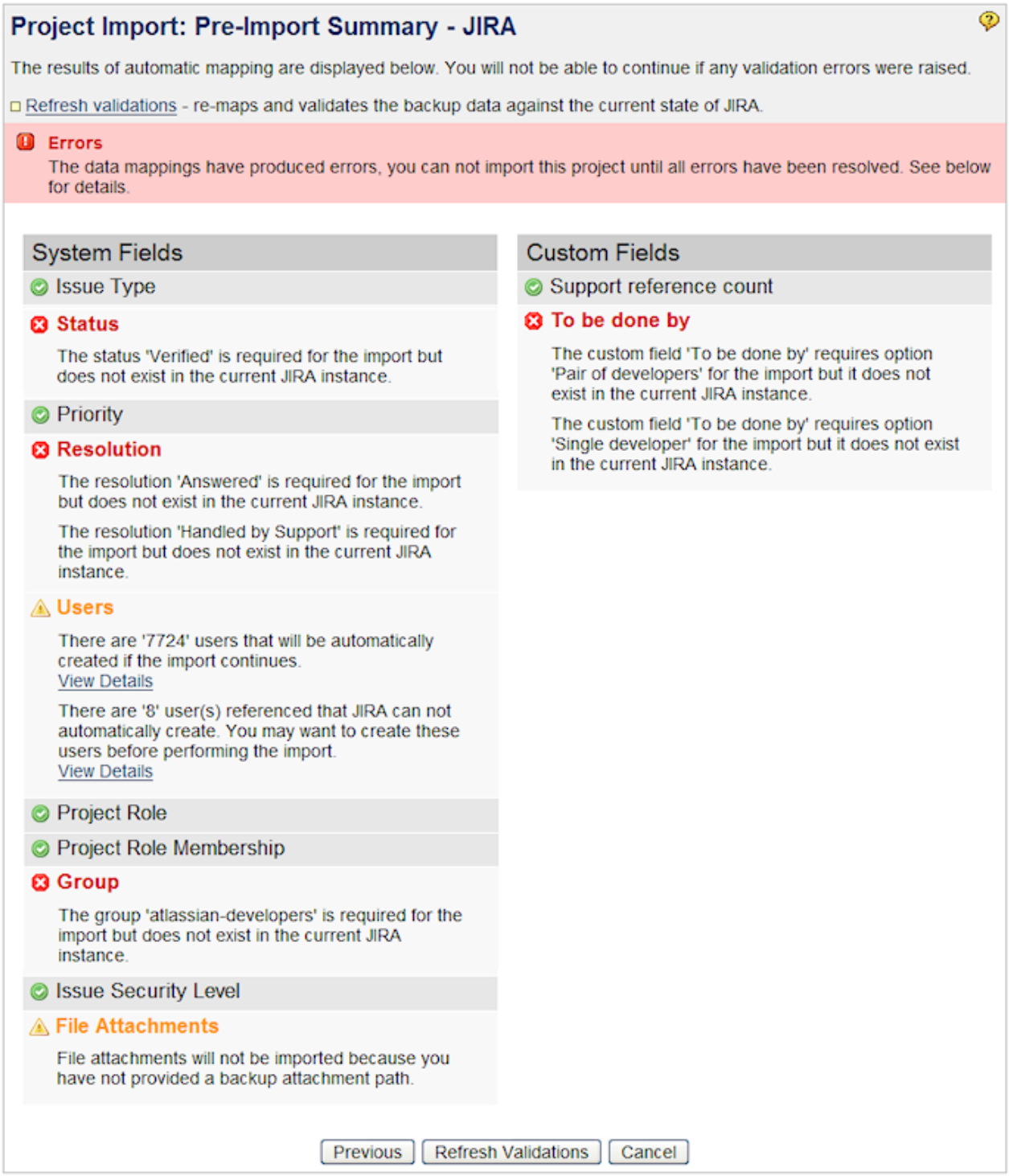
2. Select a project to restore



Project selection

1. Select a project to restore from the **'Projects from Backup'** dropdown. This dropdown will list all of the projects contained in your backup file.
2. If you have a valid project to restore from your backup, and your target JIRA instance has an existing empty project, then the '**Overwrite Project Details**' option will display. Select the '**Overwrite Project Details**' option if you want to overwrite the project details of the existing empty project with the project details from your backup. The project details are the Name, URL, Project Lead, Default Assignee and Description of the project, as well as any project role members set up on your project. If there is no existing empty project in your target instance of JIRA, this option will be checked and disabled as the Project Import will create the project with project details from your backup file.

3. Review data mapping validations



Data mapping validations

1. The Project Import wizard will attempt to validate the data mappings required to import your project from the backup file. You can review the validations at this step of the wizard and modify your target JIRA instance as required.
   - A tick symbol (

     Tick

) means that there are no problems with mapping these entities.
   - An exclamation mark symbol (

Exclamation Mark

) means that there are problems with the data mapping that you should review before importing the project, but the project can still be imported. For example, a missing optional user that cannot be created automatically by the Project Import tool.

- A cross symbol (

Cross

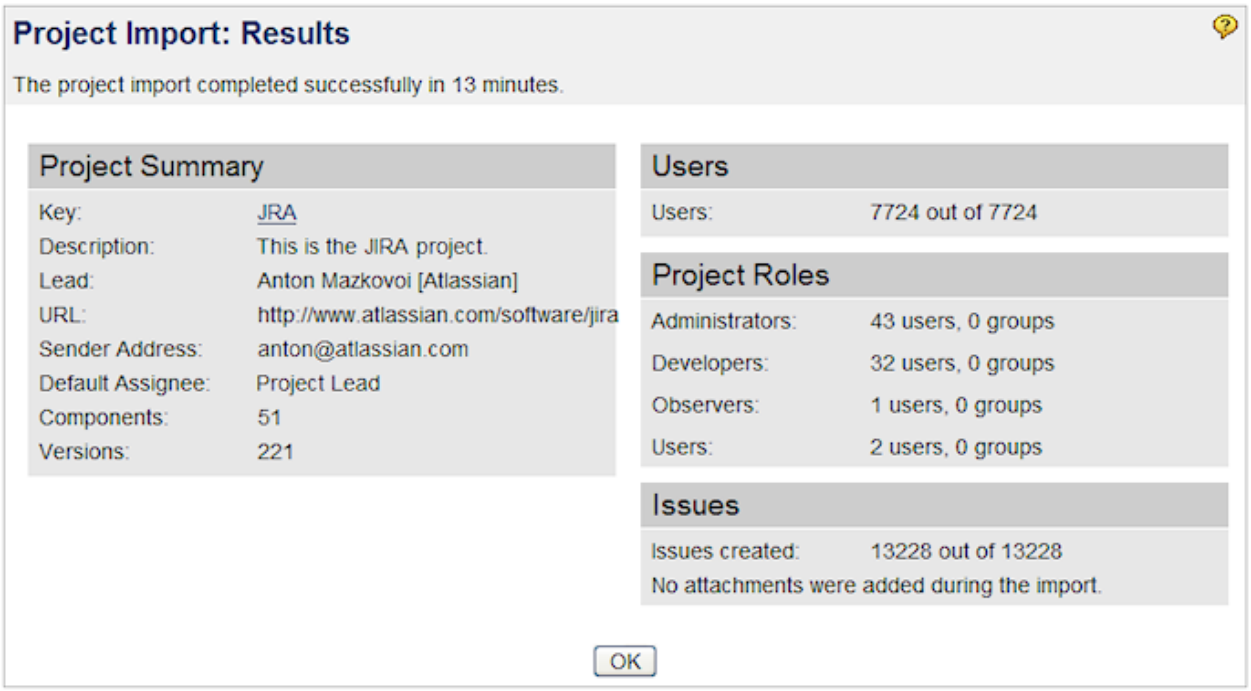
) means that there are problems with the data mapping that must be fixed before you can import the project. For example, an Issue Type that is used in the backed up project is missing in your target JIRA instance.

2. The '[Preparing your target JIRA instance]' section on this page lists the common data mapping errors.
3. Once you have resolved the data validation errors as required, click '**Import**' to commence the import of data from your backup file.

Note:

The Project Import tool will lock out your instance of JIRA during the actual data import (not during the validations), so please ensure that your instance does not need to be accessible during this time.

4. Verify the restored project



Project Import Results

1. Once the Project Tool has finished running, click '**OK**' to navigate to the restored project. You should verify that the issues, components and versions have been restored correctly. You should also check that any custom field data and links have been restored correctly.
2. Check that your attachments were correctly restored from your attachments backup directory.

What if something went wrong?

- If your project import **did not complete**, you can refer to the [JIRA log file]. The Project Import tool will log details of the operation to this file, including any unexpected errors and exceptions. e.g. database locked out, disk full... etc.

- If your project import completed but **did not restore your project as expected**, you may wish to attempt to fix the problem manually in your target JIRA instance. You may also wish to try deleting the project in your target JIRA instance and re-importing it from backup, paying special note to any warning validations (e.g. users that will not be added automatically).

If you cannot resolve the problem yourself, you can contact us for assistance. Please see the '**Need help**' section below for details.

Need Help?

Need further help? You can raise a support request in the JIRA project at https://support.atlassian.com for assistance from our support team. Please attach to the support case:

- the backup file you are trying to import projects from, and
- the following information from your target JIRA instance:
  - your log file
  - an XML backup of your target JIRA instance
  - a copy and paste of the **entire contents** of the **System Info** page (accessed via the **Administration** tab), so that we know the details of your JIRA configuration.

You can anonymise the XML backups, if your data contains sensitive information.

1.14.6. Database Integrity Checker

1.14.6.1. Overview

Searching for common data inconsistencies, the **Database Integrity Checker** attempts to ensure that all JIRA data is in a consistent state.

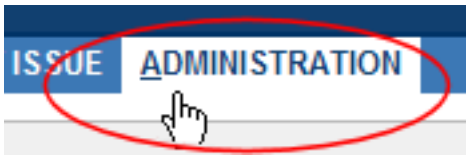
This is useful in a number of situations, e.g.:

- Before migrating a project to a new workflow
- An external program is modifying JIRA's database
- Troubleshooting a server crash

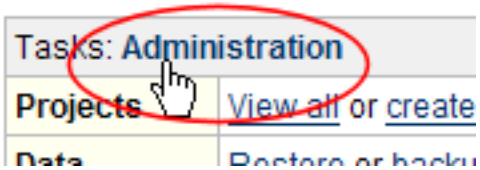
If an error is encountered, most of the integrity checks provide a 'repair' option which attempts to reset the data to a stable state.

1.14.6.2. Using the Integrity Checker

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



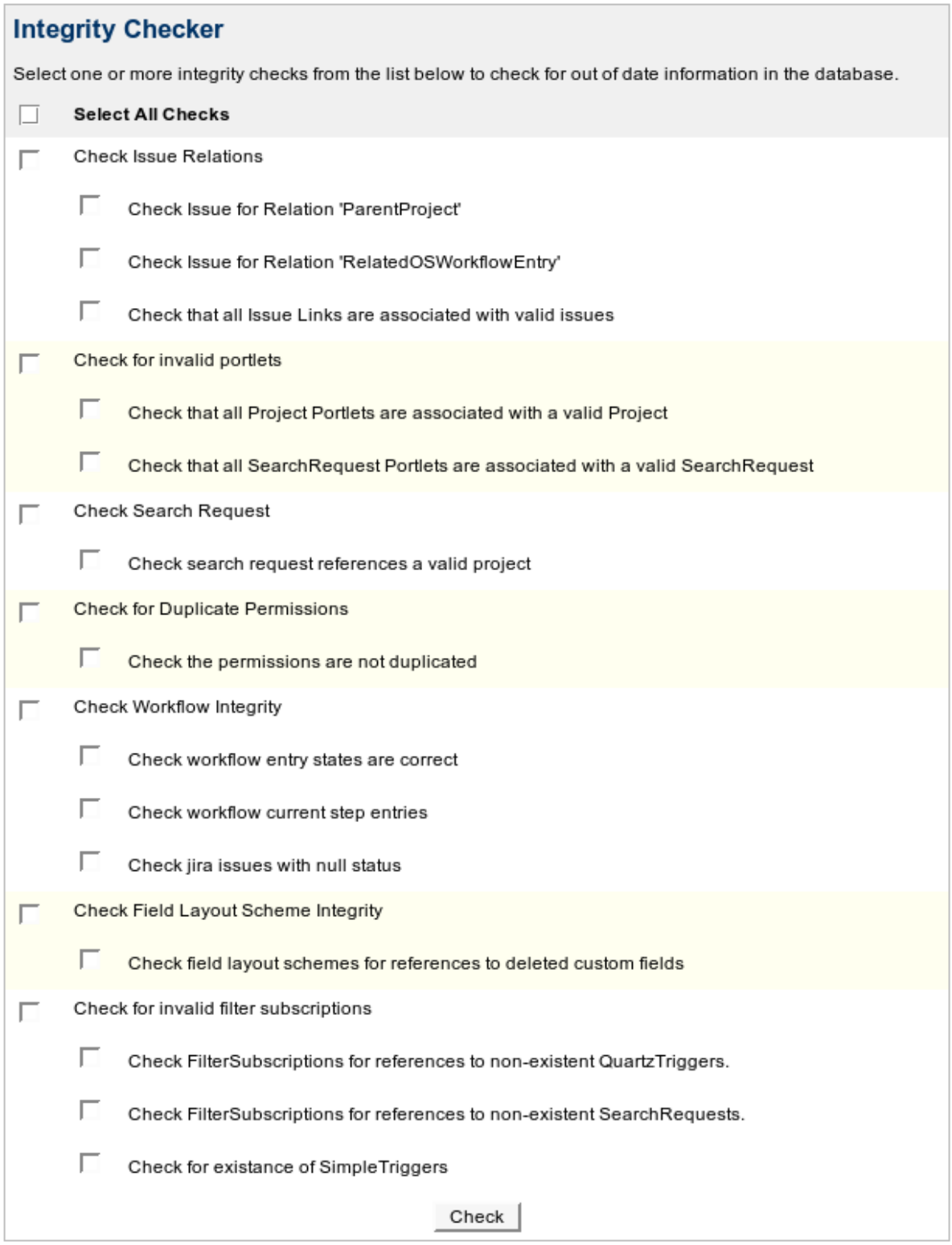
Link to Administration section



Link to Administration section

3. Under the '**System**' sub-menu in the left-hand navigation solumne, click the '**Integrity Checker**' link.

4. The '**Integrity Checker**' screen will be displayed:



IntegrityChecker Screenshot

5. The integrity checker has a number of 'integrity checks' that look for common inconsistencies in the data. Select one or more checks you would like to run, then click the '**Check**' button.
6. After the selected checks run, the preview screen will be shown.

   The screen provides details about the existing data inconsistencies. If any inconsistencies were found, the '**Fix**' button will also appear on the page. The messages in red describe

inconsistencies that the check will correct if it is chosen and the '**Fix**' button is clicked. Messages that appear in yellow are warnings that the check will not correct; JIRA will auto-recover from these inconsistencies when an action is taken on an issue.

Select any inconsistencies that you would like to correct, then click the '**Fix**' button.

Note:
We **strongly** recommend taking a backup of your data before correcting any data inconsistencies.

7. If any inconsistencies were found and you chose to correct them, you will be presented with a summary screen describing all the corrective actions that have taken place.

1.14.7. Performance

While performing some of the following steps to help improve the performance of your JIRA install, it will also help to gather some data on just where your performance bottle necks might be. The following page in our Confluence space provides a step-by-step guide on gathering the type of information you will need to help diagnose and resolve performance problems.

1.14.7.1. Profiling

To quantify performance problems, you can turn JIRA profiling on. You will then get a profile for each URL requested in the logs:

```
[Filter: profiling] Turning filter on [jira_profile=on]
[116ms] - /secure/Dashboard.jspa
  [5ms] - IssueManager.execute()
    [5ms] - IssueManager.execute()
      [5ms] - Searching Issues
  [29ms] - IssueManager.execute()
    [29ms] - IssueManager.execute()
      [29ms] - Searching Issues
        [28ms] - Lucene Query
          [23ms] - Lucene Search
```

One performance problem with JIRA can be the application server that you are running. The database connection pooling, JSP compilation times and resource allocation are different between application servers. Known slow servers include JBoss 3.x, Tomcat 4.0 and Tomcat 4.1.24. The fastest servers for JIRA at the moment are more recent versions of Tomcat, as well as Resin and Orion.

Databases can also have a large impact on performance, particularly if the database is accessed across a network, or has not been indexed properly.

If you can't change your application server, there are performance improvements available, both by tuning your server, database and through setting certain JIRA options.

1.14.7.2. Environment options

Virus checking

If you are experiencing slowness with JIRA, try running JIRA with virus checking disabled. As JIRA creates many temporary files, **virus checking software can slow JIRA dramatically**. McAfee's NetShield 4.5 in particular claims to let you exclude folders from scanning, but doesn't actually — upgrade to 7.0.0 to fix this. Symantec must be **uninstalled** — painful experience has proven that even stopping the services does not prevent it slowing JIRA down.

Network shares

JIRA needs fast access to the local filesystem. If you are hosting JIRA, or its index directory, on a network share (SMB, NFS etc), this can cause a large loss in performance. Run JIRA with fast local disk access.

SSL or HTTPS

Although some organisations have a requirement to run JIRA over SSL or HTTPS, please note that this this can affect performance.

1.14.7.3. Database options

JDBC drivers

Different JDBC drivers have different performance characteristics. Ensure that you are using the latest patched version of the JDBC drivers for your database.

Databases

JIRA Standalone (and many application servers) ship with an in-memory database like hsqldb. Using another database (eg MySQL, PostgreSQL or Oracle) will usually result in higher performance.

Network latency

The latency between the database server and the server hosting JIRA can be a source of performance problems. If the database is hosted on a different machine to JIRA, please check the ping times between the servers.

Index your database

JIRA 3.0+ automatically creates database indexes when the database is first created. However, if you have been doing in-place JIRA upgrades from earlier versions (not dropping/recreating the database tables), your database will not be indexed. Doing a full XML backup and restoring into an empty database will fix this. Additional indexes may be created by hand, but this is usually not necessary.

1.14.7.4. JDK options

Choose the latest JDK version

The latest JDKs contain performance optimisations that will improve the performance of JIRA. JIRA uses a lot of reflection, which was greatly improved in the 1.4 release.

Use the Server JVM

Sun ships two versions of the JDK, a client version and a server version. They have different characteristics such as memory management and inline optimisations. You may need to explicitly start your application server like "java -server -jar <app-server-jar>.jar". With JDK 1.5 it is best to leave this unset.

Allocate enough Memory

By default, many application servers are not started with enough memory for JIRA to run at an optimum speed. A lack of memory increases garbage collection time, as garbage collection has to

be run more frequently.

To see if lack of memory is causing slowness, please add these parameters to JIRA's startup command:

```
-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -verbose:gc -Xloggc:${LOG}/gc.log
```

Where `${LOG}/` is a filesystem path to your log directory. Garbage collection times will then be logged in `gc.log`.

You may need to start your application server like "java -server -Xms100m -Xmx300m <app-server-jar>.jar". See Increasing JIRA Memory for more details.

1.14.7.5. JIRA options

Enable GZip Compression

JIRA can compress page contents between the server and your browser, resulting in improved performance especially over slow connections. Check that GZip compression is enabled in Administration -> Global Settings -> General Configuration (unless you are using `mod_proxy`.

Remove HSQLDB parameters in JIRA Standalone

If you are using JIRA Standalone *modified to use an external database*, make sure you **delete** the highlighted section in your `conf/server.xml`, which otherwise results in poor performance:

```
    <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
      username="jirauser"
      password="jirapassword"
      driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost/jiradb?autoReconnect=true"
      minEvictableIdleTimeMillis="4000"
      timeBetweenEvictionRunsMillis="5000"
      />
```

External User Management

If External User Management is turned on (**not** the default), JIRA will not cache users & groups, potentially resulting in slow access.

1.14.7.6. Application Server options

Database Connection pooling.

Obtaining a connection to a database is an expensive operation, and most application servers maintain a pool of open connections to reduce this overhead. It is worth checking that you have a sensible number of connections pooled, sensible expiry times etc. This is configured in your app server.

If you are using the standalone version of JIRA or Apache Tomcat you can modify the DBCP connection pool in Tomcat's server.xml (or possibly jira.xml, depending on how you have setup JIRA). The values you will be most interested in modifying are the following:

```
                <Resource name="jdbc/JiraDS" auth="Container" type="javax.sql.DataSource"
                username="sa"
                password=""
                driverClassName="org.hsqldb.jdbcDriver"
                url="jdbc:hsqldb:${catalina.home}/database/jiradb"
                minEvictableIdleTimeMillis="4000"
```

```
                timeBetweenEvictionRunsMillis="5000"
                maxActive="20"
                    minIdle="4"
                    maxIdle="8"
                />
```

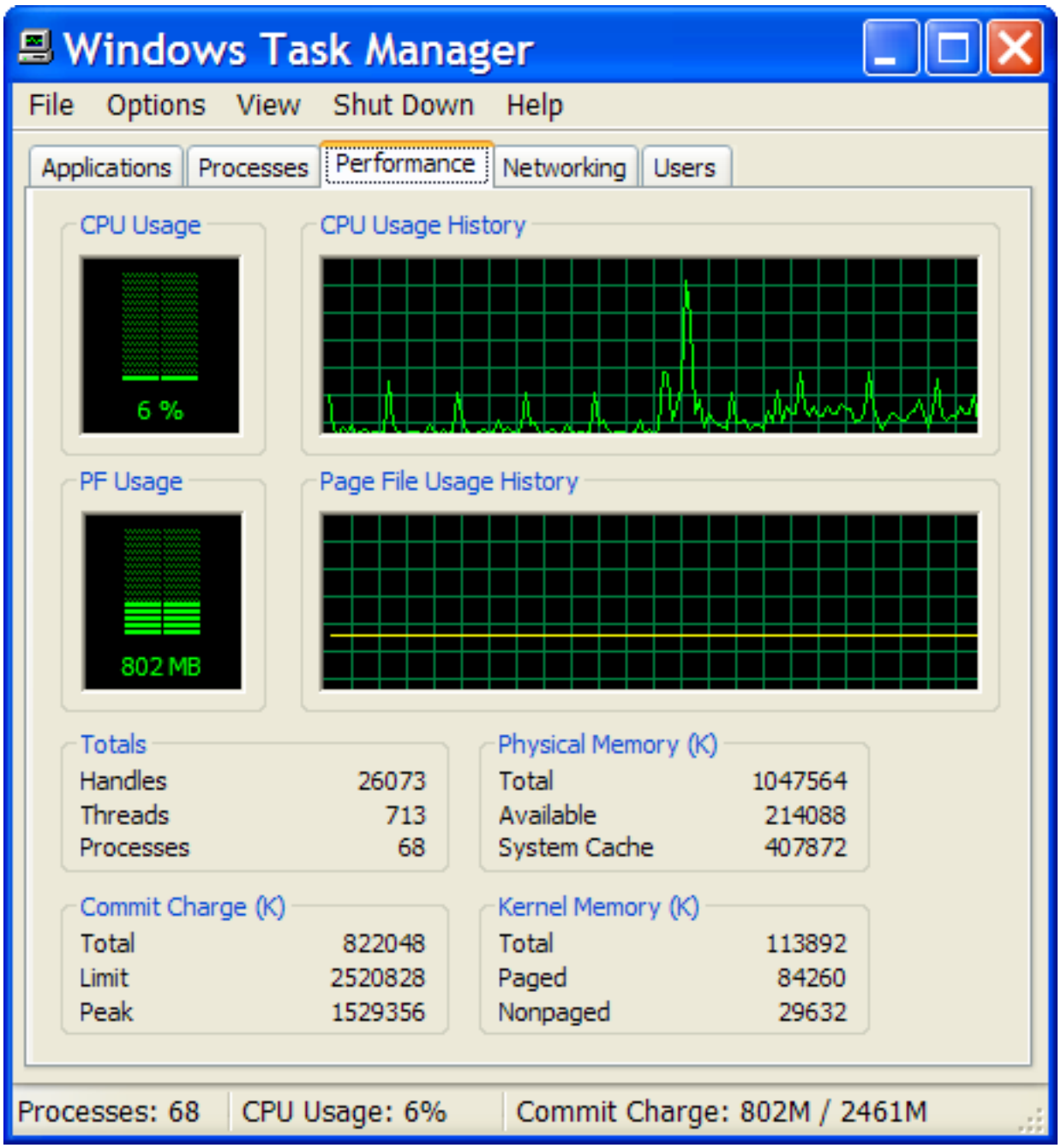
For information on what these values mean please view the Apache DBCP documentation

Other application server tuning may be of benefit. Consult your application server documentation for more information.

1.14.7.7. Server specifications

JIRA performance is very dependent on CPU and available memory. Lack of physical memory, or overly high maximum heap size setting (the -Xmx flag) can seriously degrade JIRA performance, as memory accesses result in constant swapping of data between memory and disk (virtual memory).

On Windows, you can see what your system is doing in the Task Manager:



Windows Task Manager showing CPU and memory stats

On Linux/Solaris, `vmstat 1` will print virtual memory and CPU statistics:

```
$ vmstat 1
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 9  0 520512  27132  15216 318944    3    2    65    40    0     3 10  3 85  2
12  0 520512  27004  15216 319080    0    0   104     0 2041 10992 88  8  3  0
20  0 520512  26764  15228 319068    0    0     0   436 2196 12869 85 13  2  0
11  0 520512  26700  15228 319068    0    0     0     0 1959  4041 88  9  4  0
20  0 520512  25724  15228 319068    0    0     0     4 2137  3307 84 14  2  0
17  0 520512  25724  15228 319068    0    0     4     0 2017 10488 89  8  3  0
 9  0 520512  25468  15228 319068    0    0     0     0 1886  7532 86 11  3  0
...
```

(this is a very busy server with ~97% CPU usage, but fortunately no swapping)

This system info can be captured over a long time with `vmstat -n 1 > vmstat.log`

On Linux, CPU and memory info can be obtained with `cat /proc/cpuinfo` and `cat`

`/proc/meminfo` respectively.

If you need to raise a support request, please include this info (`vmstat 1`, `/proc/cpuinfo`, `/proc/meminfo`).

1.14.8. Database Indexing

JIRA 3.0 and later creates database indices automatically when the underlying table is created in the database. This means that if you are doing a fresh install of JIRA 3.0 (or later) you do not need to create indices manually. If you are upgrading JIRA from an earlier version (e.g. JIRA 2.6) and do not wish to create the indices manually, please follow these instructions and recreate (drop and create) JIRA's database (or remove all tables in the database) **AFTER** successfully exporting your data and before doing the import into the new version of JIRA. Removing the database will force JIRA to recreate all tables in the database and hence create all required indices.

Warning:

If upgrading from JIRA 2.6.1 or earlier to JIRA 3.0 (or above), JIRA will not create indices automatically, unless the database is removed and recreated.

If you do not wish to drop and recreate JIRA's database, you can add the indices manually by running the SQL statements shown below.

The syntax for creating indices differs between databases, so consult your documentation for the your database. In addition, if you change the database tables or fields that you use in `entitymodel.xml`, you will need to change the shown SQL statements.

Below is the SQL for creating indices on PostgreSQL (you will probably need to alter this for your database):

```
CREATE INDEX action_issue ON jiraaction (issueid, actiontype);

CREATE INDEX chggroup_issue ON changegroup (issueid);

CREATE INDEX chgitem_chggrp ON changeitem (groupid);

CREATE INDEX cf_cfoption ON customfieldoption (CUSTOMFIELD);

CREATE INDEX cfvalue_issue ON customfieldvalue (ISSUE, CUSTOMFIELD);

CREATE INDEX attach_issue ON fileattachment (issueid);

CREATE INDEX subscrpt_user ON filtersubscription (FILTER_I_D, USERNAME);
CREATE INDEX subscrptn_group ON filtersubscription (FILTER_I_D, groupname);

CREATE INDEX issue_key ON jiraissue (pkey);

CREATE INDEX issuelink_src ON issuelink (SOURCE);
CREATE INDEX issuelink_dest ON issuelink (DESTINATION);
CREATE INDEX issuelink_type ON issuelink (LINKTYPE);

CREATE INDEX linktypename ON issuelinktype (LINKNAME);
CREATE INDEX linktypestyle ON issuelinktype (pstyle);

CREATE INDEX node_source ON nodeassociation (SOURCE_NODE_ID, SOURCE_NODE_ENTITY);
CREATE INDEX node_sink ON nodeassociation (SINK_NODE_ID, SINK_NODE_ENTITY);

CREATE INDEX ntfctn_scheme ON notification (SCHEME);

CREATE INDEX osgroup_name ON groupbase (groupname);

CREATE INDEX mshipbase_user ON membershipbase (USER_NAME);
CREATE INDEX mshipbase_group ON membershipbase (GROUP_NAME);
```

```
CREATE INDEX osproperty_all ON propertyentry (ENTITY_NAME, ENTITY_ID);

CREATE INDEX osuser_name ON userbase (username);

CREATE INDEX sec_scheme ON schemeissuesecurities (SCHEME);

CREATE INDEX sec_security ON schemeissuesecurities (SECURITY);

CREATE INDEX prmssn_scheme ON schemepermissions (SCHEME);

CREATE INDEX sr_author ON searchrequest (authorname);
CREATE INDEX sr_group ON searchrequest (groupname);

CREATE INDEX user_source ON userassociation (SOURCE_NAME);
CREATE INDEX user_sink ON userassociation (SINK_NODE_ID, SINK_NODE_ENTITY);

CREATE INDEX workflow_scheme ON workflowschemeentity (SCHEME);
```

Once you have created the index, you may need to tell your database to recompute its indices. For PostgreSQL, the command is `vacuumdb -U username -z -v database-name`. Consult your database documentation for your database specific command.

1.14.9. Precompiling JSP pages

If you decided to go the extra mile and extend JIRA's build process to precompile JSP pages, keep in mind that the "include" directory in the JIRA web application needs to be excluded from precompilation. The reason for this is that the JSP files in the "include" directory are not proper JSP files, but are includes that are only meant to be compiled as part of larger JSP pages.

For example, to exclude the JSP pages in the "include" directory when using Maven use the <exclude> sub-element of the <ant:jspc> task, as shown:

```
<ant:path id="jspc.classpath">
  <ant:pathelement location="${tomcat.home}/common/lib/jasper-runtime.jar"/>
  <ant:pathelement location="${tomcat.home}/common/lib/jasper-compiler.jar"/>
  <ant:pathelement location="${tomcat.home}/common/lib/servlet.jar"/>
  <ant:path refid="maven-classpath"/>
  <ant:path refid="maven.dependency.classpath"/>
  <ant:pathelement path="${maven.build.dest}"/>
  <ant:pathelement path="${java.home}/lib/tools.jar"/>
</ant:path>
<ant:jspc
  package="${pom.package}.jsp"
  destDir="${jspOutDir}"
  srcdir="${warSource}"
  uriroot="${warSource}"
  uribase="/${pom.artifactId}"
  verbose="2"
  classpathref="jspc.classpath">
  <ant:include name="**/*.jsp"/>
  <ant:exclude name="**/includes/**/*.jsp"/>
</ant:jspc>
```

1.14.10. Logging & Profiling

1.14.10.1. Logging

JIRA uses a powerful logging module called log4j.

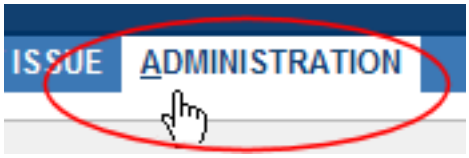
The default logging levels can be changed either;

- temporarily — your change to the logging level will not persist after you next restart JIRA, or
- permanently — your change to the logging level will persist, even after you restart JIRA

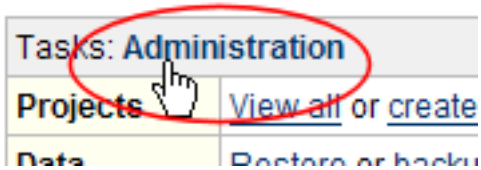
For example, when troubleshooting, you might temporarily change the logging level from 'WARNING' to 'INFO' so as to get a more detailed error message or a stack trace.

To temporarily change the logging level:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Under the '**System**' sub-menu in the left-hand navigation column, click the '**Logging & Profiling**' link.
4. The '**Logging & Profiling**' page will display. This lists all the defined log4j categories and their current logging levels. To edit the logging level of a category, click the '**Edit**' link next to the category in the list.
5. Choose the new logging level for the category, then click '**Update**'.

To permanently change the logging level:

1. Edit the `log4j.properties` file, which is found in the `WEB-INF/classes/` directory under the JIRA web application directory.

> **Note:**
> The `log4j.properties` file that ships with JIRA has the default logging levels specified. For more information about log4j (e.g. how to define new logging categories), and about the format of the `log4j.properties` file, please refer to the documentation on the log4j site.

2. Restart JIRA.

> **Note:**
> If your application server itself configures logging (e.g. JBoss), you may need to remove the `log4j.properties` file. On some servers (e.g. JBoss 3.0), you may also need to remove the entire `log4j.jar` file to get logging to work.

1.14.10.2. Profiling

If you are experiencing performance issues with JIRA, it is often helpful to see where the slow-downs occur. To do this you can enable profiling as described below, and then analyse the performance traces that JIRA will produce for every request. An example of a profiling trace is shown below:

```
[Filter: profiling] Turning filter on [jira_profile=on]
[116ms] - /secure/Dashboard.jspa
  [5ms] - IssueManager.execute()
    [5ms] - IssueManager.execute()
```

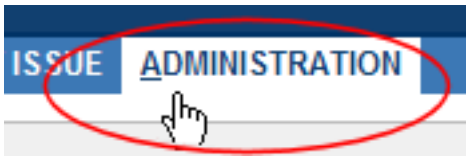
```
      [5ms] - Searching Issues
  [29ms] - IssueManager.execute()
    [29ms] - IssueManager.execute()
      [29ms] - Searching Issues
        [28ms] - Lucene Query
          [23ms] - Lucene Search
```

Profiling can be enabled either;

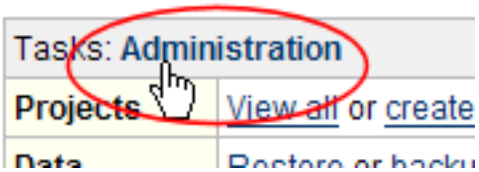
- temporarily — profiling will be enabled until you next restart JIRA, or
- permanently — profiling will remain enabled, even after you restart JIRA

To temporarily enable profiling:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Under the '**System**' sub-menu in the left-hand navigation column, click the '**Logging & Profiling**' link.
4. The '**Logging & Profiling**' page will display. Scroll to the bottom of the screen. The '**Profiling**' section will inform you whether profiling is currently turned '**ON**' or '**OFF**', and will also show or hide the '**Disable profiling**' and '**Enable profiling**' links respectively.
   - To turn Profiling '**ON**', click the '**Enable profiling**' link. JIRA will start generating profiling traces in its log.
   - To turn Profiling '**OFF**', click the '**Disable profiling**' link.

To permanently enable profiling:

1. Edit `atlassian-jira/WEB-INF/web.xml` (if you are using JIRA Standalone) or `webapp/WEB-INF/web.xml` in your JIRA installation directory (if you are using JIRA EAR/WAR).
2. Find the following entry:

```
        <filter>
            <filter-name>profiling</filter-name>
            <filter-class>com.atlassian.jira.web.filters.JIRAProfilingFilter</filter-cl
            <init-param>
                <!-- specify the which HTTP parameter to use to turn the filter on or o
                <!-- if not specified - defaults to "profile.filter" -->
                <param-name>activate.param</param-name>
                <param-value>jira_profile</param-value>
            </init-param>
            <init-param>
                <!-- specify the whether to start the filter automatically -->
                <!-- if not specified - defaults to "true" -->
                <param-name>autostart</param-name>
                <param-value>false</param-value>
            </init-param>
        </filter>
```

3. Modify the `autostart` parameter to be **true** instead of **false**. That is:

```
            <init-param>
                <!-- specify the whether to start the filter automatically -->
                <!-- if not specified - defaults to "true" -->
                <param-name>autostart</param-name>
                <param-value>true</param-value>
            </init-param>
```

4. Save the file.

   If you are running JIRA Standalone this is all you have to do. Profiling will be enabled when you restart JIRA.

5. If you are running JIRA EAR/WAR, re-build and re-deploy the JIRA web application using the `build` script and the instructions for your application server.

1.14.11. Thread Dump

Occassionally, JIRA may appear to 'freeze' during execution of an operation. During these times, it is helpful to retrieve a **thread dump** - a log containing information about currently running threads and processes within the Java Virtual Machine. Taking thread-dumps is a non-destructive process that can be run on live systems. This document describes the steps necessary to retrieve a **thread dump**.

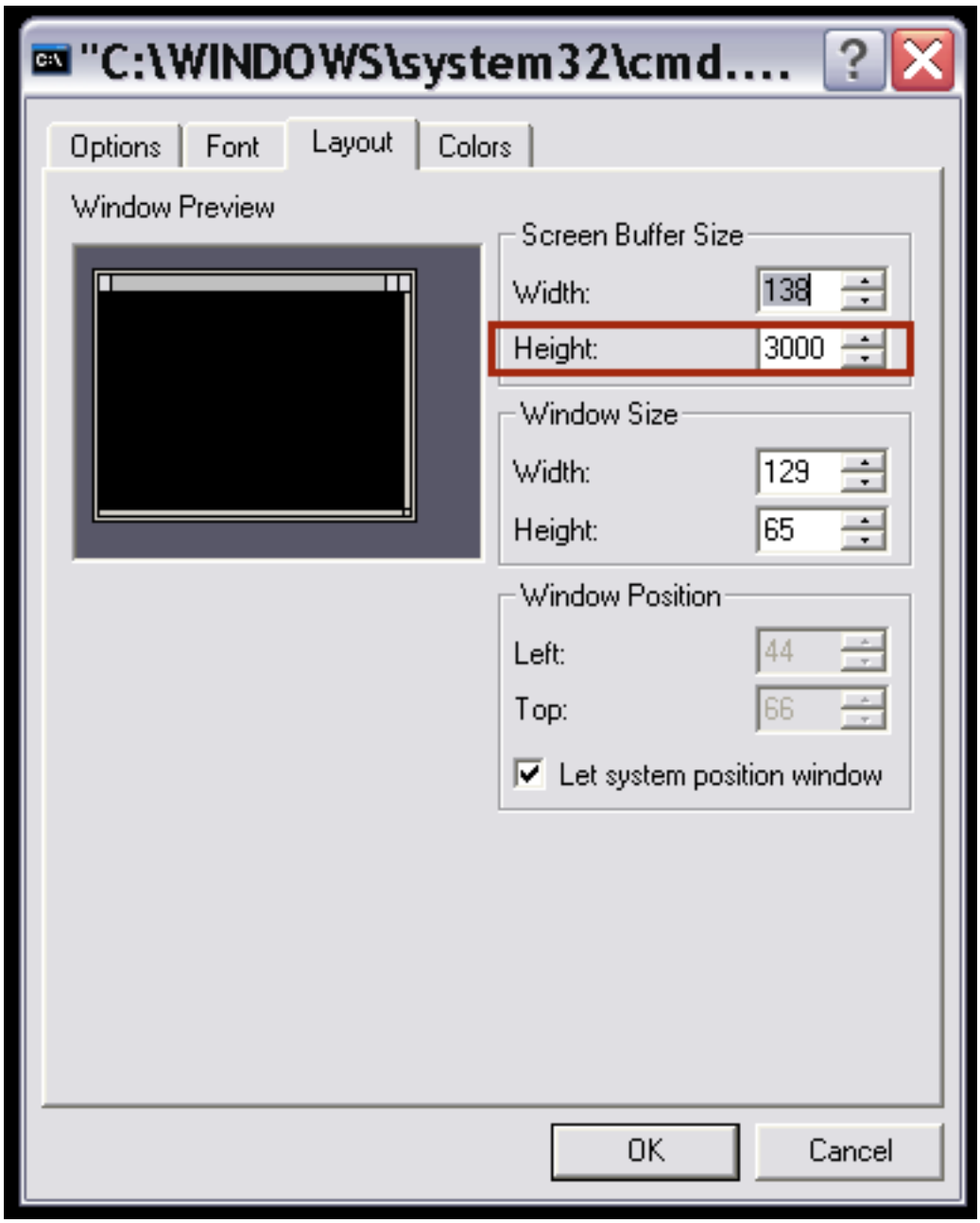
The steps necessary to retrieve the **thread dump** are dependant on the operating system JIRA is running in - please follow the appropriate steps:

- Windows Environment
- UNIX Environment

1.14.11.1. Windows Environment

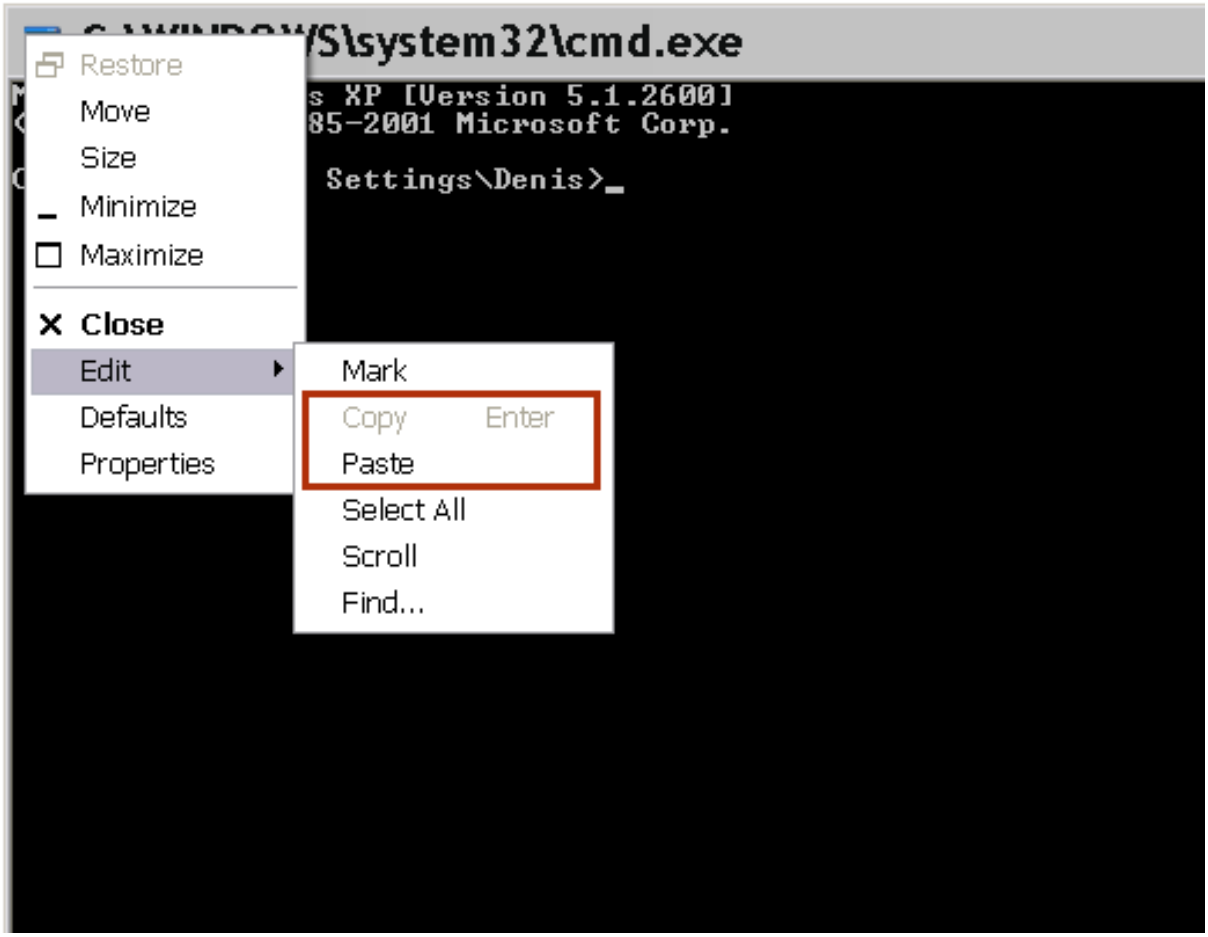
Running JIRA from startup.bat

1. In the **Command Console** window where JIRA is running, open the properties dialog box by right clicking on the title bar and selecting "**Properties**".
2. Select the **Layout** tab.
3. Under **Screen Buffer Size**, set the **Height** to **3000**.



Console Window Properties

4. Click **OK**.
5. With the same command console in focus, press **CTRL-BREAK**. This will output the thread dump to the command console.
6. Scroll back in the command console until you reach the line containing "Full thread dump". A sample thread dump is available here.
7. Right click the title bar and select **Edit -> Mark**. Highlight the entire text of the thread dump.
8. Right click the title bar and select **Edit -> Copy**. The thread dump can then be pasted into a text file.



Console Window Copy Paste

JIRA running as a Windows Service

1. Visit http://www.adaptj.com/root/main/download and click Launch
2. Click Run for any security warnings
3. Select Process -> Thread Dump
4. Under Process Id, select the '...' button.
5. From the drop-down list, select the JIRA process. Users running JIRA Standalone should select the 'Java (Tomcat) ...' option. Users running JIRA EAR/WAR should select their application server process.
6. Click OK to capture the thread dump.
7. Save the output to a file, eg 'threaddump.log'

1.14.11.2. UNIX Environment

1. Identify the **java** process that JIRA is running in. This can be achieved by running a command similar to:
   ```
   ps -ef | grep java
   ```
   The process will appear similarly as follows:
   ```
   keithb     910   873  1 17:01 pts/3    00:00:18 /usr/java/jdk/bin/java -Xms128m -Xmx
   -Xms128m -Xmx256m -Djava.awt.headless=true -Djava.util.logging.manager=org.apache.ju
   -Djava.awt.headless=true -Djava.endorsed.dirs=/tmp/atlassian-jira-enterprise-3.6-sta
   -classpath :
   ```
2. In order to retrieve the thread dump, execute the command
   ```
   kill -3 <pid>
   ```
   where **pid** is the process id - in this case 910.
3. The thread dump is logged to the console in which JIRA was started. A sample thread dump is available here.

1.14.12. Using robots.txt to hide from Search Engines

The robots.txt protocol is used to tell search engines (Google, MSN, etc) which parts of a website should not be crawled.

For JIRA instances where non-logged-in users are able to view issues, a robots.txt file is useful for preventing unnecessary crawling of the Issue Navigator views (and unnecessary load on your JIRA server).

1.14.12.1. Editing robots.txt

JIRA (version 3.7 and later) installs the following `robots.txt` file at the root of the JIRA webapp:

```
# robots.txt for JIRA
# You may specify URLs in this file that will not be crawled by search engines (Google, MS
#
# By default, all SearchRequestViews in the IssueNavigator (e.g.: Word, XML, RSS, etc) and
# (XML, Printable and Word) are excluded by the /sr/ and /si/ directives below.

User-agent: *
Disallow: /sr/
Disallow: /si/
```

Alternatively, if you already have a `robots.txt` file, simply edit it and add **`Disallow: /sr/`** and **`Disallow: /si/`**.

1.14.12.2. Publishing robots.txt

The `robots.txt` file needs to be published at the root of your JIRA internet domain, e.g. **`jira.mycompany.com/robots.txt`**.

Note:

If your JIRA instance is published at **`jira.mycompany.com/jira`**, change the contents of the file to `Disallow: /jira/sr/` and `Disallow: /jira/sr/`. However, you still need to put `robots.txt` file in the root directory, i.e. `jira.mycompany.com/robots.txt` (not `jira.mycompany.com/jira/robots.txt`).

1.15. Appendix A - Extending JIRA

1.15.1. Extending JIRA

JIRA is very flexible, and has a number of extension points where JIRA's data can be queried or its functionality extended. You can also download the JIRA Plugin Development Kit. This contains full source code for seven working plugins (and growing), skeleton project templates for creating your own plugins, full JIRA API documentation and all library dependencies.

Custom Field Types	JIRA comes with various custom field types defined. New types can be written and plugged into JIRA. See the Writing Custom Field Types Tutorial for more information.
User Formats	JIRA comes with many options to change the look and feel of features in the system. User formats are a feature that can be customised by plugins. You can write your own User Format plugin to change the display of user details in JIRA, e.g. display a profile picture. See the User Format Plugin Guide for more information.

Portlets	New portlets can be created by writing a Java class, a template and an XML descriptor file, packaged as a JIRA plugin. See How To Create a JIRA Portlet for more information.
Reports	JIRA comes with various reports built-in. Using the plugin system, new reports can be written, providing new ways of viewing and summarising JIRA's data.
Workflow functions and conditions	JIRA's issue workflow (states and state transitions an issue can go through) can be customized through the web interface (see the workflow documentation. The workflow engine (OSWorkflow) provides hooks where you can plug in your own behaviour: • Run arbitrary Java when a certain transition occurs, via post-functions • Limit visibility of transitions to certain users, via conditions • Validate input on transition screens (eg. in comments), via validators See the guide to creating custom workflow elements for how to write your own workflow post-functions, conditions and validators. Once written, these can be packaged as plugins and reused.
Issue and Project Tabs	When viewing an issue, some issue information (comments, change history) is displayed in tabs:  Issue tab panels Likewise, the 'Browse Project' page contains tab panels displaying project information:  Project tab panels By writing a plugin, you can add new issue or project tab panels to JIRA. For instance, you may wish to display project/issue data pulled in from an external source. This is how JIRA's Subversion. See the plugin guide for more information on writing these plugin types.
Listeners	JIRA has a complete event subsystem which fires events whenever anything happens. For example an `ISSUE_CREATED` event is fired whenever an issue is created. A listener is just a class which implements a `JiraListener` interface and is called whenever events occur in JIRA. Using those events, you can then perform any action you want. For example the email sent by JIRA is driven by the `MailListener`.

	This is useful when you want to drive or affect external systems from events which occur within JIRA - usually used to *push* data into outside systems. For more information, read the listeners documentation.
Services	Services are classes which implement the `JiraService` interface. When installed, you specify an update period and JIRA will call the `run()` method of your service periodically. A sample service is provided called `POPCommentService`. This service checks a particular POP mailbox periodically and if it finds messages, tries to extract an issue key from the subject. If the subject contains a key, the body of the mail is added as a comment to the message. Services are useful when you want to periodically *pull* data into JIRA from outside systems. For more information, see the services guide.
SOAP and XML-RPC remote interfaces	JIRA has a growing SOAP and XML-RPC interface. This enables you to drive JIRA automatically from external systems. For example you can have a Java program, Perl script or C# client add issues to JIRA. See the JIRA RPC overview for general information. For building RPC clients, check out the SOAP client tutorial and XML-RPC client tutorial. New RPC endpoints can also be added to JIRA as plugins - see RPC Endpoint Plugins.
Java	JIRA has a full set of Java APIs that can be used to update information with in JIRA. You can view the API here. JIRA commercial customers get full access to the JIRA source (see bottom of the downloads page), so you can modify JIRA itself if necessary. See the Building JIRA from Source page for more information.

1.15.2. Listeners

Listeners are unique to JIRA, and a very powerful way to extend it.

JIRA has a complete event subsystem which fires events whenever anything happens inside the application. For example an `ISSUE_CREATED` event is fired whenever an issue is created.

A Listener is a class that implements one of the Listener interfaces. It is then called whenever events occur in JIRA. Using those events, you can then perform any action you want. For example the email sent by JIRA is driven by the MailListener.

Listeners are most useful when you want to drive or affect external systems from events which occur within JIRA.

1.15.2.1. Listener Interfaces

There are currently two different concrete Listeners within JIRA (both of which extend the base

JiraListener interface).

com.atlassian.jira.event.**JiraListener**	The base interface which all other JIRA listener interfaces extend. Covers core listener properties like uniqueness, description, parameters etc. API doc
com.atlassian.jira.event.issue.**IssueEventListener**	The main listener interface in JIRA, used whenever anything happens to an issue. API doc
com.atlassian.jira.event.user.**UserEventListener**	This listener is called whenever anything happens to a user within JIRA. API doc

1.15.2.2. Example Listeners

The examples provided may be freely used and modified for use in your own environment. The source of all examples is available and should give you good overview of how simple it is to write your own listeners. Both example listeners are included with JIRA 2.1, and both implement `UserEventListener` and `IssueEventListener`.

DebugListener (source)

This is a very simple listener that prints events and their content to System.out whenever they are received.

To test this listener, add a listener with the class `com.atlassian.jira.event.listeners.DebugListener`.

MailListener (source)

This listener is how mail notifications are currently sent from within JIRA, and a good example of a more complex listener. It basically listens for events, and turns them into email notifications using Velocity templates to generate the mail bodies.

This listener is usually always turned on in JIRA - see Email Notifications for more details. If you want to write more complex or more specific notifications, you can disable the internal MailListener and add your own.

Other examples of useful tasks that can be accomplished with listeners are:

Send SMS or IM notifications
A listener could easily send notifications for various events via SMS or instant messenger (eg ICQ or AIM) - or anywhere that you have a Java library to send messages.

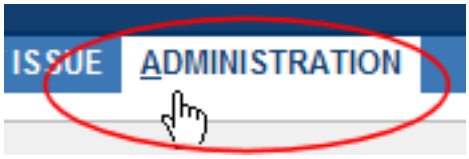
Group notifications:
A listener could notify certain groups of issue changes, depending on the content of the issue. For example any issue containing "windows" in the environment could notify your "windows-developers" group.

1.15.2.3. Registering a Listener

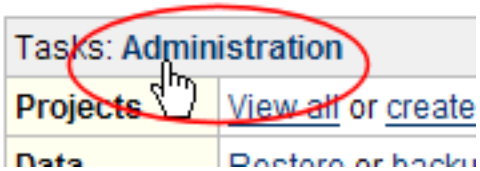
To register a listener:

1. Make sure your listener class is in the classpath where JIRA can see it - the best locations are usually the `WEB-INF/classes` or `WEB-INF/lib` (as a JAR) directories within the JIRA

web application.
2. Log in as a user with the '**JIRA System Administrators**' global permission.
3. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

4. Click the **Listeners** link in the **System** section of the left-hand navigation column.
5. Enter a **Name** (a nice name for this listener) and **Class** (the fully qualified class of your listener) for your listener and click **Add**.
6. The listener will now be added. Click **Edit** for your listener to edit its properties.

1.15.2.4. Editing Listener Properties

At any time you can edit a listeners properties by clicking **Edit** for that listener in the **Listeners** section of the **Administration** tab.

1.15.2.5. Removing a Listener

To remove a listener, just click **Del** for that listener in the **Listeners** section of the **Administration** tab.

1.15.2.6. Custom Events

With the ability to add custom events to JIRA, the Listener must be updated to deal with the event as appropriate. This is possible by providing an implementation for the method `customEvent(IssueEvent event)` in the Listener. For example, the MailListener implementation passes the custom event on for notification processing. The DebugListener logs that the custom event has been fired.

1.15.3. Services

A service is a class that runs periodically within JIRA. Since a service runs inside JIRA, it has the ability to use all of the JIRA API — and, as it is written in Java, it can use any Java libraries.

Services are useful because they enable you to integrate with external systems by pulling data into JIRA periodically. JIRA comes with a number of pre-written services, and custom services can be written and plugged in at runtime. If you want a service to perform typical operations on JIRA issues (eg. close a list of issues meeting certain criteria), then the Jelly Service can be configured to run a custom Jelly script.

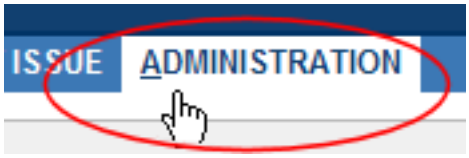
1.15.3.1. Registering a Service

Services are set up as follows.

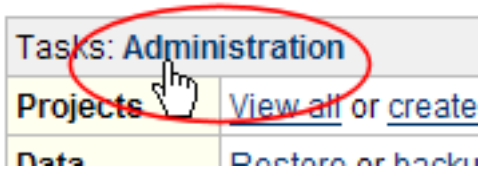
1. For custom-written services, make sure your service class is in the classpath where JIRA can see

it — the best locations are usually the **WEB-INF/classes** or **WEB-INF/lib** (as a JAR) directories within the JIRA web application.

2. Log in as a user with the '**JIRA System Administrators**' global permission.
3. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section

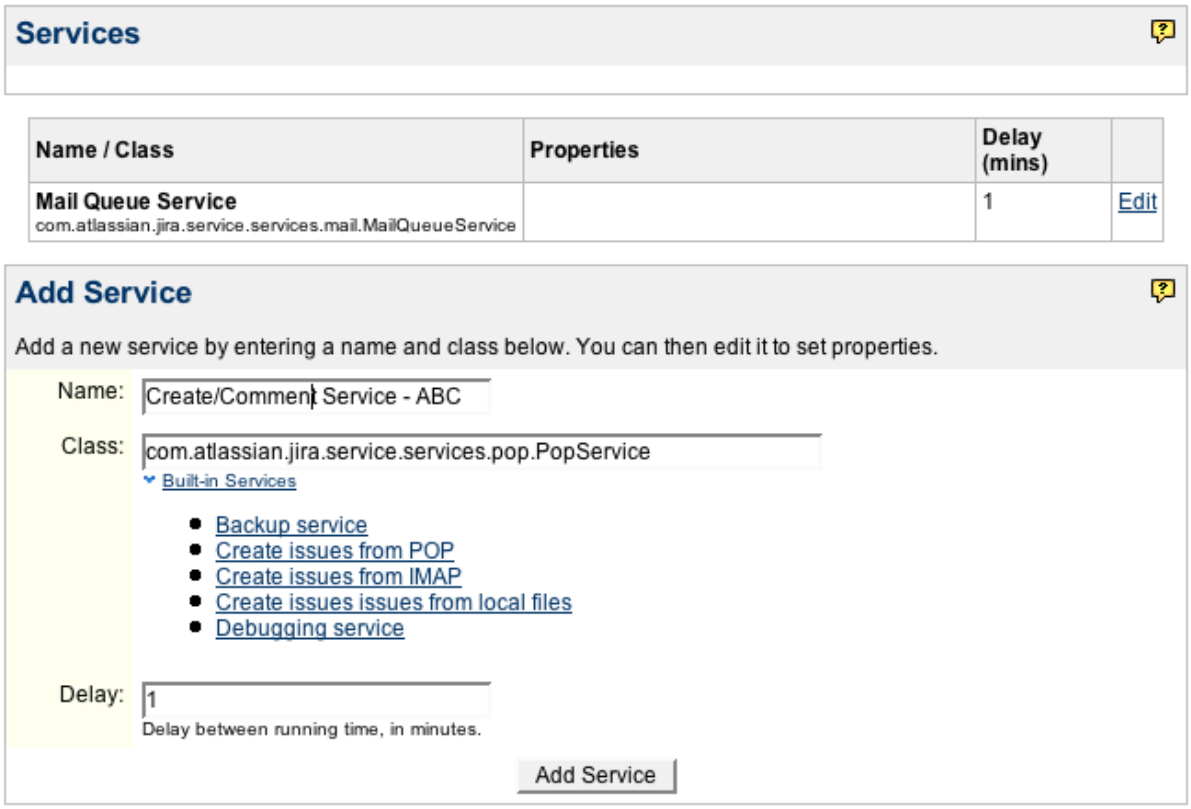


Link to Administration section

4. Under the '**System**' sub-menu on the left, click the '**Services**' link.
5. Fill out the **Add Service** form with the following parameters:

   Name
   a descriptive name for this service
   Class
   the fully qualified class of your service. Will likely be of the form **com.atlassian.jira.service.services.*type*.*Type*Service**. See Sample services for provided service class names.
   Delay
   the delay (in minutes) between service runs

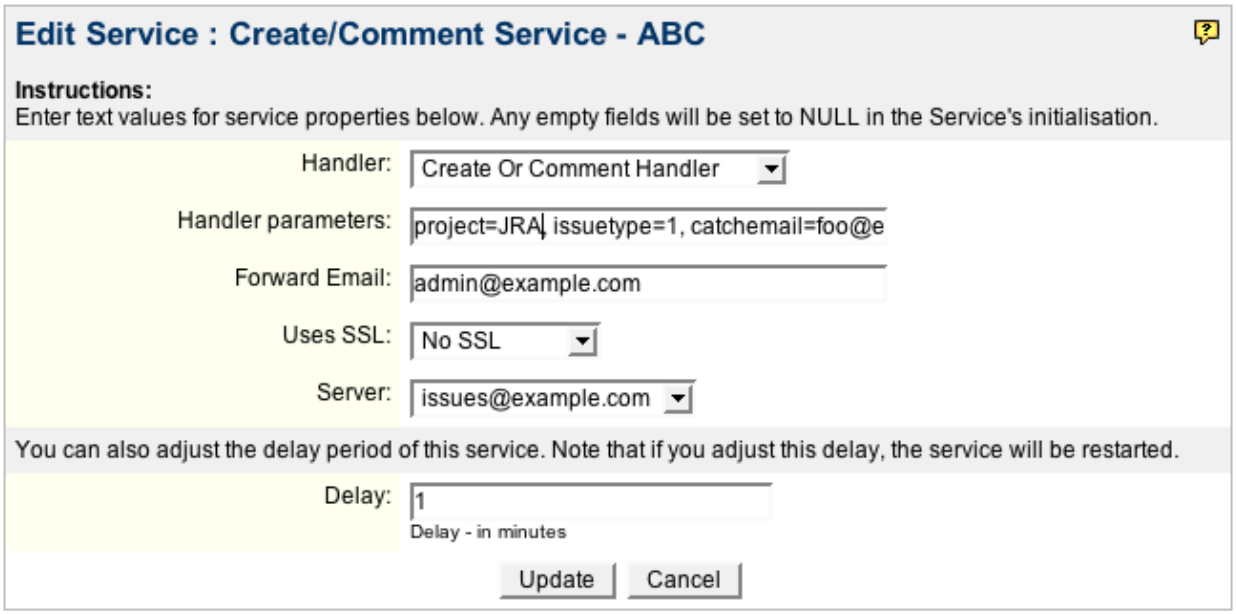
   For example, adding a POP Service:



Add service

Now click **Add Service**

6. This will bring up the '**Edit Service**' screen to configure the service. Depending on the service, you may now be required to specify a MessageHandler, a helper class that processes email messages. MessageHandlers are configured with a parameter string, a comma-separated list of name=value pairs. Consult the tables below as to what parameters each MessageHandler accepts. The following screenshot shows a CreateIssueHandler being attached to a POP Service:



Editing a service

1.15.3.2. Editing Service Properties

At any time you can edit a service's properties by clicking **Edit** for that service in the **Services** section of the **Administration** tab.

1.15.3.3. Mail Service Properties

In addition to Message Handlers, the mail services POP Service and IMAP Service can be further configured with further properties on how the mail is found and handled.

Forward Emails

If the mail service is unable to handle a message you can define an email address to forward these messages to. Just add the desired email address into the 'ForwardEmail' textbox.

Note:
You will need to configure a SMTP mail server before this functionality can be used.

SSL

The mail service can be configured to connect to the email server using an SSL connection. To do this select the appropriate SSL connection in the 'Use SSL' select list. If you do not want JIRA to use SSL, select 'No SSL'.

If you are using SSL, you will need to preload the IMAPS/POPS server's public key in JIRA (actually, the Java virtual machine running JIRA). See Connecting to SSL Services for information on how to do this.

Folder (IMAP Only)

For the IMAP Service it is possible to specify the folder you wish JIRA to read when scanning for messages. To do this, add the desired folder name to the 'Folder' field.

Note:
If a folder is not specified the mail service will default to 'INBOX'.

1.15.3.4. Removing a Service

To remove a service, click **Del** for that service in the **Services** section of the **Administration** tab.

1.15.3.5. Built-in Services

JIRA has some useful services out the box, which may be used as-is or modified for use in your own environment. The source of all built-in services is available and should give you good overview of how simple it is to write your own services. All built-in services are included with JIRA, and need only be configured to be used.

Export Service (source)

The Export Service is useful for periodically backing up JIRA. It exports all data from JIRA every time it is run into a directory supplied as a parameter. The export files are timestamped, thus the service can act as a backup system.

To test this service, add a service with the class **com.atlassian.jira.service.services.export.ExportService**. JIRA sets up an ExportService during the setup wizard, so you may find you already have one.

POP Service (source)

The POP service reads messages from a defined POP3 mail box, and then performs operations within JIRA based on the message contents. A MessageHandler (see below for more information) is configured for each instance of the POP Service, which determines how the message is handled.

To test this service, add a service with the class **com.atlassian.jira.service.services.pop.PopService** and configure the POP details and message handler in the properties of the service.

To make the POP service even more useful, you can connect it with the email notifications sent by JIRA. Simply set the `FROM` address in the MailListener to be the same address as the POP mailbox being monitored. This allows you to do things like reply to email notifications and have your replies added as comments on the issue.

IMAP Service (source)

Similar to POP service, except that it reads from an IMAP mailbox's "INBOX" instead. Like the POP service, it removes emails after reading.

To test this service, add a service with the class **com.atlassian.jira.service.services.imap.ImapService** and configure the IMAP details and message handler in the properties of the service.

File Service (source)

The File service is very much like the POP service above, except that instead of reading emails from a POP mailbox, they are read from a directory on disk. This is useful because you do not need an anonymous POP mail box (which could be a potential security risk) to use it. Instead you can simply get your mail server to dump incoming email messages into a particular directory, which the File service scans periodically.

The setup of this service is identical to the POP Service above, except that the service class is **com.atlassian.jira.service.services.file.FileService** and the service is configured with the directory to watch instead of the POP mailbox details. Both File and POP services can use the same MessageHandlers.

Jelly Service (source)

Jelly is a scripting language which allows tasks in JIRA to be automated. The Jelly Service periodically runs a Jelly script. For example, you could use this to periodically run a search request, loop through the results and add a comment, change the issue state (see the Jelly examples. If you're considering writing a custom service, often a periodically invoked Jelly script may be an easier alternative.

JIRA Service classes must all extend com.atlassian.jira.service.JiraService. Most do so by extending com.atlassian.jira.service.AbstractService or some more specialised subclass.

1.15.3.6. Message Handlers

POPService, IMAPService and FileService above use MessageHandlers (API doc) to perform operations within JIRA based on the format of incoming email messages.

You can design your own MessageHandlers to integrate JIRA with your own processes, and plug them into any of these three services via the **Administration** interface. (Please also see Adding your own email handling classes.)

MessageHandlers are configured with a comma-separated list of name-value pairs, known as the **handler parameters**.

There are a number of default message handlers that ship with JIRA, described below:

CreateIssueHandler

com.atlassian.jira.service.util.handler.**CreateIssueHandler** | API doc | Source

This message handler creates a new issue for each incoming message.

Parameter	Meaning	Value
project	Default project where new issues are created.	Project key, e.g. `JRA`
issuetype	Default type for new issues.	Integer representing issue type: 1. Bug 2. New Feature 3. Task 4. Improvement
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA.	`true` or `false`

	Note: this parameter is not compatible with `reporterusername`. If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:) are processed. **Note:** if this parameter is set, emails addressed to anyone other than the specified recipient will be **deleted**. Please note, setting this parameter also suppresses the generation of the addressee as watcher.	E.g. `issues@mycompany.com`
ccassignee	If an email has a Cc address listing an assignable user already present in JIRA, by default JIRA will **assign** the issue to that user. In JIRA 3.1 and above, if you do not want this behaviour, set **ccassignee** to **false**. By default (if not supplied), `ccassignee` is set to true.	`true` or `false`
ccwatcher	If an email has a **To**, **Cc** or **Bcc** address listing a user already present in JIRA, it is possible to automatically add that user to an issue's watchers list, by setting the `ccwatcher` parameter to `true`. Please note that users created by the `createusers` parameter cannot be added to an issue's watchers list by the `ccwatcher` parameter. Users must already exist in JIRA's userbase, and must have an	`true` or `false`

	email address. By default (if not supplied), `ccwatcher` is set to false.	
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: • `ignore` - Ignore the email and do nothing • `forward` - Forward the email to the address set in the "Forward Email" text field • `delete` - Delete the email permanently Any other values are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

Table 1: CreateIssueHandler parameters

CreateOrCommentHandler

com.atlassian.jira.service.util.handler.**CreateOrCommentHandler** | API doc | Source

This message handler creates a new issue, or adds a comment to an existing issue. If the subject contains an issue key, the message is added as a comment to that issue. If no issue key is found, a new issue is created in the default project.

Parameter	Meaning	Value
project	Default project where new issues are created. The `project` parameter is only relevant for issue creation, not for issue commenting. If an email contains an issue key in the email subject, and that issue exists in the JIRA instance, the handler will add the email as a comment on the issue, regardless of which project the issue is in.	Project key, e.g.`JRA`
issuetype	Default type for new issues.	Integer representing issue type: 1. Bug 2. New Feature 3. Task 4. Improvement
stripquotes	If set (to anything), quoted text is removed from comments.	(anything)
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the	`true` or `false`

	message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername`. If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	E.g. `issues@mycompany.com`
ccassignee	If an email has a Cc address listing an assignable user already present in JIRA, by default JIRA will **assign** the issue to that user. In JIRA 3.1 and above, if you do not want this behaviour, set **ccassignee** to **false**. By default (if not supplied), `ccassignee` is set to true.	`true` or `false`
ccwatcher	If an email has a **To**, **Cc** or **Bcc** address listing a user already present in JIRA, it is possible to automatically add that user to an issue's watchers list, by setting the `ccwatcher` parameter to `true`. Please note that users created by the `createusers` parameter cannot be added to an issue's watchers list by the `ccwatcher` parameter. Users must already exist in JIRA's userbase, and must have an email address. By default (if not supplied),	`true` or `false`

	`ccwatcher` is set to false.	
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` - Ignore the email and do nothing 2. `forward` - Forward the email to the address set in the "Forward Email" text field 3. `delete` - Delete the email permanently Any other value's are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

Table 1: CreateOrCommentHandler parameters

FullCommentHandler

com.atlassian.jira.service.util.handler.**FullCommentHandler** | API doc | Source

This message handler creates a comment based on the entire body of the email received.

The issue to use is chosen from the first issue key found in the email subject. The author of the comment is taken from the from address of the email.

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername`. If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they	`true` or `false`

	will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	E.g. `issues@mycompany.com`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` - Ignore the email and do nothing 2. `forward` - Forward the email to the address set in the "Forward Email" text field 3. `delete` - Delete the email permanently Any other value's are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

Table 1: FullCommentHandler parameters

NonQuotedCommentHandler

com.atlassian.jira.service.util.handler.**NonQuotedCommentHandler** | API doc | Source

This message handler also creates a comment, but only uses the "non quoted" lines of the email body. A quoted line is any line that starts with">" or "|".

The issue to use is chosen from the first issue key found in the email subject. The author of the comment is taken from the from address of the email.

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername`. If `createusers` is set to true, and the `reporterusername`	`true` or `false`

	is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	E.g. `issues@mycompany.com`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` - Ignore the email and do nothing 2. `forward` - Forward the email to the address set in the "Forward Email" text field 3. `delete` - Delete the email permanently Any other value's are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

Table 1: NonQuotedCommentHandler parameters

RegexCommentHandler

com.atlassian.jira.service.util.handler.**RegexCommentHandler** | API doc | Source

This message handler creates a comment from an email body - but ignores any part of the email body past a specified marker or separator. For mail systems like Lotus Notes and Outlook, emails are separated from the quoted email by some predictable text like "---- Original Message ----" or "Extranet\n email.address/DOM/REG/CONT/CORP@CORPMAIL" The RegexCommentHandler can take any valid regular expression — and in fact filter quoted mails from various different mail systems simultaneously.

- If the pattern is found, it returns the text before the 1st match - and discards the rest of the email body
- If the pattern is not found, it returns the email body unchanged
- If the regex is not specified, it returns the email body unchanged
- If there is any error (i.e. regex expression error), it returns the email body unchanged

Parameter	Meaning	Value

reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername`. If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied), `createusers` is set to false.	`true` or `false`
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	E.g. `issues@mycompany.com`
splitregex	Regular expression matching the text separating the mail from any previous mails. **Note that the regexp must begin and end with a delimiter character, typically '/'.**. Also note that currently, commas are not allowed in regexps, as commas are used to separate handler parameters and there is not (as yet) an escape syntax.	E.g. `/----\s*Original Message\s*----/` or `/_____*/`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on	`ignore` or `forward` or `delete`

	the value of this option: 1. `ignore` - Ignore the email and do nothing 2. `forward` - Forward the email to the address set in the "Forward Email" text field 3. `delete` - Delete the email permanently Any other value's are invalid, and the handler will perform normally.	

Table 1: RegexCommentHandler parameters

CVSLogHandler

com.atlassian.jira.service.util.handler.**CVSLogHandler** | API doc | Source

This message handler parses CVS Log messages and adds the relevant sections as a comment.

The comment is added to *any* issue whose key is mentioned in the CVS commit message.

For instance if you commit to CVS with the message "*This commit fixes JRA-57 and JRA-58.*", a comment will be added to issues JRA-57 and JRA-58.

The body of the comment includes the commit message entered by the developer and the files involved in the commit.

Warning:
JIRA no longer uses CVSLogHandler for its CVS integration - this service is kept here purely as an example.

To use this message handler, setup your CVS server to email commit messages using something like SyncMail.

Parameter	Meaning	Value
reporterusername	Username of default reporter, if sender not recognised.	JIRA username, e.g. `admin`
createusers	If a message comes from unrecognised address, create a new JIRA user with the user name and email address set to the 'From' address of the message. The password for the new user is randomly generated, and an email is sent to the new user informing them about their new account in JIRA. **Note:** this parameter is not compatible with `reporterusername`. If `createusers` is set to true, and the `reporterusername` is also supplied, users will be created if they cannot be found using the from addresses of the received messages. That is, the `reporterusername` will be ignored. By default (if not supplied),	`true` or `false`

	`createusers` is set to false.	
notifyusers	This parameter is only used if **createusers** is set to true. If **notifyusers** is set to false they will not receive a notification that their account has been created via email. The default value is true to preserve the behaviour before this parameter was added. By default (if not supplied), `notifyusers` is set to true.	`true` or `false`
catchemail	If set, only emails to the specified recipient (To:, Cc:, Bcc:) are processed.	E.g. `issues@mycompany.com`
bulk	This option only affects emails with the 'Precedence: bulk' or emails with an 'Auto-Submitted' header that is not set to "no". One of the following actions will be performed, depending on the value of this option: 1. `ignore` - Ignore the email and do nothing 2. `forward` - Forward the email to the address set in the "Forward Email" text field 3. `delete` - Delete the email permanently Any other value's are invalid, and the handler will perform normally.	`ignore` or `forward` or `delete`

Table 1: CVSLogHandler parameters

1.15.3.7. Message Handlers and Events

The message handlers will dispatch an JIRA event depending on the actions they perform. For more information on JIRA events please refer to the Notification Schemes section.

The CreateIssueHandler will dispatch an 'Issue Created' event whenever it creates a new issue.

The CreateOrCommentHandler will dispatch one of 'Issue Created', 'Issue Commented' or 'Issue Updated' events:

- 'Issue Created' event is dispatched whenever it creates a new issue.
- 'Issue Commented' event is dispatched if the issue already exists and a comment is added only.
- 'Issue Updated' event is dispatched if the issue already exists and a comment with attachment(s) is added.

Each of the comment handlers (FullCommentHandler, NonQuotedCommentHandler, RegexCommentHandler and CVSLogHandler) will dispatch the 'Issue Commented' event if the message only contains a comment. However, if the message contains an attachment as well, it will dispatch the 'Issue Updated' event instead.

1.15.3.8. Building Services from source

To build any of the linked sample code:

- If you have JIRA Standalone, download to the **external-source/src/** directory, and read the instructions in **external-source/README.txt**) for build instructions.
- If you have the JIRA WAR/Webapp distribution, download the source to the **src/** directory. The code will be compiled into the webapp when `build.sh/build.bat` is run.

1.15.4. Jelly Tags

Jelly is a scripting and templating language from Apache's Jakarta project. It is similar to Ant, in that scripts are XML, and each tag maps to a Java class, but has a more sophisticated internal pipeline model for tag interaction, much like JSP taglibs. See the Jelly website for more details.

JIRA comes with a number of Jelly tags implementing core operations in JIRA. This provides a scriptable interface to JIRA. There are many possible uses for JIRA Jelly tags, the most common being importing data into JIRA from other systems, and automating common administrative tasks (see the examples below).

1.15.4.1. Enabling Jelly

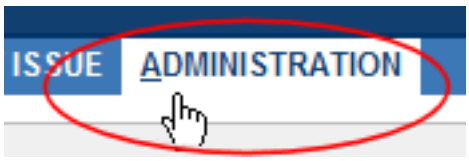
JIRA's Jelly support is disabled by default, as it is a potential security hazard. To enable Jelly support, set the **jira.jelly.on** system property when starting your app server. System properties are set with parameters to the `java` command, e.g.: `java -Djira.jelly.on=true ...`.

How to set this property depends on your application server. For example, with Tomcat (JIRA standalone distribution), set the environment variable **JAVA_OPTS=-Djira.jelly.on=true**, or when running JIRA Standalone as a service, set the service JVM parameter.

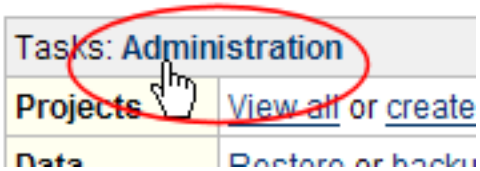
1.15.4.2. Running a Jelly script

To run a Jelly script once:

1. Log in as a user with the '**JIRA System Administrators**' global permission.
2. Bring up the administration page by clicking either the '**Administration**' link on the top bar or the title of the Administration box on the dashboard:



Link to Administration section



Link to Administration section

3. Under the '**Options & Settings**' sub-menu in the left-hand navigation column, click the '**Jelly Runner**' link.
4. Paste your Jelly script into the text area.

To run a Jelly script periodically:

Configure a service with the following class: `com.atlassian.jira.jelly.service.JellyService`.

1.15.4.3. Writing a Jelly script

Scripts are generally of the form:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<!--
    Add your own Jelly XML here
-->
</JiraJelly>
```

JIRA Enterprise has a few extra tags available; to use them, the outer tag should instead be:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
<!--
    Add your own Jelly XML here
-->
</JiraJelly>
```

In addition to the JIRA tags, you can use tags from the email, http, soap, sql, and core Jelly taglibs. More can be added by the user if necessary.

Many of JIRA's Jelly tags set *context variables*, so subsequent tags can refer to their output by dereferencing the context variable (e.g. `${jira.new.username}`). Other tags let you explicitly specify the name of a variable to store some output in. E.g., <jira:CreateUser> has `issueKeyVar` and `issueIdVar` parameters:

```
<jira:CreateIssue project-key="TP" summary="Issue One" issueKeyVar="issuekey" issueIdVa
          Raised issue ${issuekey} with id ${issueid}
```

Note that the variable is only set **after** the tag is closed, not inside the tag.

Warning:

Due to this variable interpolation, if your text contains anything of the form `${something}`, you need to escape this as `$${something}` to prevent the 'variable' being expanded to a blank string.

The list of currently available tags:

- jira:AddComment
- jira:AddComponent
- jira:SelectComponentAssignees
- jira:AddUserToGroup
- jira:AddVersion
- jira:CreateGroup
- jira:AssignIssue
- jira:CreateIssue
- jira:LinkIssue
- jira:TransitionWorkflow
- jira:CreateProject
- jira:CreateUser
- jira:RemoveUser
- jira:CreatePermissionScheme
- jira:AddPermission
- jira:Login
- jira:CreateCustomField
- jira:AddFieldToScreen
- jira:AttachFile
- jira:RunSearchRequest
- jira:AddActorsToDefaultProjectRole
- jira:AddActorsToProjectRole
- jira:CreateProjectRole
- jira:DeleteProjectRole
- jira:GetDefaultRoleActors

- jira:GetProjectRole
- jira:GetProjectRoleActors
- jira:IsProjectRoleNameUnique
- jira:RemoveActorsFromProjectRole
- jira:RemoveActorsFromDefaultProjectRole
- jira:UpdateProjectRole

1.15.4.4. jira:AddComment

This function adds a comment to an Issue.

Attribute name	Type	Default value	Description
issue-key	string		The issue to add the comment to (required).
commenter	string	Currently logged in user	Username of the user to make the comment (Must have browse and comment permissions).
comment	string		Comment to be added to the issue (required).
groupLevel	string	none	Name of group that can see this comment. NOTE: If this is specified you can not specify the roleLevel parameter.
roleLevel	string	none	Name or Id of Project Role that can see this comment. NOTE: If this is specified you can not specify the groupLevel parameter.
created	string	Current Date/Time	Date/Time the Comment was created in format yyyy-MM-dd hh:mm:ss.0
updated	string	Current Date/Time	Date/Time the Comment was last updated in format yyyy-MM-dd hh:mm:ss.0. This can be used if you are trying to import a comment with specific pre-existing values.
editedBy	string	Currently logged in user	Username of the user who last updated the comment. This can be used if you are trying to import a comment with specific pre-existing values.

Table 1: Attributes

Examples

Create comment

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddComment comment="Issue comment" issue-key="ABC-1"
groupLevel="admin-group"/>
</JiraJelly>
```

Create Issue and Comment

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreateIssue project-key="TP" issueType="Bug" summary="Issue summary" issueKeyVa
  <jira:AddComment issue-key="${key}" comment="A comment on ${key}"/>
</JiraJelly>
```

1.15.4.5. jira:AddComponent

Adds a component to a project.

Attribute name	Type	Default value	Description
project-key	string		The key of the project you want to add the component to (not required if nested inside a jira:CreateProject tag).
name	string		Name of the component (required).
description	string		Description of the component.
componentLead	string		For JIRA Enterprise only. The username of the Component's lead. Leave blank for no lead.

Table 1: Attributes

Examples

Create Component

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddComponent project-key="ABC" name="Comp 1" description="Comp 1 description"/>
</JiraJelly>
```

Create Component in a Project

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="logged-in-user">
<jira:AddComponent name="Comp 1"/>
</jira:CreateProject>
</JiraJelly>
```

Create Component with a Component Lead (Enterprise-only)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddComponent project-key="ABC" name="Comp 1" description="Comp 1 with lead" componen
</JiraJelly>
```

1.15.4.6. jira:SelectComponentAssignees

Selects the default assignees for newly created issues of the component. This tag is only available in JIRA Enterprise.

Attribute name	Type	Default value	Description
project-key	string		The key of the project you want to add the component to (required).
componentName	string		Name of the component (required).
assigneeType	string		Default assignee type (required).

As
projectDefault
componentLead
projectLead
unassigned

Table 1: Attributes

Examples

Select a Component Assignee

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
<jira:SelectComponentAssignees project-key="ABC" componentName="Comp 1" assigneeType="comp
</JiraJelly>
```

1.15.4.7. jira:AddUserToGroup

Makes a user a member of a Group. Adds the username and/or group name into the context if specified.

Attribute name	Type	Default value	Description
username	string		Username to add to Group (required if not in a jira:CreateUser tag).
group-name	string		Group to add User to (required if not in a jira:CreateGroup tag). Note: if the group has the 'JIRA System Administrators' global permission, and the logged-in user does not, an error message will be displayed and

			the operation will not succeed.

Table 1: Attributes

Context Variable	Type	Description
jelly.new.username	string	Username is set in the context if specified in the tag.
jelly.group.name	string	Group name is set in the context if specified in the tag.

Table 2: Context Variables

Examples

Add User to Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddUserToGroup username="new-user" group-name="new-group"/>
</JiraJelly>
```

Add New User to Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateUser username="new-user" password="password" confirm="password"
fullname="Full name" email="test@test.com">
<jira:AddUserToGroup group-name="new-group"/>
</jira:CreateUser>
</JiraJelly>
```

Add User to New Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateGroup group-name="new-group">
<jira:AddUserToGroup username="new-user"/>
</jira:CreateGroup>
</JiraJelly>
```

Add New User to New Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateUser username="new-user" password="password" confirm="password"
fullname="Full name" email="test@test.com"/>
<jira:CreateGroup group-name="new-group">
<jira:AddUserToGroup/>
</jira:CreateGroup>
</jira:CreateUser>
</JiraJelly>
```

1.15.4.8. jira:AddVersion

Adds a version to a project.

Attribute name	Type	Default value	Description
project-key	string		The key of the project you want to add the component too (not required if nested inside a jira:CreateProject tag).
name	string		Name of the version

			(required).
description	string		The description of the version.
releaseDate	string		The release date of the version.
schedule	string		Schedule of the version.

Table 1: Attributes

Examples

Create a Version

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:AddVersion project-key="ABC" name="Ver 1"/>
</JiraJelly>
```

Create a Version in a Project

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="logged-in-user">
<jira:AddVersion name="Ver 1"/>
</jira:CreateProject>
</JiraJelly>
```

1.15.4.9. jira:CreateGroup

Creates a Group in JIRA.

Attribute name	Type	Default value	Description
group-name	string		Name of group to create (required).

Table 1: Attributes

Context Variable	Type	Description
jelly.group.name	string	Name of group being created.

Table 2: Context Variables

Examples

Create Group

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateGroup group-name="new-group"/>
</JiraJelly>
```

1.15.4.10. jira:AssignIssue

Assigns an issue to a user.

Attribute name	Type	Default value	Description
key	string		Key of the issue to assign.
assignee	string		User to assign issue to.

Table 1: Attributes

Examples

Create and assign issue

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreateIssue project-key="TST" summary="My Issue summary" issueKeyVar="keyvar"/>
  <jira:AssignIssue key="${keyvar}" assignee="testuser"/>
</JiraJelly>
```

1.15.4.11. jira:CreateIssue

This tag creates a new issue in JIRA and places the issue id in the context.

Attribute name	Type	Default value	Description
project-key	string		Key of the project to add the issue to (required if not nested in a jira:CreateProject tag).
issueType	string	First issue type	The string name of the Issue Type this issue should be created for (e.g. Bug).
summary	string		Summary of the issue being created (required).
priority	string	First priority	The string name of the Priority (e.g. Major).
components	string		The string name of the Component.
versions	string		The string name of the Affected Version.
fixVersions	string		The string name of the Fix For Version.
assignee	string		The username of the user to assign this issue to (logged in user requires the assign issue permission and user specified requires the assignable permission). Set to "-1" for Automatic assignment.
reporter	string	(see description)	The username of the user who is reporting this issue. The user is logged in and then the issue is created. The user is logged out again when the Create Issue tag closes. If the logged in user does

			not have Modify Reporter privilege, then the default value of this attribute is the username of the logged in user. If, however, the logged in user does have Modify Reporter privilege, there is not a default value, and this attribute is mandatory. See JRA-12984 for further explanation. (Broken? See JRA-5620.)
environment	string		Description of the environment.
description	string		Detailed description of the issue.

duedate	string		Due date of the issue. The format required is the current JIRA date format.
			
			Note: As the default JIRA date format is locale-specific (e.g. 12/Jan/05), you may wish to use the yyyy-mm-dd ISO format instead. To do this, set the following in WEB-INF/classes/jira-application.properties:
			`jira.date.picker.java.format = yy`
			`jira.date.picker.javascript.forma`

created	string	Current Date/Time	Date/Time the Issue was created in format yyyy-MM-dd hh:mm:ss.0
updated	string	Current Date/Time	Date/Time the Issue was updated in format yyyy-MM-dd hh:mm:ss.0
issueIdVar	string		The name of the variable to place the ID of the new Issue.
issueKeyVar	string		The name of the variable to place the Key of the new Issue.
duplicateSummary	string		Setting this attribute to 'ignore' will allow Issue with the same summary to be created.

security-level	string		Enterprise-only - sets the security level of an issue. Value is the name of a level, e.g. 'Secret'.

Table 1: Attributes

Examples

Create Issue

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateIssue project-key="ABC" assignee="-1" summary="Issue summary">
<!-- other jelly tags -->
</jira:CreateIssue>
</JiraJelly>
```

Create Issue from Project

This example is more complicated as a permission scheme is required for the project before an issue can be created.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="logged-in-user">
<jira:CreatePermissionScheme name="admin-scheme">
<jira:AddPermission permissions="Assignable,Browse,Create,Assign"
type="group"/>
<jira:SelectProjectScheme/>
</jira:CreatePermissionScheme>
<jira:CreateIssue summary="Issue summary">
<!-- other jelly tags -->
</jira:CreateIssue>
</jira:CreateProject>
</JiraJelly>
```

Create Issue with Custom Field values

Use the subtag `jira:AddCustomFieldValue`

Attribute Name	Type	Description
id	long	ID of the custom field with the customfield_ prefix
value	string	string representation of the custom field value. Note that this may be different to the displayed value (e.g. The project picker uses the project id as the String value but displays the project name)
key	string	Key is used for multi-dimensional data. Currently, only Cascading selects supports its use. Omit to specify the value of parent, use "1" as the value for child
name	String	**deprecated** Name of the custom field.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateIssue project-key="ABC" summary="Issue summary">
<jira:AddCustomFieldValue id="customfield_10000" value="field value"/>
<jira:AddCustomFieldValue name="Environment Select list" value="Windows XP"/>

<!-- For Cascading Selects : Note also that the value for cascading selects is the optionI
<jira:AddCustomFieldValue id="customfield_10001" value="Parent Option Id" />
<jira:AddCustomFieldValue id="customfield_10001" value="Child Option Id" key="1" />

<!-- For Version Pickers and Single Version Pickers : Note also that the value for version
<jira:AddCustomFieldValue id="customfield_10002" value="Version Id"/>

<!-- For Multi Selects -->
<jira:AddCustomFieldValue id="customfield_10003" value="Value 1" />
<jira:AddCustomFieldValue id="customfield_10003" value="Value 2" />

<!-- For Multi User Pickers : Note also that the value for multi user pickers is a comma s
<jira:AddCustomFieldValue id="customfield_10004" value="User 1,User 2" />

</jira:CreateIssue>
</JiraJelly>
```

Note:

Using the name attribute has been **deprecated**. While it will work in 3.0 its use is discouraged.

1.15.4.12. jira:LinkIssue

This tag creates a link from one issue to another issue.

Attribute name	Type	Default value	Description
key	string		The key of the issue to link from (origin of link - required)
linkKey	string		The key of the issue to link to (destination of link - required)
linkDesc	string		linkDesc is taken from the 'Inward Description' or the 'Outward Description' of the link. (required)

Table 1: Attributes

Examples

Create a Link between two existing issues

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:LinkIssue key="TST-1" linkKey="TST-2" linkDesc="duplicates"/>
</JiraJelly>
```

Create two issues and link them

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateIssue project-key="HSP" assignee="-1" summary="Issue summary 1" reporter="admi
<jira:CreateIssue project-key="NDT" assignee="-1" summary="Issue summary 2" reporter="admi
<jira:LinkIssue key="${issuekey1}" linkKey="${issuekey2}" linkDesc="duplicates"/>
</JiraJelly>
```

1.15.4.13. jira:TransitionWorkflow

Warning:
Broken in 3.3 and 3.3.1 - see JRA-7690

This tag executes a workflow transition on an issue.

Please keep in mind that if you are specifying field attribute/value pairs in your Jelly tag then these fields MUST be on the associted workflow transition screen. If the field is not on the screen then the value will not be set on the issue. For example, if you want to set the resolution attribute in your Jelly XML then your transition MUST have a screen associated with it that includes the resolution field on that screen.

Attribute name	Type	Default value	Description
user	string	Currently logged in user	Username of the user to execute the workflow transition. The user needs to have the adequate permissions to execute the transition. Please note that the permissions required also depend on the fields that are updated during the transition. (See other attributes below).
key	string		The key of the issue to execute the transition on.
workflowAction	string		The id or name of the workflow transition to execute. If the argument can be converted to a number it is assumed to be an id of the transition. Otherwise it is assumed to be a name.
resolution	string		The id or name of the resolution to set on the issue during the transition. Please note that the transition must expect the resolution to be updated, otherwise an error is generated if this attribute is supplied. If the argument can be converted to a number it is assumed to be an id of the resolution. Otherwise it is assumed to be a name.
assignee	string		

			The username of the user to assign an issue to during the transition. The "user" executing the transition must have permissions to assign issues if this attribute is supplied. Please note that the transition must expect the assignee to be updated, otherwise an error is generated if this attribute is supplied. Use value "-automatic-" to let JIRA assign the issue to the default assignee.
fixVersions	string		A comma separated list of version ids or names to set as "fix for" versions during the transition. The "user" executing the transition must have permissions to set "fix for" versions if this attribute is supplied. Please note that the transition must expect the "fix for" versions to be updated, otherwise an error is generated if this attribute is supplied. If a value in the provided comma separated list can be converted to a number it is assumed to be an id of a version. Otherwise it is assumed to be a name.
comment	string		The comment to add to the issue during the transition. The "user" executing the transition must have permissions to add comments and the transition must be expecting comments to be added during its execution for the comment to be added successfully.
groupLevel	string		The level for the comment. The level must be a name of a group the user is a member of. NOTE: If

			this is specified you can not specify the roleLevel parameter.
roleLevel	string		Name or Id of Project Role that can see this comment. NOTE: If this is specified you can not specify the groupLevel parameter.

Table 1: Attributes

Examples

Execute Workflow Transition

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
                    <jira:TransitionWorkflow key="TST-6" user="testuser" workflowAction=
                    resolution="fixed" fixVersions="version 1,version 3" assignee="-aut
                    comment="Test comment" groupLevel="jira-developers" />
</JiraJelly>
```

1.15.4.14. jira:CreateProject

This tag creates a new project in JIRA and places the project id in the context.

Attribute name	Type	Default value	Description
key	string		The project key used to create Issue Keys (required).
name	string		The name of the project (required).
lead	string		The username of the user that is the project lead (required).
url	string		The URL of the site for this project.
description	string		The description of this project.

Table 1: Attributes

Context Variable	Type	Description
jelly.project.id	string	Id of the Project that was created.
jelly.project.key	string	Key of the Project that was created.

Table 2: Context Variables

Examples

Create Project

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateProject key="ABC" name="A Project" lead="a-user">
<!-- other jelly tags -->
```

```
</jira:CreateProject>
</JiraJelly>
```

1.15.4.15. jira:CreateUser

Creates a user in JIRA and places their username in the context.

Attribute name	Type	Default value	Description
username	string		Username of the user being created (required).
password	string		User's password. If the password field is left blank, a random password will be auto-generated.
confirm	string		Confirmation of users password (required).
fullname	string		Descriptive name of the user (required).
email	string		Email address of the user (required).
sendEmail	boolean	false	If provided, specifies whether to send a confirmation email.

Table 1: Attributes

Context Variable	Type	Description
jelly.new.username	string	Username of the user being created.

Table 2: Context Variables

Examples

Create User

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateUser username="new-user" password="password" confirm="password"
fullname="Full name" email="test@test.com"/>
</JiraJelly>
```

1.15.4.16. jira:RemoveUser

Removes an existing JIRA user by their username

Attribute name	Type	Default value	Description
name	string		Username of the user to remove (required).

Table 1: Attributes

Examples

Remove User

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:RemoveUser name="existing-user"/>
</JiraJelly>
```

1.15.4.17. jira:CreatePermissionScheme

Creates a Permission Scheme

Attribute name	Type	Default value	Description
name	required string		Name of the permission scheme.
description	string		Permission scheme description.

Table 1: Attributes

Context Variable	Type	Description
jelly.permission.scheme.id	string	Id of the created permission scheme

Table 2: Context Variables

1.15.4.18. jira:AddPermission

Grants permissions within a permission scheme. Often nested within a CreatePermissionScheme tag.

Attribute name	Type	Default value	Description
schemeId	string		If not nested in a CreatePermissionScheme tag, specifies the scheme Id to add the permission to (0 is the default permission scheme).
permissions	required string		A comma-separated list of permissions to grant:

String	
Project	
Browse	
Create	
Edit	
ScheduleIssue	
Move	
Assign	
Assignable	
Resolve	
Close	
ModifyReporter	
Comment	

			CommentEditAll CommentEditOwn CommentDeleteAll CommentDeleteOwn Delete Work WorklogEditAll WorklogEditOwn WorklogDeleteOwn WorklogDeleteAll Link Attach AttachDeleteAll AttachDeleteOwn ViewVersionControl ViewVotersAndWatchers ManageWatcherList SetSecurity
type	string		Type of recipient for the permission group projectrole user lead assignee reporter userCF groupCF
group	string		If type is 'group' (or type is unspecified), specifies the group name to grant permissions to.
projectroleid	int		If type is 'projectrole', specifies the id of the projectrole to grant permissions to.

user	string		If type is 'user', specifies the user name to grant permissions to.
userCF	string		If type is 'userCF', specifies the id of a User custom field, e.g. 'customfield_10000', identifying the user to be granted the permission.
groupCF	string		If type is 'groupCF', specifies the id of a group-selecting custom field (e.g. a select-list with group names as values) whose members should be granted this permission. E.g. 'customfield_10000'.

Table 1: Attributes

Examples

Grant permissions to jira-users and jira-developers in a new permission scheme

(See also the example scripts)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:CreatePermissionScheme name="New Permission Scheme">
    <jira:AddPermission group="jira-users" permissions="Browse,Create,Comment,Attach" t
    <jira:AddPermission group="jira-developers" permissions="Move,Assignable,Link,ViewV
  </jira:CreatePermissionScheme>
</JiraJelly>
```

Grant issue reporters the ability to edit/delete their own issues, in a new permission scheme (Enterprise-only)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
  <jira:CreatePermissionScheme name="New Permission Scheme">
    <jira:AddPermission type="reporter" permissions="Delete, Edit"/>
  </jira:CreatePermissionScheme>
</JiraJelly>
```

Make projects using default permission scheme visible to certain users (Enterprise-only)

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib">
  <jira:AddPermission schemeId="0" permissions="Browse" type="user" user="johnc"/>
  <jira:AddPermission schemeId="0" permissions="Browse" type="user" user="ebf"/>
</JiraJelly>
```

Granting a group selector custom field's members the ability to assign/be assigned the issue.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:AddPermission schemeId="10164" type="groupCF" groupCF="customfield_10000"
                      permissions="Assign,Assignable" />
</JiraJelly>
```

1.15.4.19. jira:Login

This tag logs a user into JIRA using the username and password provided. Use this tag when you

are running the Jelly script in a manner in which you are not logged in (for example, if you are running a JellyService instead of using the Jelly Runner), or if you want to run the Jelly script as a different user to the one you are logged in as.

Attribute name	Type	Default value	Description
username	string		Username of the user to log in.
password	string		Password of the user to log in.

Table 1: Attributes

Context Variable	Type	Description
jelly.user	User	User logged in.
jelly.username	string	Username of the User logged in.

Table 2: Context Variables

Examples

Login a user in with username and password and set in context

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:Login username="misc-user" password="password">
<!-- other jelly tags -->
</jira:Login>
</JiraJelly>
```

1.15.4.20. jira:CreateCustomField

The tag creates a new Custom Field. Only System custom fields can be added with Jelly tags.

Attribute name	Type	Default value	Description
fieldType	string		Field type as appears as the key in the plugin descriptor
fieldScope	string		One of global, project or issuetype
fieldName	string		Name of custom field
projectKey	string		Key of the related project. Only valid for scope "project"
issueType	string		Issue type. Only valid for scope "issuetype"
description	string		Description of the field to be displayed when adding a value
searcher	string		A valid related custom field searcher
customFieldIdVar	string		The name of the variable to place the new custom field.

Table 1: Attributes

Examples

Create Cascading Custom Field

`jira:AddCustomFieldSelectValue` subtag can be used to add values for select lists. They can also be nested for Cascading Select Lists.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateCustomField fieldType="cascadingselect"
fieldScope="issuetype"
fieldName="Issue cascadingselect Bug"
issueType="Bug"
description="Bank have requested Y2K fixes to be sent as an EBF."
searcher="cascadingselectsearcher"
>
<jira:AddCustomFieldSelectValue value="Parent 1" />
<jira:AddCustomFieldSelectValue value="Parent 2">
<jira:AddCustomFieldSelectValue value="Child 1" />
<jira:AddCustomFieldSelectValue value="Child 2" />
</jira:AddCustomFieldSelectValue>
<jira:AddCustomFieldSelectValue value="Parent 3" />
</jira:CreateCustomField>
</JiraJelly>
```

1.15.4.21. jira:AddFieldToScreen

Adds a field to a specific tab on a screen. Can also specify in which position to insert the field.

Attribute name	Type	Default value	Description
ffldId	string		Field ID of the field to add (required). e.g. "description", "duedate", etc.
screen	string		Screen ID or Name (required). e.g. "1" or "Default Screen".
tab	string	0	Tab ID or Name. e.g. "0" or "Field Tab".
fieldPosition	int	last position	Position to insert the field into. Range of values is from 1 to the number of fields on the screen.

Table 1: Attributes

Examples

Add Fields to a Screen

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<!-- Adds 'description' field to the 'Field Tab' on 'Default Screen' -->
<jira:AddFieldToScreen fieldId="description" screen="Default Screen" tab="Field Tab"/>

<!-- Adds 'duedate' field to same screen as above. duedate is inserted in position 1 --
```

```
<jira:AddFieldToScreen fieldId="duedate" screen="1" tab="0" fieldPosition="1"/>
</JiraJelly>
```

Create a new Customfield and Add it to a Screen

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
<jira:CreateCustomField fieldType="cascadingselect"
fieldScope="issuetype"
fieldName="Issue cascadingselect Bug"
issueType="Bug"
description="Bank have requested Y2K fixes to be sent as an EBF."
searcher="cascadingselectsearcher"
customFieldIdVar="customField"
>
<jira:AddCustomFieldSelectValue value="Parent 1" />
<jira:AddCustomFieldSelectValue value="Parent 2">
<jira:AddCustomFieldSelectValue value="Child 1" />
<jira:AddCustomFieldSelectValue value="Child 2" />
</jira:AddCustomFieldSelectValue>
<jira:AddCustomFieldSelectValue value="Parent 3" />
</jira:CreateCustomField>

<jira:AddFieldToScreen screen="Default Screen" fieldId="${customField.getId()}"/>
</JiraJelly>
```

1.15.4.22. jira:AttachFile

Attaches a file to an issue.

Attribute name	Type	Default value	Description
key	string		Key of the issue to attach the file to. (Required)
filepath	string		Path (on the server) of the file to attach. (Required)
option	string	add	Behaviour when a file with same name is already attached. (Optional). The options are: • **skip** - do not attach file if a file with this name is already attached. • **override** - overwrite existing attached file • **add** - add the file as another attachment
created	string	Current Date/Time	Date/Time the attachment was created, in format yyyy-MM-dd hh:mm:ss.0 (Optional)

Table 1: Attributes

Examples

Adding an attachment

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
  <jira:AttachFile key="TST-1" filepath="/tmp/somefile" option="override"/>
</JiraJelly>
```

1.15.4.23. jira:RunSearchRequest

This tag runs a search request against JIRA using a predefined filter.

Note: This tag will return a *GenericValue* for each issue which matches the search request. A GenericValue consists of key-value pairs, e.g.:

```
[GenericEntity:Issue][created,2007-11-01 15:51:25.0]
[summary,Testing]
[component,null]
[workflowId,12530]
[timeoriginalestimate,null]
[fixfor,null][type,2]
[timespent,null]
[environment,Windows]
[resolution,null]
[status,1]
[updated,2007-11-01 15:51:25.0]
[timeestimate,null]
[id,11540]
[key,TSTA-5]
[duedate,null]
[description,Test]
[project,10063]
[reporter,admin]
[security,null]
[votes,0]
[assignee,null]
[priority,3]
```

To retrieve a value, e.g. `key`, you can call `gv.getString("key")`. For full details, see the OFBiz GenericValue API.

Attribute name	Type	Default value	Description
filterid	int		The id of the filter which will be used to run the search request.
size-var	string		The variable that will hold the number of issues returned from the search request.
var	string		The variable that will hold the issues returned from the search request.

Table 1: Attributes

Examples

Running a search request and iterating through the keys of the returned issues

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:RunSearchRequest filterid="10524" var="issues" size-var="issuecount"/>
    <core:forEach var="issue" items="${issues}">
        ${issue.key}
    </core:forEach>
</JiraJelly>
```

1.15.4.24. jira:AddActorsToDefaultProjectRole

This tag will add 'actors' to the default membership for a given project role. Actors can be defined as groups or users, i.e. you can add both users and groups to a project role.

Attribute name	Type	Default value	Description
projectroleid	int		This is the id of the project role.
actors	string		A comma delimited list of either users or groups
actortype	string		This defines the type 'actor' you are sending to the tag. Currently this field can contain either 'atlassian-user-role-actor' for users, or 'atlassian-group-role-actor' for groups.

Table 1: Attributes

Examples

Adding a list of default users or groups to a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:AddActorsToDefaultProjectRole projectroleid="1" actors="fred,admin,tom"
        actortype="atlassian-user-role-actor" />
</JiraJelly>
```

1.15.4.25. jira:AddActorsToProjectRole

This tag will add 'actors' to a given project role for a particular project. Actors can be defined as groups or users, ie you can add both users and groups to a project role.

Attribute name	Type	Default value	Description
projectroleid	int		This is the id of the project role.
actors	string		This a comma

			delimited list of either user names or group names
actortype	string		This defines the 'actor' type. Currently this field can contain either 'atlassian-user-role-actor' for users, or 'atlassian-group-role-actor' for groups.
projectkey	string		This is the key of the project you wish to add users or groups to for the specified role.

Table 1: Attributes

Examples

Adding a list of users or groups to a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:AddActorsToProjectRole projectroleid="1" actors="jira-administrators,jira-use
        projectkey="MKY" actortype="atlassian-group-role-actor" />
</JiraJelly>
```

1.15.4.26. jira:CreateProjectRole

This tag will create a project role with the given name and description.

Attribute name	Type	Default value	Description
name	string		The name for the project role you will be creating
description	string		The description for the project role you will be creating

Table 1: Attributes

Context Variable	Type	Description
jelly.role.id	Long	The id of the project role
jelly.role.name	string	The name of the project role
jelly.role.description	string	The description of the project role

Table 2: Context Variables

Examples

Creating a new project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
```

```
    <jira:CreateProjectRole name="lion-tamer" description="tames the lions">
        ${jelly.role.id} ${jelly.role.name} ${jelly.role.description}
    </jira:CreateProjectRole>
</JiraJelly>
```

1.15.4.27. jira:DeleteProjectRole

This tag will delete the project role with the given id.

Attribute name	Type	Default value	Description
projectroleid	int		The id of the project role you want to delete.
confirm	string		To delete the project role this value must be set to 'true'.

Table 1: Attributes

Examples

Deleting a project role from JIRA

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:DeleteProjectRole projectroleid="1" confirm="true" />
</JiraJelly>
```

1.15.4.28. jira:GetDefaultRoleActors

This tag will return a **ProjectRoleActors** object for a given project role for a particular project. This object carries the members of a project role, i.e. users and/or groups. To get the collection of users in this object, use the expression ${roleactors.users} where roleactors is the variable name of the object. For more information on the RoleActors object, consult the JIRA API

Attribute name	Type	Default value	Description
projectroleid	int		The id of the project role you want to query
var	string		The name of the variable you wish to have the returned role actors placed into

Table 1: Attributes

Examples

Returning a List of role actors and iterating over the users in each of these actors.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:core="jelly:core">
    <jira:GetDefaultRoleActors projectroleid="1" var="roleactors" >
        <core:forEach var="actor" items="${roleactors.users}">
            ${actor.name}
        </core:forEach>
    </jira:GetDefaultRoleActors>
</JiraJelly>
```

```
```

1.15.4.29. jira:GetProjectRole

This tag will return the project role with the given id.

Attribute name	Type	Default value	Description
projectroleid	int		The id of the project role you want
var	string		The name of the variable you wish to have the project role assigned to

Table 1: Attributes

Examples

Returning a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:GetProjectRole projectroleid="1" var="role" >
            ${role.name}
    </jira:GetProjectRole>
</JiraJelly>
```

1.15.4.30. jira:GetProjectRoleActors

This tag will return a **ProjectRoleActors** object for the given project role and project. This object is a placeholder for the internal members of a project role, i.e. users and/or groups. To get the collection of users in this object, use the expression ${roleactors.users} where roleactors is the variable name of the object. For more information on the RoleActors object, consult the JIRA API.

Attribute name	Type	Default value	Description
projectkey	string		The key of the project you want to query
projectroleid	int		The id of the project role you want to query
var	string		The name of the variable you want the returned 'role actors' object assigned to

Table 1: Attributes

Examples

Return a list of users for a given 'Role Actors' object

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:core="jelly:cor
    <jira:GetProjectRoleActors projectkey="MKY" projectroleid="1" var="roleactors" >
        <core:forEach var="actor" items="${roleactors.users}">
```

```
            ${actor.name}
        </core:forEach>
    </jira:GetProjectRoleActors>
</JiraJelly>
```

1.15.4.31. jira:IsProjectRoleNameUnique

This tag will return 'true' or 'false' to let you know if there is already a project role with the given name.

Attribute name	Type	Default value	Description
name	string		The name of the project role
var	string		The name of the variable you want the returned result assigned to.

Table 1: Attributes

Examples

Determining if a project role is unique.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:IsProjectRoleNameUnique name="unique name" var="isUnique" >
        ${isUnique}
    </jira:IsProjectRoleNameUnique>
</JiraJelly>
```

1.15.4.32. jira:RemoveActorsFromProjectRole

This tag will remove a list of role actors from a given project role for a given project.

Attribute name	Type	Default value	Description
projectroleid	int		The id of the project role you wish to remove members from
actors	string		A comma delimited list of users or groups you wish to remove from the project role
projectkey	string		The key of the project the project role is associated with
actortype	string		The type of 'actor' you are working with. Currently the available options are 'atlassian-group-role-actor' or 'atlassian-user-role-actor'

Table 1: Attributes

Examples

Removing a list of groups from a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:RemoveActorsFromProjectRole projectroleid="1"
        actors="jira-administrators, jira-users" projectkey="MKY"
        actortype="atlassian-group-role-actor" />
</JiraJelly>
```

1.15.4.33. jira:RemoveActorsFromDefaultProjectRole

This tag will remove a list of role actors (i.e. users and/or groups) from the default membership of a given project role.

Attribute name	Type	Default value	Description
projectroleid	int		The id of the project role you wish to remove default actors from
actors	string		A comma delimited list of users or groups you wish to remove from the default project role
actortype	string		The type of 'actor' you are removing. Currently the available options are 'atlassian-group-role-actor' or 'atlassian-user-role-actor'

Table 1: Attributes

Examples

Removing a list of groups from a default project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:RemoveActorsFromDefaultProjectRole projectroleid="1"
        actors="jira-administrators, jira-users" actortype="atlassian-group-role-actor"
</JiraJelly>
```

1.15.4.34. jira:UpdateProjectRole

This tag will update the name and description for a given project role id.

Attribute name	Type	Default value	Description
projectroleid	int		The id of the project role you want to query
name	string		The name you want the project role

			updated with
description	string		The description you want the project role updated with

Table 1: Attributes

Examples

Updating a project role

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">
    <jira:UpdateProjectRole projectroleid="123" name="unique name"
        description="my project role is nice" />
</JiraJelly>
```

1.15.4.35. Beta Tags

There are also a number of BETA tags that have not been fully tested or documented. The following list contains the tags and the attributes that can be passed to them:

- AddIssueSecurity
  - schemeId (required)
  - security (required)
  - type (required)
- AddIssueSecurityLevel
  - name (required)
  - description (required)
  - Output
    - jelly.issue.scheme.level.id
- CreateIssueSecurityScheme
  - name (required)
  - description (required)
  - Output
    - jelly.issue.scheme.id
- LoadManager
  - var (variable to put manager in)
  - manager (name of manager e.g. IssueManager)
- LoadProject
  - var (variable to put project in)
  - project-name (name of project)
- RemoveGroup
  - name (required)
- RemovePermissionScheme
  - schemeId (required)
  - confirm (required))
- RemoveProject

    - pId (required)
- SelectProjectScheme
    - projectKey (required)
    - permission-scheme (Name of permission scheme)

  or
    - issue-scheme (Name of issue security scheme)
- StringContains
    - value (String to look in)
    - possiblyContains (String to look for)
    - doesContain (true or false) if value contains possiblyContains == doesContain, the inside of the tag is executed.

If you would like more information on how to use the Beta tags, please read the source and/or post to the JIRA Development Forum.

1.15.4.36. Sample scripts

Creating a new Project

To properly partition projects, one needs a permission scheme per project, and project-specific groups to allocate permissions to. Setting up a new project can be a time-intensive process. The following sample Jelly scripts automate this:

This script might be used for a publicly visible project:

```
<?xml version="1.0"?>
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:j="j

  <j:set var="name" value="Test Project"/>
  <j:set var="key" value="TEST"/>
  <j:set var="lowerkey" value="test"/>
  <j:set var="lead_username" value="joe"/>
  <j:set var="lead_password" value="joe"/>
  <j:set var="lead_fullname" value="Joe Bloggs"/>
  <j:set var="lead_email" value="joe@example.com"/>
  <j:set var="url" value="http://example.com/TestProj"/>

  <jira:CreateUser username="${lead}" password="${lead}" confirm="${lead}"
    fullname="${lead_fullname}" email="${lead_email}"/>
  <jira:CreateGroup group-name="${lowerkey}-developers">
    <jira:AddUserToGroup username="${lead}"/>
  </jira:CreateGroup>

  <jira:CreateProject key="${key}" name="${name}" url="${url}" lead="${lead}">
    <jira:CreatePermissionScheme name="${name} permissions">
      <jira:AddPermission type="reporter" permissions="Close"/>
      <jira:AddPermission group="jira-administrators" permissions="Close,Delete" type="
      <jira:AddPermission group="jira-users" permissions="Create,Edit,Comment,Link,Atta
      <jira:AddPermission group="${lowerkey}-developers"
        permissions="Project,ScheduleIssue,Move,Assign,Assignable,Resolve,Close,Work" ty
      <jira:AddPermission group="Anyone" permissions="Browse,ViewVersionControl"/>
      <jira:SelectProjectScheme/>
    </jira:CreatePermissionScheme>
  </jira:CreateProject>
</JiraJelly>
```

This script is more complicated, with multiple groups per project:

```
<?xml version="1.0"?>
<!-- This script handles some of the administrative chores required when adding
a new project to JIRA. It creates the project, groups, permission scheme, and gives
groups the relevant permissions in the permission scheme. -->
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:j="jelly:core">
```

```
<!-- Name of the project to create -->
<j:set var="name" value="Jelly Test Project"/>
<!-- Key for the new project -->
<j:set var="key" value="TEST"/>
<!-- Existing user who will become the project lead (default assignee) -->
<j:set var="admin" value="admin"/>

<jira:CreateGroup group-name="${key}-users"/>
<jira:CreateGroup group-name="${key}-developers"/>
<jira:CreateGroup group-name="${key}-managers"/>
<jira:CreateGroup group-name="${key}-bizusers"/>
<jira:CreateGroup group-name="${key}-qa"/>

<jira:CreateProject key="${key}" name="${name}" lead="${admin}">
<jira:CreatePermissionScheme name="${key} Permission Scheme">
<jira:AddPermission type="reporter" permissions="Edit"/>
<jira:AddPermission type="assignee" permissions="Resolve"/>
<jira:AddPermission group="jira-administrators" permissions="Project,Delete" type="group"/
<jira:AddPermission group="${key}-users" permissions="Browse,Create,Comment,Attach" type="
<jira:AddPermission group="${key}-developers" permissions="Move,Assignable,Link,ViewVersio
    type="group"/>
<jira:AddPermission group="${key}-managers" permissions="Edit,Assign,Assignable,Resolve,Cl
    type="group"/>
<jira:AddPermission group="${key}-bizusers" permissions="Assignable" type="group"/>
<jira:AddPermission group="${key}-qa" permissions="Assignable" type="group"/>
<jira:AddPermission group="opsmgrs" permissions="Browse,Edit,Assignable,Comment" type="gro
<jira:AddPermission group="dba-user-group" permissions="Browse,Assign,Assignable,Comment"
<jira:AddPermission group="help-desk-group" permissions="Browse,Assign,Assignable,Comment"
<jira:AddPermission group="webadmin-group" permissions="Browse,Assign,Assignable,Comment"
<jira:AddPermission group="unix-admin-group" permissions="Browse,Assign,Assignable,Comment
<jira:SelectProjectScheme/>
</jira:CreatePermissionScheme>
</jira:CreateProject>
</JiraJelly>
```

For a list of projects, perform a project-specific operation.

This script iterates through a (comma-separated) list of projects, creates a project-specific group, and adds a user to that group.

```
<?xml version="1.0"?>
<!-- Jelly script to create 'support' group per project -->
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib" xmlns:util="jelly:util"
<util:tokenize var="projects" delim=",">ARM,QWI,DWI,DBOR,DBSQ,LYX,MMM,MOI,TPAI,SEP,AMR,SLA
<j:forEach var="proj" items="${projects}">
<jira:CreateGroup group-name="${proj}-support"/>
<jira:AddUserToGroup username="jeff" group-name="${proj}-support"/>
</j:forEach>
</JiraJelly>
```

Create a user, issue, and assign the issue to the user

The following script creates a user (called `new-user`), creates a new issue, adds the user to the `jira-developers` group and assigns the issue to the user. It illustrates the use of context variables.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">

<jira:CreateUser username="new-user" password="password" confirm="password"
    fullname="Full name" email="test@test.com"/>
Username is ${jelly.new.username}
 <jira:CreateIssue project-key="TP" summary="New issue summary" issueKeyVar="ik"/>
 <jira:AddUserToGroup username="new-user" group-name="jira-developers"/>
 <jira:AssignIssue key="${ik}" assignee="${jelly.new.username}"/>

</JiraJelly>
```

Assigning and Starting Progress

Here we create an issue, assign it to 'bob' (who must be in `jira-developers`), and start progress:

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.JiraTagLib">

 <jira:CreateIssue project-key="TP" summary="New issue" issueKeyVar="ik"/>
 <jira:AssignIssue key="${ik}" assignee="bob"/>
 <jira:TransitionWorkflow key="${ik}" user="bob" workflowAction="Start Progress" />
</JiraJelly>
```

Moving unreplied-to issues into an 'Inactive' state

When JIRA is used for interacting with customers, this script is useful for finding issues which are awaiting customer response, and haven't been responded to in a while. It moves such issues into an 'Inactive' state.

You would typically invoke this script periodically with the Jelly Service.

```
<JiraJelly xmlns:jira="jelly:com.atlassian.jira.jelly.enterprise.JiraTagLib" xmlns:core=
<jira:Login username="customersupport" password="XXXXXX">
        <log:warn>Running Inactivate issues service</log:warn>
        <core:set var="comment">This issue has not been updated for 5 business days.

If you have an update, please use "Add Comments For Vendor" action to let us know.
If you need more time to gather information please let us know and we will 'freeze' this
If you have no other questions, please Close this issue.

If no update is received in the next 5 business days, this issue will be automatically

Thank you,

  The Support Team</core:set>
        <core:set var="workflowStep" value="Mark Inactive" />
        <core:set var="workflowUser" value="customersupport" />

        <!-- Run the SearchRequestFilter -->
        <jira:RunSearchRequest filterid="11505" var="issues" />

        <core:forEach var="issue" items="${issues}">
                <log:warn>Inactivating issue ${issue.key}</log:warn>
                <jira:TransitionWorkflow key="${issue.key}" user="${workflowUser}" work
        </core:forEach>
</jira:Login>
</JiraJelly>
```

Where:

- **workflowStep** is the name of a workflow transition, e.g "Close Issue", "Start Progress", just as they appear in the left-hand menu on the issue screen.
- **workflowUser** is the user to run the transition as
- **filterid** is the id of a saved search (filter), which finds issues needing to be inactivated (transitioned). This ID can be discovered from the filter URL on the "Manage" tab in "Find issues".

The JIRA Toolkit is useful in conjuction with this script, to find issues awaiting customer response.

1.15.5. The JIRA Toolkit (Customer Support Extensions)

As an extension to JIRA, Atlassian have developed a set of JIRA custom fields, collectively called the "JIRA Toolkit". It can be found online at:

http://confluence.atlassian.com/display/JIRAEXT/JIRA+Toolkit

These custom fields are particularly useful in customer-facing JIRA instances. They were initially developed for use in Atlassian's own JIRA Support installation at http://support.atlassian.com. See the JIRA Toolkit documentation for details.

1.15.6. Developer Guides

1.15.7. JIRA Source

1.15.7.1. Building JIRA from source

Commercial users get access to JIRA source. This documentation shows how to build the JIRA source back into an application that can be deployed.

You would only be interested in this documentation if you are making modifications to the JIRA source code. Changes to JSP files do not require rebuilding JIRA. Also, you should be aware of the possibilities the plugin system affords - often changes can be developed and packaged as a plugin without requiring core source modifications.

Building JIRA WAR from JIRA Source release

1. Ensure you have JDK 1.4 or higher.
2. Download Maven 1.0.x from http://maven.apache.org
3. Extract Maven somewhere, say c:\Dev\testing
4. Set MAVEN_HOME:
   ```
   > set MAVEN_HOME=c:\Dev\testing\maven-1.0
   ```
5. Download Maven 2.0.x from http://maven.apache.org
6. Follow the general instructions for setting up Maven 2.0.x
7. Download JIRA Source zip from http://www.atlassian.com/software/jira/JIRASourceDownloads.jspa. You will need to log in as a user with a commercial licence to access this page.
8. Extract the JIRA Source zip somewhere, say c:\Dev\testing.
9. Your c:\Dev\testing should look somewhat like:

```
C:\Dev\testing>dir

Volume in drive C is COOKIE
Volume Serial Number is 3F3F-14F0

Directory of C:\Dev\testing

31/01/2009  04:30p      <DIR>          .
31/01/2009  04:30p      <DIR>          ..
31/01/2009  04:18p      <DIR>          atlassian-cache-servlet
31/01/2009  04:18p      <DIR>          atlassian-core
31/01/2009  04:18p      <DIR>          atlassian-gzipfilter
31/01/2009  04:18p      <DIR>          atlassian-jdk-utilities
31/01/2009  04:18p      <DIR>          atlassian-ofbiz
31/01/2009  04:18p      <DIR>          atlassian-profiling
31/01/2009  04:18p      <DIR>          atlassian-renderer
31/01/2009  04:18p      <DIR>          atlassian-velocity
31/01/2009  04:18p      <DIR>          bandana
31/01/2009  04:18p      <DIR>          bonnie
31/01/2009  04:18p      <DIR>          configurableobjects
31/01/2009  04:18p      <DIR>          jira
31/01/2009  04:18p      <DIR>          jira-bamboo-plugin-v2
31/01/2009  04:18p      <DIR>          jira-fisheye-plugin
31/01/2009  04:18p      <DIR>          johnson
31/01/2009  04:18p      <DIR>          mail
31/01/2009  04:18p      <DIR>          plugins
```

```
31/01/2009  04:18p      <DIR>          rpc-jira-plugin
31/01/2009  04:18p      <DIR>          scheduler
31/01/2009  04:18p      <DIR>          seraph
31/01/2009  04:18p      <DIR>          trackback

0 File(s)              0 bytes
21 Dir(s)  16,352,509,952 bytes free
```

10. Change into the jira\ subdirectory, and build using Maven by executing the following command:

```
C:\Dev\testing\jira> maven war:webapp
```

If you would like to build a closed WAR file, then do not use the command displayed above. You will need to run the following maven command instead:

```
maven -Djira.build.bundle.plugins=false include-rpc-plugin war:war
```

Note:

The **-Djira.build.bundle.plugins=false include-rpc-plugin** part prevents JIRA trying to build the Fisheye plugin, which was bundled with 3.12, but which is not buildable from the JIRA source distribution. It is not required in earlier or later releases.

Note:

If you are attempting to build JIRA 3.13, you will need to make changes to the **build.properties** file before running your build, as the maven repository information is incorrect. Hence, your build will not be able to find dependent JARs, such as atlassian-mail. See JRA-15648 for detailed instructions. Please note, this issue only affects JIRA 3.13, it does not apply to JIRA 3.13.x.

11. Confirm that the open .war has been created in .\target\atlassian-jira

```
C:\Dev\testing\jira\target\atlassian-jira>dir
Volume in drive C is COOKIE
Volume Serial Number is 3F3F-14F0

Directory of C:\Dev\testing\jira\target\atlassian-jira

24/02/2003  04:41p      <DIR>          .
24/02/2003  04:41p      <DIR>          ..
24/02/2003  04:41p      <DIR>          decorators
24/02/2003  04:41p      <DIR>          images
24/02/2003  04:41p      <DIR>          includes
24/02/2003  04:41p      <DIR>          portlets
24/02/2003  04:41p      <DIR>          secure
24/02/2003  04:41p      <DIR>          styles
24/02/2003  04:41p      <DIR>          template
24/02/2003  04:41p      <DIR>          views
24/02/2003  04:41p      <DIR>          WEB-INF
24/02/2003  04:41p                 8781 500page.jsp
24/02/2003  04:41p                 1593 bugzillasearch.jsp
24/02/2003  04:41p                 328 default.jsp
24/02/2003  04:41p                 894 favicon.ico
24/02/2003  04:41p                 211 login-error.jsp
24/02/2003  04:41p                 203 login.jsp
24/02/2003  04:41p                 733 logoutconfirm.jsp
24/02/2003  04:41p                 939 logout.jsp
8 File(s)  13,682
11 Dir(s)  56931786752 bytes free
```

You should now be able to point a suitably configured Servlet 2.3+ compliant app server at this directory, and run JIRA.

Developing using Intellij IDEA

If you are an IDEA user, you may wish to use the atlassian-idea plugin we have developed to quickly generate a work environment.

Building the Atlassian source dependencies

JIRA's source distribution not only ships with JIRA's source code, it also includes the source of the internal Atlassian projects that JIRA depends on (e.g. atlassian-bonnie, atlassian-core, etc.). These dependencies are included in JIRA in binary format when you build the JIRA source (they are downloaded from the Atlassian maven repository).

You can, however, compile the provided source to generate the binaries yourself. These projects use a mix of Maven 1 and Maven 2 build systems to compile and package their source. You can tell a project uses Maven 1 if the project contains a file called 'project.xml' in the top level directory. If a project uses Maven 2, it will contain a file called 'pom.xml' in the top level directory.

Building a Maven 1 project you will invoke 'maven jar', whereas for a Maven 2 project you will invoke 'mvn package'. In order to run the 'mvn' command you will have to install Maven 2. Please follow the general instructions regarding setting up a development environment. Please note that you will also have to add the Atlassian Maven 2 repository to your Maven 2 configuration. To do this you will need to edit your settings.xml as described here.

Obtaining the source of JIRA's dependencies

Most of JIRA's dependencies are either shipped in binary (compiled) form with the source distribution, or are available on Maven's public repository. Maven will fetch the dependencies that it requires automatically during the build process, so you do not have to do it manually. Hence, you do not need the source of every dependency to build JIRA from source. However, sometimes you might want to "look inside" these dependencies. If so, this section is for you.

The source distribution of JIRA is shipped with a `project.xml` file. All of JIRA's dependencies are listed inside this file. Most of the dependencies are open source libraries but some are Atlassian's code. All of the Atlassian code is included in the source distribution. The source of the other dependencies is usually available on the library's website (try googling for the library name).

In some cases JIRA uses unofficial 'snapshot' releases of a library, sometimes additionally patched to fix bugs or add features. In these cases the library source can be obtained from Atlassian's repository, at http://repository.atlassian.com/**dependencyId**/distributions/, where **dependencyId** is the dependency name found in the `project.xml` record.

For example, source for the dependency:

```
    <dependency>
      <id>javacvs</id>
      <version>20050531-patched</version>
      <properties>
        <war.bundle>true</war.bundle>
      </properties>
    </dependency>
```

can be found at http://repository.atlassian.com/javacvs/distributions/javacvs-20050531-patched-src.tar.gz. If source modifications were made, a patch is usually available at http://repository.atlassian.com/**dependencyId**/patches/

If you have any questions regarding the build process, please post to the JIRA Development Forum, which is monitored continually by the development community, and by Atlassian as often as possible.

Compiling classes into JIRA Standalone

If you just want to compile one class (perhaps a service) and you're using the JIRA Standalone distribution, there is an Ant-based mini-build system available in the `external-source` directory. See JIRA Standalone quick source modifications for details.

1.15.7.2. API Docs

The JIRA API docs are available online. They are most useful for:

- users writing Plugins, Listeners and Services
- users with commercial licenses who wish to modify JIRA
- partners embedding JIRA as a J2EE component

The latest API docs are available at http://docs.atlassian.com/software/jira/docs/api/latest/. The 3.13.3 docs are available at http://docs.atlassian.com/software/jira/docs/api/3.13.3. For previous versions, substitute the appropriate version in the URL.