



Administering JIRA applications 7.8

Contents

Administering JIRA Server 7.8 applications	7
JIRA applications and project types overview	7
Getting started as an administrator	9
Setting up your instance	10
Creating a project	11
Adding new users	12
Managing permissions	13
Installing JIRA applications	15
JIRA applications installation requirements	15
Installing Java	17
Supported platforms	19
End of support announcements	22
Evaluation installation	26
Installing JIRA applications on Windows	28
Uninstalling JIRA applications from Windows	32
Installing JIRA applications on Windows from Zip File	33
Installing JIRA applications on Linux	37
Uninstalling JIRA applications from Linux	42
Installing JIRA applications on Linux from Archive File	42
Unattended installation	48
Installing additional applications and version updates	49
Connecting JIRA applications to a database	52
Connecting JIRA applications to PostgreSQL	52
Connecting JIRA applications to MySQL	56
Connecting JIRA applications to Oracle	62
Connecting JIRA applications to SQL Server 2012	65
Connecting JIRA applications to SQL Server 2014	70
Tuning database connections	74
Surviving connection closures	82
Switching databases	84
Installing JIRA Data Center	85
Running the setup wizard	89
Licensing and application access	92
License compatibility	93
Extending JIRA applications	94
Integrating with development tools	95
Getting started with Bitbucket and JIRA Cloud	100
Enable Smart Commits	107
Synchronize an account	110
Configure automatic team invitations	111
Linking a Bitbucket or GitHub repository with JIRA	112
Integrating with other tools	117
Listeners	118
Managing webhooks	120
Services	121
Using AppLinks to link to other applications	123
Administering projects and links across multiple applications	125
Integrating with collaboration tools	126
Managing add-ons	130
Upgrading JIRA applications	131
Upgrading JIRA applications using the installer	131
Upgrading JIRA applications manually	136
Upgrading JIRA applications with a fallback method	141
Upgrading JIRA Data Center	143
Managing zero downtime upgrades	146
Zero downtime upgrade checklist	150

Upgrade task troubleshooting	151
Rolling back a JIRA application upgrade	151
Establishing staging server environments for JIRA applications	152
Migrating JIRA applications to another server	154
Migrating from JIRA Cloud to Server applications	156
Getting started with JIRA Data Center on AWS	159
Configuring JIRA Data Center on AWS	161
Administering JIRA Data Center on AWS	163
Upgrading JIRA Data Center on AWS	164
Getting started with JIRA Data Center on Azure	166
Administering JIRA Software Data Center on Azure	168
Layout and design	169
Configuring the look and feel of your JIRA applications	170
Configuring an announcement banner	172
Configuring the default dashboard	172
Using dashboard gadgets	173
Adding a gadget to the directory	175
Subscribing to another application's gadgets	177
Choosing a default language	178
Translating JIRA	180
Configuring the default issue navigator	180
Creating links in the application navigator	182
Configuring the user default settings	183
User management	184
Managing users	184
Create, edit, or remove a user	185
Assign users to groups, project roles, and applications	189
Monitor a user's activity	190
Manage password security	191
Prevent automatic login	192
SAML SSO for JIRA Data Center applications	193
Enabling public signup and CAPTCHA	196
Managing groups	198
View, create, or delete a group	199
Modify group membership	200
Assign group access to a project role	200
Manage group access to applications	201
Advanced user management	202
Allowing connections to JIRA for user management	202
Diagrams of possible configurations for user management	205
Managing nested groups	212
User management limitations and recommendations	214
Configuring user directories	218
Configuring the internal directory	220
Connecting to an LDAP directory	221
Configuring an SSL connection to Active Directory	230
Reducing the number of users synchronized from LDAP to JIRA applications	240
Connecting to an internal directory with LDAP authentication	240
Connecting to Crowd or another JIRA application for user management	246
Managing multiple directories	253
Migrating users between user directories	254
Synchronizing data from external directories	257
Configuring projects	259
Defining a project	259
Editing a project key	264
Changing the project key format	266
Configuring issues	268
Configuring built-in fields	270
Defining issue type field values	270
Defining priority field values	274
Defining resolution field values	278
Defining status field values	278

Translating resolutions, priorities, statuses, and issue types	279
Issue fields and statuses	280
Configuring issue-level security	283
Configuring permissions	286
Managing global permissions	290
Managing project permissions	293
Customizing JIRA Service Desk permissions	299
Resolving JIRA Service Desk permission errors	300
Using Manage Sprints permission for advanced cases	302
Managing project roles	304
Managing project role membership	306
Allowing anonymous access to your project	307
Managing components	307
Managing versions	309
Creating release notes	311
Project screens, schemes and fields	312
Adding a custom field	313
Configuring a custom field	314
Specifying field behavior	317
Associating field behavior with issue types	321
Configuring renderers	324
Defining a screen	329
Associating a screen with an issue operation	332
Associating screen and issue operation mappings with an issue type	335
Creating a notification scheme	337
Using the issue collector	340
Advanced use of the JIRA issue collector	344
Working with workflows	357
Managing your workflows	361
Configuring workflow schemes	364
Sharing your workflow	366
Advanced workflow configuration	369
Working in text mode	376
Adding a custom event	378
Configuring the initial status	381
Configuring workflow triggers	382
Using validators with custom fields	392
Using XML to create a workflow	392
Workflow properties	393
Configuring JIRA Service Desk approvals	394
Importing and exporting data	397
Migrating from other issue trackers	398
Importing data from CSV	399
Commonly asked CSV questions and known issues	406
How to import CSV data with PVCS command	408
Importing data from Excel	408
Importing data from Bitbucket	409
Importing data from Github	411
Importing Data from Asana	413
Importing data from TFS or Visual Studio	414
Importing data from Rally	414
Importing data from VersionOne	415
Importing data from YouTrack	415
Importing data from Axosoft	416
Importing data from Pivotal Tracker	417
Importing data from Bugzilla	420
Importing data from FogBugz On Demand	425
Importing data from FogBugz for your Server	428
Importing data from Trac	434
Importing data from Redmine	437
Importing data from BaseCamp	439
Importing data from JSON	439

Importing data from Mantis	449
Moving or archiving individual projects	453
Archiving a project	453
Splitting JIRA applications	455
Exporting issues	455
Importing issues from JIRA server applications	457
Migrating data with 3rd party add-ons	458
Migrating data with Adaptavist	459
Promoting configuration changes from staging to production	460
Migrating projects to another JIRA instance	467
Merging JIRA instances	476
Migrating data with Botron	478
Promoting JIRA configuration from development to production	478
Migrating JIRA projects	487
Consolidating multiple JIRA instances	494
Configuring JIRA application emails	494
Configuring email notifications	494
Configuring an SMTP mail server to send notifications	497
Customizing email content	502
Creating issues and comments from email	503
Configuring JIRA applications to receive email from a POP or IMAP mail server	516
JIRA system administration	518
System administration	519
Finding your Server ID	519
Increasing JIRA application memory	520
Using the database integrity checker	525
Precompiling JSP pages	526
Logging and profiling	527
Logging email protocol details	530
Backing up data	533
Automating JIRA application backups	534
Preventing users from accessing JIRA applications during backups	536
Restoring data	537
Restoring a project from backup	537
Anonymising JIRA application data	546
Restoring data from an xml backup	548
Restoring information from a native backup	549
Search indexing	550
Re-indexing after major configuration changes	553
Using robots.txt to hide from search engines	554
Licensing your JIRA applications	554
Viewing your system information	556
Live monitoring using the JMX interface	562
Monitoring database connection usage	564
Viewing JIRA application instrumentation statistics	566
Generating a thread dump	569
Finding your JIRA application Support Entitlement Number (SEN)	575
Auditing in JIRA applications	576
Important directories and files	578
JIRA application installation directory	581
JIRA application home directory	583
Setting your JIRA application home directory	584
Integrating JIRA applications with a Web server	586
Integrating JIRA applications with IIS	586
Integrating JIRA with Apache	591
Securing JIRA applications with Apache HTTP Server	611
Using Apache to limit access to the JIRA administration interface	611
Using Fail2Ban to limit login attempts	626
Changing JIRA application TCP ports	628
Connecting to SSL services	629
Running JIRA applications over SSL or HTTPS	634
Configuring security in the external environment	647

Data collection policy	647
JIRA Admin Helper	648
Raising support requests as an administrator	650
Start and Stop JIRA applications	652
Managing LexoRank	655
Configuring global settings	657
Configuring time tracking	658
Configuring JIRA application options	660
Configuring advanced settings	664
Configuring the Base URL	666
Setting properties and options on startup	667
Recognized system properties for JIRA applications	673
Advanced JIRA application configuration	676
Changing the constraints on historical time parameters in gadgets	677
Changing the default order for comments from ascending to descending	678
Limiting the number of issues returned from a search view such as an RSS feed	678
Configuring file attachments	679
Configuring issue linking	683
Configuring issue cloning	686
Configuring the whitelist	686
Configuring sub-tasks	688
Managing shared filters	690
Managing shared dashboards	691
Enabling logout confirmation	693
Rich text editing	693
Server optimization	694
Configuring secure administrator sessions	694
JIRA application cookies	695
Preventing security attacks	697
Using the JIRA application configuration tool	699
Running JIRA applications as a Windows service	703
Tuning garbage collection (GC)	709
Getting help	710

Administering JIRA Server 7.8 applications

A one-stop shop for administering all of your JIRA applications: JIRA Core, JIRA Software, and JIRA Service Desk.

Get started

New to JIRA? Check out our guide for new administrators.

[View guide](#)

What's new

Time to upgrade? Get the lowdown on the latest and greatest in JIRA 7.8.

[View latest changes](#)

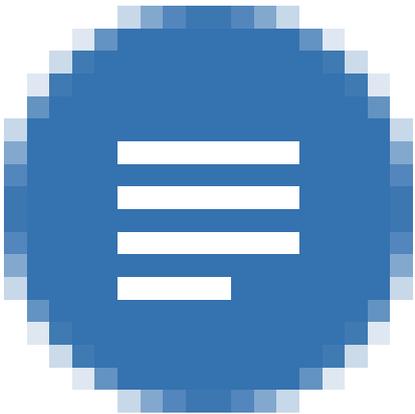
JIRA applications and project types overview

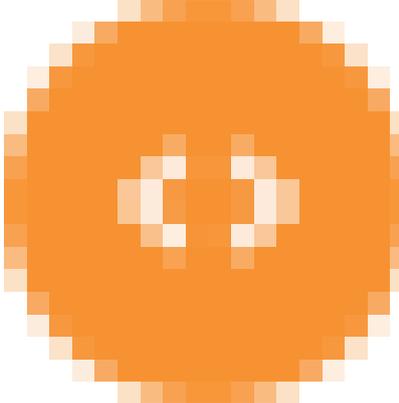
The JIRA family of applications are built to deliver a tailored experience to their user. JIRA Core is the default application of JIRA, and will always be present in a JIRA instance. You may also choose to include other applications in your instance, such as JIRA Software or JIRA Service Desk. A user may require access to one, all, or any combination of these applications.

Note that as JIRA Core is the default application, if you have a license for JIRA Software or JIRA Service Desk, your users automatically have access to JIRA Core *without* requiring an additional license. For example, a JIRA Software user can view development information on an agile board, and can also view business projects.

Application features and project types

Each application delivers a tailored experience for its users, and has an associated project type, which in turn, offers application-specific features. Below is a list of the project types, and their associated application-specific features.

Application	Project type	Application-specific feature set
JIRA Core	 <p>Business projects</p>	<ul style="list-style-type: none"> Available to all licensed users of JIRA

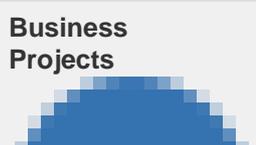
<p>JIRA Software</p>	 <p>Software projects</p>	<ul style="list-style-type: none"> • Integration with development tools • Agile boards • Release hub for software version release
<p>JIRA Service Desk</p>	 <p>Service Desk projects</p>	<ul style="list-style-type: none"> • Service Level Agreements (SLAs) • A customizable web portal for customers • Permission schemes allowing customer access

Application features and users

All users that can log in to a JIRA instance will be able to see all the projects in that instance (pending permissions), but they will only be able to see the application-specific features when they have application access. For example, a Software project is able to display information from linked development applications, such as Bitbucket and FishEye on a Software project, and you can create agile boards, but this information is only viewable by a JIRA Software user. A JIRA Core user would be able to see the Software project, but would not be able to see the application-specific features, like agile boards or development information. Likewise, a JIRA Software user would not be able to see any JIRA Service Desk application-specific features on a Service Desk project — only a basic view of the project and its issues.

- Only a JIRA administrator can create a project for an installed application. They do not need application access to create the project, but they do need application access if they'd like to view or use the project.
- Anonymous users will have access equivalent to JIRA Core users. In other words, they can view issues and work in any type of project, but they won't see application-specific features, e.g. agile boards, which are JIRA Software-specific features. To know how to allow anonymous users access to projects, see [Allowing anonymous access to your instance](#).

A list of the applications, their default user groups, and their project's application-specific features is listed below:

			JIRA Core	JIRA Software	JIRA Service Desk
			jira-core-user	jira-software-user	jira-servicedesk-agent
 <p>Business Projects</p>	Project level	View			
	Issue level	Create			

		View	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
		Comment			
		Transition			
	JIRA Gadgets	View			
Software Projects 	Project level	View			
	Issue level	Create			
		View			
		Comment			
		Transition			
		View development information			
		View release information			
	Board level	Create			
		View			
	JIRA Software gadgets	View			
Service Desk Projects 	Project level	View			
	Issue level	Create			
		View			
		Comment			
		Transition			
	SLA level	Create			
		View			
	Queue level	Create			
		View			
	JIRA Service Desk gadgets	View			

Getting started as an administrator

This tutorial will get you up to speed with JIRA, regardless of whether you're using JIRA Core, JIRA Software, or JIRA Service Desk. All administrators should complete it to understand the basic concepts of managing a JIRA instance.

Here's what you can count on learning:

1. How to set up an evaluation instance of JIRA
2. How to add new users
3. How to create a project and customize it
4. How to manage permissions

By the end of this tutorial, you'll have a fully functioning JIRA instance, several users with different access permissions, and you'll have created your first project.

Audience

Administrators

Time

45 minutes

Ready to dive in and get your hands dirty? Get started and learn how to set up your own JIRA.

[Let's get started](#)

Setting up your instance

1. Setting up your instance
2. Creating a project
3. Adding new users
4. Managing permissions

As a first step, you'll set up an evaluation instance of JIRA. We'll guide you through three simple steps to get JIRA up and running in no time!

On this page:

[Before you begin](#)
[Download the installer](#)
[Install your JIRA application](#)
[Set up your JIRA application](#)

If you're ready to set up a production JIRA site or you want more control, check out our [full installation guides](#).

Before you begin

Our installers come with all the bits and pieces you need to run the application, but there's a few things you'll need to get up and running:

- A computer or laptop with a supported operating system - you'll be installing JIRA so you'll need admin rights.
 - ▾ [Supported operating systems...](#)
 You can install JIRA on a Windows or Linux operating system.
 - Apple Mac isn't supported for production sites, but if you're comfortable setting up applications on your Mac from scratch, you can download the `tar.gz` file and follow the instructions for [Installing JIRA applications on Linux from Archive File](#) as the process is similar.
- A supported web browser - you'll need this to access JIRA, we support the latest versions of Chrome and Mozilla Firefox, Internet Explorer 11, and Microsoft Edge.
- A valid email address - you'll need this to generate your evaluation license and create an account.

Ready to get going? Let's start with grabbing the installer.

Download the installer

Head to www.atlassian.com/software/jira/core/download and download the installer for your operating system.

Install your JIRA application

The installer allows you to choose Express or Custom installations.

The Custom installation allows you to pick some specific options for JIRA, but for this guide we'll use the Express installation.

▼ For Windows

1. Run the installer - we recommend running with a Windows administrator account. If prompted, make sure you allow the installer to make changes to your computer. This will allow you to install JIRA as a service.
2. Choose **Express Install**, then click **Next**.
3. Once installation is complete, it will ask you if you want to open JIRA in your browser. Make sure this option is selected then click **Done**.
4. JIRA will open in your default browser, and you're ready to start the set up wizard.

▼ For Linux

1. Change to the directory where you downloaded JIRA then execute this command to make it executable:

```
$ chmod a+x atlassian-jira-software-X.X.X-x64.bin
```

Where `jira-software.X.X.X` is the JIRA version you downloaded.

2. Run the installer - we recommend using `sudo` to run the installer as this will create a dedicated account to run JIRA and allow you to run JIRA as a service.

```
$ sudo ./atlassian-jira-software-X.X.X-x64.bin
```

3. When prompted, choose **Express Install** (option 1).
4. Once installation is complete head to <http://localhost:8080> in your browser to begin the setup process.

Set up your JIRA application

The set up wizard is the last step in getting JIRA up and running. You'll need your email address to generate your evaluation license.

1. Select **Set it up for me** and then **Continue to MyAtlassian**. This will allow JIRA to set up everything it needs to run, including an H2 database.
2. Create an account (or log in if you already have an Atlassian ID account).
3. Follow the prompts to **generate a license** for the JIRA application you want to try, and apply it to your new installation.
4. Enter and confirm the details you want to use for your administrator account, and click **Next**.
5. It will take a few minutes to get everything connected and operational.

That's it! Let's now create your first project.

Next

Creating a project

1. [Setting up your instance](#)
2. Creating a project
3. Adding new users
4. Managing permissions

A JIRA project is a container that holds issues. Issues can be viewed as the packets of work required within a project. To create issues, you must have an available project to contain them. JIRA comes with several default project types with preconfigured workflows and issue types, so you can quickly get your project up and running. In this step of the tutorial, you will use the project management template to help your team plan, organize, and collaborate on their work.

Note that creating and configuring a project is done by an administrator. A project administrator controls user access to the project, and can only configure certain aspects of the look and feel of the project. You should still be logged in to JIRA as an administrator from the previous step. If not, log into your administrator account.

Create a project

When creating a project, you will need to give it a name, a key, and a project lead. The title can be as descriptive as you want, and the key should be something meaningful. The project lead is usually the project manager, but can effectively be any user you select when creating the project.

1. If you're following from the previous step, you should see different project options on the welcome screen. Click **Create new project**.
2. Select **Project management** as the project type.
3. Enter **Dragon Design Tees** as the project name. Note that JIRA creates a Project key for you, but you can overwrite this if you want to.
4. Select **Submit** to create your new project.

About project keys

Each project has a unique *name* (e.g. **Dragon Design Tees**) and a unique *key* (e.g. **DDT**). The project key becomes the first part of that project's *issue keys*, e.g. **DDT-1**, **DDT-2**, etc.

Customize your project

In this step, you will be customizing your project avatar and project details to help your team identify the project more easily. These customizations are helpful if you have several projects in your JIRA instance. If you have navigated away from your project, simply go to **Projects > Dragon Design Tees**.

1. Click  **> Projects**.
2. Select **Edit** next to your project.
3. Click the **Avatar** image.
4. Select an available icon or upload an image.
5. Enter a URL and Description for your project to make it easier for your team to identify. Note that these fields are optional and only for display.
6. Click **Save details** to save your changes.

Congratulations! You've now created and customized your first project. Next, we'll add users to your project and look at how you can set up and restrict access to projects.

Next

Adding new users

1. [Setting up your instance](#)
2. [Creating a project](#)

3. Adding new users
4. Managing permissions

Working alone isn't much fun, so let's add some test users to your JIRA site. You can add users directly, or allow new users to sign up themselves. In this step in the tutorial, you'll add three users directly to your site.

Add a few users

You will be adding three users: **Jason**, **Kate** and **Emma**. You can add more or choose your own usernames if you like, but please note that we will be referring to these usernames later in the tutorial. You can always disable or delete any users you set up.

If you've logged out of JIRA, log in with the administrator account you created.

1. Navigate to the **User Management** screen by selecting



> **User management.**

2. Choose **Create User** to add a new user. Specify the username as **jason**. Set the rest of the fields to whatever you want. You're going to be creating a couple more users, so check the **Create another** checkbox before selecting **Create user**.
3. Now create two more users, with the usernames **emma** and **kate**, following the same process outlined above.

You should have a screen that looks something like this:

Full name	Username	Email address	Last session	Action
Emma	emma	emma@example.com	Invitation sent 10 days ag...	Resend
Administrator	ixadmin	admin@example.com	18 Sep 2015 5:23PM	Deactivate
Jason	jason	jason@example.com	Invitation sent 10 days ag...	Resend
Kate	kate	kate@example.com	Invitation sent 10 days ag...	Resend
System Administrator	sysadmin	noreply@atlassian.com	18 Sep 2015 5:00PM	Deactivate

Note that usernames are **not** case sensitive. Emma can enter her username as Emma, emma, or even EmMA to log into JIRA. Passwords, on the other hand, are case sensitive.

Well done! You've added three new users to your JIRA instance. Next, you'll learn how to manage access to your project with site and project permissions.

Next

Managing permissions

1. [Setting up your instance](#)
2. [Creating a project](#)
3. [Adding new users](#)
4. Managing permissions

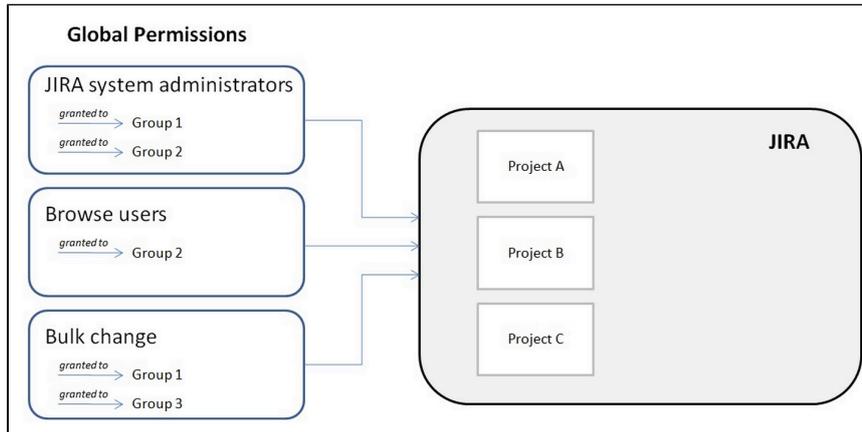
You won't want every user in your team to have the same level of access to JIRA. For example, you may want to restrict who can administer JIRA, or prevent users from viewing a project. In this step, you will learn about the different permissions in JIRA and set permissions for a new project.

Overview of roles, groups, and users

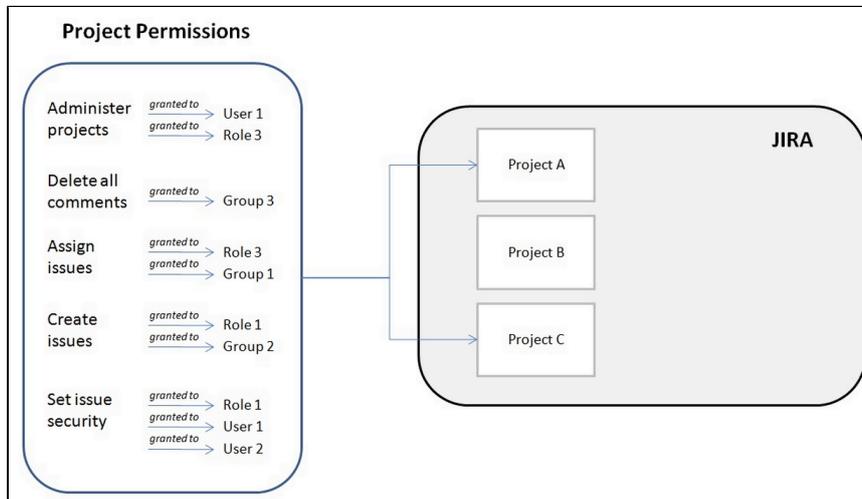
A role is a project-specific set of groups and/or individual users. In our example of the design project in the t-shirt business, all product managers need to be able to assign work (issues) across all projects, while senior designers need to be able to assign work on specific design projects. In JIRA, you can define a product manager role that includes all product managers. You can then define a set of permissions with the 'Assign issue' permission for this role, and apply this set of permissions to all projects. Individual senior designers can be added to the product manager role on each project, as needed.

Overview of global and project permissions

Global permissions cover a small set of functions that affect all projects in JIRA (for example, permission to administer JIRA). They can only be assigned to groups:



Project permissions cover a set of more granular functions that affect a single project in JIRA. For example, permission to create issues in a project. They can be assigned to groups, users and project-specific roles:



Now let's put this into practice! You're going to go through the tasks involved to use project permissions to hide a new, secret t-shirt design project from some of your users.

Create a new project role

This project role will only contain users that you want to view a particular project. We will assign permissions to this role in the next step.

1. Select



> **System.**

2. In the security section, select **Project roles**.
3. Below the existing project roles, add another project role named "Review". Leave the Description field blank for now and select **Add project role**.

4. Select **Manage Default Members** and then, under Default Users, select **Edit** to add yourself and **Jason** to the Review project role. Do not add Kate or Emma.

Configure a new permission scheme

The 'Browse Projects' permission controls whether a user can browse a project, i.e. whether they can view the project. Let's assign this permission to your new project role.

1. Select  > **Issues** > **Permission Schemes**.
2. Copy the **Default Permission Scheme**.
3. Edit the copied permission scheme and change the name to **Confidential Permission Scheme**. Select **Update**.
4. Select **Permissions** for the **Confidential Permission Scheme**. For the **Browse Projects** permission:
 - Click **Remove** for 'Application Role (Any logged in user)'.
 - Click **Edit**, select **Project Role**, and choose **Review** in the drop-down. Click **Grant**.

Associate the scheme with a new project

For the last step, let's associate the permission scheme with your new project.

1. Select **Projects** > **Create Project** and choose **Task management**.
2. Name the project **Top Secret Tee** and **Submit**.
3. In the bottom-left corner, select **Project settings** > **Permissions**.
4. On the Default Permission Scheme screen, select **Actions** > **Use a different scheme**.
5. Set the Scheme to **Confidential Permission Scheme**, and click **Associate**.

The only users that will be able to browse your new project are Jason and yourself. Note that default members are only added to a role for new projects. You can also use this approach to restrict users from creating issues, adding comments, closing issues, etc, in a project.

Well done! You've completed the Administrator Getting started tutorial!

Installing JIRA applications

Use these installation pages if you need to install a JIRA application, or add an additional JIRA application to your existing JIRA installation. If you are upgrading please refer to the [Upgrade Guide](#). It's important to review the [supported platforms](#) and [JIRA applications installation requirements](#) pages for more information on what you'll need to run a JIRA application.

Installing a JIRA application

Follow the instructions for your operating system:

- [Installing JIRA applications on Windows](#)
- [Installing JIRA applications on Linux](#)

Installing an additional JIRA application

Once you have installed your initial JIRA application, it's possible to [install additional applications](#) directly through the Versions and licenses page.

JIRA applications installation requirements

- [JIRA applications installation requirements](#)
- [Client-side requirements](#)

- [Server-side requirements for evaluation purposes](#)
- [Server-side requirements for production](#)
- [Next Steps](#)

No hardware? No problem! Try using JIRA applications in the Cloud.

- No installation required, get started in 5 minutes
- Option to migrate to your own server later
- Choose from a set of supported add-ons to install

 [Start My Free Trial](#)

JIRA applications installation requirements

JIRA is a 'web application', meaning it runs centrally on a server, and users interact with it through web browsers from any computer on the same network. As such, JIRA must be able to communicate and authenticate with itself. If you're upgrading to JIRA 7.8 be sure to review the latest release and upgrade notes [here](#).

Please read the [Supported platforms](#) page for JIRA applications, which lists the required server and client software supported by JIRA applications for:

- Browsers (client-side)
- Java platforms (JDK/JRE) (server-side)
- Operating systems (server-side)
- Application servers (server-side)
- Databases (server-side)

Please also read the information below regarding server and client software and hardware requirements for JIRA.

Client-side installation requirements

Client-side requirements

Browser	Enable your browser to execute JavaScript from your JIRA applications to access their full functionality. You can consult the supported versions here .
----------------	---

Server-side installation requirements for evaluators

Server-side requirements for evaluation purposes

Java	If you intend to use the Windows Installer or Linux Installer to install JIRA, there is no need to install and configure a separate JDK/JRE since these executable files will install and configure their own JRE to run JIRA, otherwise you will have to install a supported version of the ORACLE Java runtime. Consult the supported versions here .
Memory	500MB – 1GB of Java heap size is enough for most evaluation purposes.
Database	JIRA applications come pre-configured with the H2 database, which is suitable for evaluation purposes only, it shouldn't be used in production environments.
Security	Symantec must be uninstalled from the server that you want to install JIRA applications on, as it is known to dramatically reduce application performance. For more information, see this knowledge base article: Crashes and Performance Issues Troubleshooting .

▼ [Server-side installation requirement for production](#)

Server-side requirements for production

Java	If you intend to use the Windows Installer or Linux Installer to install JIRA, there is no need to install and configure a separate JDK/JRE since these executable files will install and configure their own JRE to run JIRA, otherwise you will have to install a supported version of the ORACLE Java runtime. Consult the supported versions here .
Hardware	<ul style="list-style-type: none"> • For a small number of projects (less or equal to 100) with 1,000 to 5,000 issues in total and about 100-200 users, a recent server (multicore CPU) with 8GB RAM and a reasonably fast hard drive (7200 rpm or faster) should cater for your needs. Additionally, allocate additional 1GB for JIRA's JVM even for small installations. Note that if your projects have multiple workflows, custom schemes and/or new permissions schemes more memory might be required. • For more than 100 projects you should monitor JIRA memory usage and allocate more memory if required. • If your system will experience a large number of concurrent requests, running JIRA applications on a multicore CPU machine will increase the concurrency of processing the requests, and therefore, speed up the response time for your users. • For reference, we have a server that has a 2 Intel(R) Xeon(R) CPU E5520 @ 2.27GHz (16 logical cores) with 32GB of RAM. This server runs Apache, various monitoring systems, and two JIRA application instances: <ul style="list-style-type: none"> • Our public site has approximately: 145,000 issues, 255,000 comments, 120 custom fields, and 115 projects. • Our support site has approximately: 285,000 issues, 2,500,000 comments, 75 custom fields, and 22 projects. <p>For more information, you can also refer to Scaling JIRA 7.8.</p>
Database	Using the embedded H2 database is not supported in production. You must install and connect your JIRA instance to an enterprise database supported by Atlassian .
Security	Symantec must be uninstalled from the server that you want to install JIRA applications on, as it is known to dramatically reduce application performance. For more information, see this knowledge base article: Crashes and Performance Issues Troubleshooting .

If you are considering running JIRA applications on VMware, please read [Virtualizing JIRA \(JIRA on VMware\)](#).

Next Steps

Installing JIRA applications

Installing Java

Here you will find instructions for installing the Java Development Kit (JDK). This is a manual step that's only required if you're installing a JIRA application from a zip or archive file. If you're using the [Windows installer](#) or [Linux installer](#), you don't need to install Java manually.

Check the [Supported platforms](#) page to find which Java versions are supported for JIRA.

Installing Java

You'll need to install the JDK on the same server that will have your JIRA application.

On Linux and Mac OS X

Before you start, check whether a JDK is already installed.

1. Open a shell console and type `echo $JAVA_HOME` and hit **Enter**:
 - If it returns something like `/opt/JDK7` or `/usr/lib/jvm/java-7` then your JDK is installed and configured
 - If nothing displays, you'll need to install the JDK or set the `$JAVA_HOME` environment variable
2. Check the [Supported platforms](#) page to find out which JDK versions are supported for your version of JIRA.
3. Download the appropriate [JDK version](#).
4. Run the Java installer. Detailed installation instructions are provided on <http://www.oracle.com/technetwork/java/javase/index-137561.html>.
5. Open a shell console and type `echo $JAVA_HOME` and hit **Enter** to check that it has installed correctly.

On Windows

Before you start, check whether a JDK is already installed.

1. Go to **Control Panel > Programs and Features** to see what JDK version is installed.
2. Check the [Supported platforms](#) page to find out which JDK versions are supported for your version of JIRA applications.
3. Download the right [JDK version](#).
4. Run the Java installer. Make a note of the installation directory, as you'll need this later.
5. Open a command prompt and type `echo %JAVA_HOME%` and hit **Enter**:
 - If you see a path to your Java installation directory, the `JAVA_HOME` environment variable has been set correctly.
 - If nothing is displayed, or only `%JAVA_HOME%` is returned, you'll need to set the `JAVA_HOME` environment variable manually.

Set the JAVA_HOME

If you installed the JDK, you'll be setting the `JAVA_HOME` environment variable. If you installed the Java Runtime Environment (JRE), follow the same steps, but set the `JRE_HOME` environment variable instead.

On Linux

The `JAVA_HOME` environment variable is sometimes set in the `/etc/environment` file. You'll need to modify its value to `JAVA_HOME="path/to/JAVA_HOME"`.

1. If `JAVA_HOME` is not defined in this file, set it using the following command at a shell prompt, when logged in with 'root' level permissions:

```
export JAVA_HOME="path/to/JAVA_HOME" >> /etc/environment
```

2. Log out for these changes to apply.

On Mac OS X

The `JAVA_HOME` environment variable is set in the `~/ .bash_profile` file. You'll need modify its value to `JAVA_HOME="path/to/JAVA_HOME"`.

1. If `JAVA_HOME` is not defined in this file, set it using the following command at a shell prompt, when logged in with 'root' level permissions:

```
export JAVA_HOME="path/to/JAVA_HOME" >> ~/.bash_profile
```

2. You'll need to open a new terminal for these changes to apply.

On Windows

1. Locate your Java installation directory, it will be something like `C:\Program Files\Java\jdk1.8.0_65`
2. Do one of the following:
 - a. **Windows 7** – Right click **My Computer** and select **Properties > Advanced**
 - b. **Windows 8** – Go to **Control Panel > System > Advanced System Settings**
 - c. **Windows 10** – Search for **Environment Variables** then select **Edit the system environment variables**
3. Click the **Environment Variables** button.
4. Under **System Variables**, click **New**.
5. In the **Variable Name** field, enter:
 - `JAVA_HOME` if you installed the JDK
 - `JRE_HOME` if you installed the JRE
6. In the **Variable Value** field, enter your JDK or JRE installation path.

For Windows users on 64-bit systems:

```
Progra~1 = 'Program Files'
Progra~2 = 'Program Files(x86)'
```

7. Click **OK** and **Apply Changes** as prompted.
8. You'll need to close and re-open any command windows that were open before you made these changes. If the changes don't take effect after reopening the command window, restart Windows.

If you start JIRA and you get an error like **Windows cannot find '-Xms128m'** you've probably not set `JAVA_HOME` correctly.

Supported platforms

Before installing **JIRA 7.8**, make sure you have the right software and infrastructure to run it. If a platform and version is not listed on this page, it means we don't test it, fix bugs or provide assistance for it.

Definitions:

-  Supported - you can use **JIRA 7.8** with this platform.
-  Limited - you can evaluate **JIRA 7.8** on this platform, but you can't run a production site on it.
-  Deprecated - support for this platform will end in an upcoming release.

Further information:

- Please read [JIRA applications installation requirements](#), since not all the platforms listed below may be required for your specific JIRA setup.

Java

Oracle JRE / JDK:

-  Java 1.8

Good to know:

- We recommend using the Critical Path Update (CPU) releases.
- You don't need to install Java if using the [Windows Installer](#) or [Linux Installer](#), as the JRE is bundled with JIRA.

Operating systems

Operating systems:

- ✔ Microsoft Windows
- ✔ Linux / Solaris
- i Mac OS X
- ✔ Amazon Web Services (AWS)
- ✔ Microsoft Azure

Browsers**Desktop browsers:**

- ✔ Chrome (latest stable version)
- ✔ Internet Explorer 11
- ✔ Microsoft Edge
- ✔ Mozilla Firefox (latest stable version)
- ✔ Safari on Mac OS X only (latest stable version)

Mobile browsers:

- ✔ Chrome (latest stable version)
- ✔ Safari on iOS only (latest stable version)
- ✔ Android 4.0 (Ice Cream Sandwich)

Databases**PostgreSQL:**

- ✔ PostgreSQL 9.6
- ✔ PostgreSQL 9.5
- ✔ PostgreSQL 9.4
- ✔ PostgreSQL 9.3

MySQL:

- ✔ MySQL 5.7

Good to know:

- JIRA is a pure Java-based application and should run on any supported operating system, provided that the JDK / JRE requirements are satisfied.

Microsoft Windows:

- Read [Anti-Virus in JIRA applications](#).

Linux / Solaris:

- We only test the [Linux Installer](#) on CentOS Linux. If you have any problems with the Linux Installer, we recommend [Installing JIRA on Linux from an archive file](#).

AWS:

- We only support JIRA Software and JIRA Service Desk, using PostgreSQL on Amazon RDS.
- Read [Getting started with JIRA Data Center on AWS](#).

Azure:

- We only support JIRA Software, running on Azure SQL.
- Read [Getting started with JIRA Data Center on Azure](#).

Good to know:

- We support a minimum screen resolution of 1024 x 768 (when browsers are maximized).

Internet Explorer:

- 'Compatibility View' won't work on Internet Explorer.

Mobile:

- You can only view JIRA on a mobile device using the mobile view. The native app is only available for Cloud sites.

PostgreSQL:

JIRA is tested and bundled with the 9.4 JDBC driver. You can also use the latest JDBC driver for your [PostgreSQL version](#), though we can't guarantee it will work with your version of JIRA. To use a different JDBC driver:

1. Stop your JIRA instance.
2. Remove the bundled driver from <JIRA_INST>/lib/.
3. Download the new driver and place it in <JIRA_INST>/lib/.

✔ MySQL 5.6

✔ MySQL 5.5

Oracle:

✔ Oracle 12c R1

Microsoft SQL Server:

✔ SQL Server 2014

✔ SQL Server 2012

Embedded database:

i H2 (evaluation only)

4. Restart your JIRA instance.

MySQL:

- JIRA will not work on:
 - 4 byte characters, regardless of MySQL version
 - MariaDB nor PerconaDB
- We recommend running MySQL in strict mode.
- Supported MySQL databases use the [MySQL Connector/J 5.1 driver](#).

Oracle:

- JIRA will not work on:
 - Oracle Advanced Compression Option (ACO)
- Supported Oracle databases use the 12.2.0.1 [driver listed here](#).

Microsoft SQL Server:

- JIRA will not work on:
 - Express Editions of Microsoft SQL Server
- Supported Microsoft SQL databases use the Microsoft JDBC 6.2.1 Driver.

H2:

- JIRA ships with a built-in database (H2).

Environment and Infrastructure

Hardware

- NFS mounts are supported only for JIRA Data Center's shared home directory. They won't work for Server, or DC's local home directory due to Lucene requirements. Read the [IndexWriter docs](#) for more information.
- We only support JIRA running on x86 hardware and 64-bit derivatives of x86 hardware.
- If you are installing JIRA from an archive, create a dedicated user account on the operating system to run JIRA, since JIRA runs as the user it is invoked under, it can potentially be abused.

Virtualization

- VMware supports all of the operating systems listed under 'Operating systems'.
- We don't provide support for VMWare itself.
- Read [Virtualizing JIRA](#) for information on how to configure VMWare.

Application server

- We support Apache Tomcat 8.5.6.
- We don't support deploying multiple Atlassian applications in a single Tomcat container.

Internet protocols (IP)

- We support IPv4.
- We don't support IPv6 (see [JRASERVER-36676 - IPv6 Support for JIRA](#) **RESOLVED**).

External user directories

You can manage users in the following LDAP directories:

- Apache Directory Server 1.0.x
- Apache Directory Server 1.5.x
- Apple Open Directory (Read-Only)
- FedoraDS (Read-Only Posix Schema)
- Generic Directory Server
- Generic Posix/RFC2307 Directory (Read-Only)
- Microsoft Active Directory
- Novell eDirectory Server
- OpenDS
- Open LDAP
- Open LDAP (Read-Only Posix Schema)
- Sun Directory Server Enterprise Edition

Mail servers

- SMTP servers must be able to support the multipart content type.

End of support announcements

This page contains announcements of the end of support for various platforms and browsers used with JIRA. These are summarized for upcoming JIRA releases in the table below. Please see the following sections for the full announcements.

End of support matrix for JIRA

We're not planning any changes in the next releases. You can see previous announcements below.

Why is Atlassian ending support for these platforms?

Atlassian is committed to delivering improvements and bug fixes as fast as possible. We are also committed to providing world class support for all the platforms our customers run our software on. However, as new versions of databases, web browsers, etc, are released, the cost of supporting multiple platforms grows exponentially, making it harder to provide the level of support our customers have come to expect from us. Therefore, we no longer support platform versions marked as end-of-life by the vendor, or very old versions that are no longer widely used.

Previous announcements

Platform/Functionality	JIRA end of support
Postgres 9.2	From JIRA 7.4 (announcement)
Internet Explorer 10	From JIRA 7.2 (announcement)
Microsoft SQL Server 2008	From JIRA 7.2 (announcement)
Postgres 9.0 and 9.1	From JIRA 7.2 (announcement)
MySQL 5.1	From JIRA 7.2 (announcement)
HSQLDB	From JIRA 7.0 (announcement)
Oracle JDK 1.7	From JIRA 7.0 (announcement)
Oracle 11G	From JIRA 7.0 (announcement)
Internet Explorer 9	From JIRA 7.0 (announcement)
SOAP API (replaced with REST)	From JIRA 7.0 (announcement)
Jelly script	From JIRA 6.4 (announcement)

WAR download distribution	From JIRA 7.0 (announcement)
Microsoft SQL Server 2005	From JIRA 7.0 (announcement)

End of support for Postgres 9.2

Announced May 2017

Atlassian will end support for Postgres 9.2 in **JIRA 7.4**. End of support means that Atlassian will not fix bugs related to Postgres 9.2 past the support end date.

We are making this decision in order to reduce our database testing and support, and help us speed up our ability to deliver market-driven features. If you have questions or concerns regarding this announcement, please email [eol-announcement at atlassian dot com](mailto:eol-announcement@atlassian.com).

Postgres 9.2 End of Support Notes:

- JIRA 7.3 will be the last major version of JIRA to officially support Postgres 9.2.
- JIRA 7.3.x and earlier versions should continue to work with Postgres 9.2.
- JIRA 7.4 will not be tested against Postgres 9.2.
- Postgres 9.3, 9.4 and 9.5 will continue to be supported in JIRA 7.4.x (see [Supported platforms](#)).

End of support for Internet Explorer 10

Announced May 2016

Atlassian will end support for Internet Explorer 10 (IE10) in **JIRA 7.2**. End of support means that Atlassian will not fix bugs related to IE10 past the support end date.

We are making this decision in order to reduce our browser testing and support, and help us speed up our ability to deliver market-driven features. If you have questions or concerns regarding this announcement, please email [eol-announcement at atlassian dot com](mailto:eol-announcement@atlassian.com).

Internet Explorer 10 (IE10) End of Support Notes:

- JIRA 7.1 will be the last major version of JIRA to officially support IE10.
- JIRA 7.1.x and earlier versions should continue to work with IE10.
- JIRA 7.2 will not be tested against IE10.
- Internet Explorer 11 will continue to be supported in JIRA 7.2.x (see [Supported platforms](#)).

End of support for Microsoft SQL Server 2008

Announced October 2015

Atlassian will end support for Microsoft SQL Server 2008 in **JIRA 7.2**. End of support means that Atlassian will not fix bugs related to Microsoft SQL Server 2008 past the support end date.

We are making this decision in order to reduce our database testing and support, and help us speed up our ability to deliver market-driven features. If you have questions or concerns regarding this announcement, please email [eol-announcement at atlassian dot com](mailto:eol-announcement@atlassian.com).

Microsoft SQL Server 2008 End of Support Notes:

- JIRA 7.1 will be the last major version of JIRA to officially support Microsoft SQL Server 2008.
- JIRA 7.1.x and earlier versions should continue to work with Microsoft SQL Server 2008.
- JIRA 7.2 will not be tested against Microsoft SQL Server 2008.
- Microsoft SQL Server 2012 will continue to be supported in JIRA 7.2.x (see [Supported platforms](#)).

End of support for Postgres 9.0 and 9.1

Announced October 2015

Atlassian will end support for Postgres 9.0 and Postgres 9.1 in **JIRA 7.2**. End of support means

that Atlassian will not fix bugs related to Postgres 9.0 or 9.1 past the support end date.

We are making this decision in order to reduce our database testing and support, and help us speed up our ability to deliver market-driven features. If you have questions or concerns regarding this announcement, please email [eol-announcement at atlassian dot com](mailto:eol-announcement@atlassian.com).

Postgres 9.0 and Postgres 9.1 End of Support Notes:

- JIRA 7.1 will be the last major version of JIRA to officially support Postgres 9.0 and 9.1.
- JIRA 7.1.x and earlier versions should continue to work with Postgres 9.0 and 9.1.
- JIRA 7.2 will not be tested against Postgres 9.0 or 9.1.
- Postgres 9.2 and Postgres 9.3 will continue to be supported in JIRA 7.2.x (see [Supported platforms](#)).

End of support for MySQL 5.1

Announced October 2015

Atlassian will end support for MySQL 5.1 in **JIRA 7.2**. End of support means that Atlassian will not fix bugs related to MySQL 5.1 past the support end date.

We are making this decision in order to reduce our database testing and support, and help us speed up our ability to deliver market-driven features. If you have questions or concerns regarding this announcement, please email [eol-announcement at atlassian dot com](mailto:eol-announcement@atlassian.com).

MySQL 5.1, 5.2, 5.3 and 5.4 End of Support Notes:

- JIRA 7.1 will be the last major version of JIRA to officially support MySQL 5.1.
- JIRA 7.1.x and earlier versions should continue to work with MySQL 5.1.
- JIRA 7.2 will not be tested against MySQL 5.1.
- MySQL 5.5 and 5.6 will continue to be supported in JIRA 7.2.x (see [Supported platforms](#)).

End of support for HSQLDB

Announced February 2015

Atlassian will end support for HSQLDB (HyperSQL DataBase) in **JIRA 7.0**. End of support means that Atlassian will not fix bugs in HSQLDB past the support end date.

JIRA ships with a built-in database for evaluation purposes, and currently this is HSQLDB. As of **JIRA 7.0**, JIRA will ship with H2 (H2 Database Engine) as its built-in database.

HSQLDB (HyperSQL DataBase or HSQLDB) End of Support Notes:

- JIRA 6.4 will be the last major version of JIRA to officially support HSQLDB (HyperSQL DataBase) for evaluation use.
- JIRA 6.4.x and earlier versions will continue to work with HSQLDB (HyperSQL DataBase) for evaluation use. However, we will not fix bugs affecting HSQLDB (HyperSQL DataBase) past the support end date.
- JIRA 7.0 will not be tested with HSQLDB (HyperSQL DataBase).

End of support for Oracle JDK 1.7

Announced February 2015

Atlassian will end support for Java 7 (JRE and JDK 1.7) in **JIRA 7.0**. End of support means that Atlassian will not fix bugs in Java 7 (JRE and JDK 1.7) past the support end date.

We are ending support for Java 7 (JRE and JDK 1.7), as Oracle Corporation has announced the end of public updates for Java 7: [Java SE 7 End of Public Updates Notice](#).

Java 7 (JRE and JDK 1.7) End of Support Notes:

- JIRA 6.4 will be the last major version of JIRA to officially support Java 7 (JRE and JDK 1.7).
- JIRA 6.4.x and earlier versions will continue to work with Java 7 (JRE and JDK 1.7). However, we

- will not fix bugs affecting Java 7 (JRE and JDK 1.7) past the support end date.
- JIRA 7.0 will not be tested with Java 7 (JRE and JDK 1.7).
- Java 8 (JRE and JDK 1.8) is supported, but not bundled with JIRA 6.4

End of support for Oracle 11G

Announced February 2015

Atlassian will end support for Oracle 11G in **JIRA 7.0**. End of support means Atlassian will not fix bugs related to Oracle 11G past the support end date, except for security-related issues.

We are making this decision as Oracle Corporation have ended support for Oracle 11G as of [January 2015](#). Testing on Oracle 12C will conclude shortly and we'll announce support soon.

Oracle 11G End of Support Notes

- JIRA 6.4 will be the last major release that supports Oracle 11G
- JIRA 6.4.x and earlier versions will continue to work on Oracle 11G
- JIRA 7.0 will not be tested against Oracle 11G

End of support for Internet Explorer 9

Announced February 2015

Atlassian will end support for Internet Explorer 9 in **JIRA 7.0**. End of support means that Atlassian will not fix bugs related to Internet Explorer 9 past the support end date, except for security-related issues.

We are making this decision to enable us to provide the best user experience to our customers, accelerate our pace of innovation, and give us the ability to utilize modern browser technologies.

Internet Explorer 9 (IE9) End of Support Notes

- JIRA 6.4 will be the last major release that supports Internet Explorer 9
- JIRA 6.4.x and earlier versions should continue to work on Internet Explorer 9
- JIRA 7.0 will not be tested against Internet Explorer 9
- Internet Explorer 10 and Internet Explorer 11 will continue to be supported in JIRA 7.0.x.

End of support for SOAP

Announced November 2014

Atlassian will end support for SOAP API in **JIRA 7.0**. The SOAP API's have been replaced by [REST API's](#) as Atlassian's recommended and supported remote API.

SOAP End of Support Notes

- JIRA 6.4 will be the last major release that supports SOAP
- JIRA 6.4.x and earlier versions should continue to work with SOAP
- JIRA 7.0 will not include any SOAP API's
- If you need an alternative that Atlassian supports, the [REST API](#) is fully supported by JIRA.

End of support for Jelly Scripts

Announced November 2014

Atlassian will end support for Jelly scripts in **JIRA 6.4**. If you are using Jelly scripts with JIRA, we suggest you move to [Groovy Script Runner](#) or utilize the [JIRA Command Line Interface](#), which will provide you with more flexible options.

Jelly Script End of Support Notes

- JIRA 6.3 will be the last major release to support Jelly scripts
- JIRA 6.3.x and earlier versions should continue to work fine with Jelly scripts

- JIRA 6.4 will not include Jelly.
- If you need an alternative to Jelly scripts, [Groovy Script Runner](#) or the [JIRA Command Line Interface](#) are the suggested alternatives that work with JIRA.

End of support for WAR distribution

Announced August 2014

Atlassian will stop releasing the WAR distribution of JIRA in **JIRA 7.0**.

Why are we ending support for this?

- We are trying to reduce the amount of combinations and confusion around this for customers downloading a Server (BTF) edition
- The WAR edition is a bit more complex to install and gets more difficult as the installation ages and gets bigger - we want to reduce that complexity
- We can't and don't test every permutation of environments + app servers that a customer might deploy into, nor can we control what else might be in that environment, which can lead to a poor user experience
- We only support Tomcat - JIRA doesn't work on WLS or WebSphere anyways, other app servers - maybe.

Anything we release, we want to make sure users get a good experience in installation and usage and don't have to deal with app server quirks etc.

End of support for Microsoft SQL Server 2005

Announced June 2014

Atlassian will end support for Microsoft SQL Server 2005 in **JIRA 7.0**. End of support means that Atlassian will not fix bugs related to Microsoft SQL Server 2005 past the support end date.

We are making this decision in order to reduce our database testing and support, and help us speed up our ability to deliver market-driven features. If you have questions or concerns regarding this announcement, please email eol-announcement@atlassian.com.

Microsoft SQL Server 2005 End of Support Notes:

- JIRA 6.4 will be the last major version of JIRA to officially support Microsoft SQL Server 2005.
- JIRA 6.4.x and earlier versions should continue to work with Microsoft SQL Server 2005.
- JIRA 7.0 will not be tested against Microsoft SQL Server 2005.
- Microsoft SQL Server 2008 and 2008 R2 will continue to be supported in JIRA 7.0.x (see [Supported Platforms](#)).
- We will start supporting Microsoft SQL Server 2012 in JIRA 6.4.

Evaluation installation

Want to get up and running with a JIRA application? This page will guide you through three simple steps to install and set up an evaluation JIRA site.

On this page:

[Before you begin](#)

1. [Download the installer](#)
2. [Install your JIRA application](#)
3. [Set up your JIRA application](#)

If you're ready to set up a production JIRA site or you want more control, check out our [full installation guides](#).

Before you begin

Our installers come with all the bits and pieces you need to run the application, but there's a few things you'll need to get up and running:

- A computer or laptop with a supported operating system - you'll be installing JIRA so you'll need admin

rights.

▼ Supported operating systems...

You can install JIRA on a Windows or Linux operating system.

Apple Mac isn't supported for production sites, but if you're comfortable setting up applications on your Mac from scratch, you can download the `tar.gz` file and follow the instructions for [Installing JIRA applications on Linux from Archive File](#) as the process is similar.

- A supported web browser - you'll need this to access JIRA, we support the latest versions of Chrome and Mozilla Firefox, Internet Explorer 11, and Microsoft Edge.
- A valid email address - you'll need this to generate your evaluation license and create an account.

Ready to get going? Let's start with grabbing the installer.

1. Download the installer

Download the installer for your operating system:

- JIRA Core at <https://www.atlassian.com/software/jira/core/download>
- JIRA Software at <https://www.atlassian.com/software/jira/download>
- JIRA Service Desk at <https://www.atlassian.com/software/jira/service-desk/download>

2. Install your JIRA application

The installer allows you to choose Express or Custom installations.

The Custom installation allows you to pick some specific options for JIRA, but for this guide we'll use the Express installation.

▼ For Windows

1. Run the installer - we recommend running with a Windows administrator account. If prompted, make sure you allow the installer to make changes to your computer. This will allow you to install JIRA as a service.
2. Choose **Express Install**, then click **Next**.
3. Once installation is complete, it will ask you if you want to open JIRA in your browser. Make sure this option is selected then click **Done**.
4. JIRA will open in your default browser, and you're ready to start the set up wizard.

▼ For Linux

1. Change to the directory where you downloaded JIRA then execute this command to make it executable:

JIRA Core

```
$ chmod a+x atlassian-jira-core-X.X.X-x64.bin
```

JIRA Software

```
$ chmod a+x atlassian-jira-software-X.X.X-x64.bin
```

JIRA Service Desk

```
$ chmod a+x atlassian-servicedesk-X.X.X-x64.bin
```

Where `-x.x.x` is the JIRA version you downloaded.

2. Run the installer, we recommend using `sudo` to run the installer as this will create a dedicated account to run JIRA and allow you to run JIRA as a service.

JIRA Core

```
$ sudo ./atlassian-jira-core-X.X.X-x64.bin
```

JIRA Software

```
$ sudo ./atlassian-jira-software-X.X.X-x64.bin
```

JIRA Service Desk

```
$ sudo ./atlassian-servicedesk-X.X.X-x64.bin
```

3. When prompted, choose **Express Install** (option 1).
4. Once installation is complete head to <http://localhost:8080> in your browser to begin the setup process.

3. Set up your JIRA application

The set up wizard is the last step in getting JIRA up and running. You'll need your email address to generate your evaluation license.

1. Select **Set it up for me** and then **Continue to MyAtlassian**.
This will allow JIRA to set up everything it needs to run, including an H2 database.
2. Create an account (or log in if you already have an Atlassian ID account).
3. Follow the prompts to **generate a license** for the JIRA application you want to try, and apply it to your new installation.
4. Enter and confirm the details you want to use for your administrator account, and click **Next**.
5. It will take a few minutes to get everything connected and operational.

That's it! You're ready to team up with some colleagues and start using JIRA.

Installing JIRA applications on Windows

In this guide we'll run you through installing a JIRA application in a production environment, with an external database, using the Windows installer.

This is the most straightforward way to get your production site up and running on a Windows server.

Other ways to install JIRA:

- [Evaluation](#) - get your free trial up and running in no time.
- [Zip](#) – install JIRA manually from a zip file.
- [Linux](#) – install JIRA on a Linux operating system

On this page:

- [Before you begin](#)
- [Install a JIRA application](#)
 - [1. Download JIRA](#)
 - [2. Run the installer](#)

- Set up your JIRA application
 - 3. Choose set up method
 - 4. Connect to your database
 - 5. Set application properties
 - 6. Enter your license
 - 7. Create your administrator account
 - 8. Set up email notifications
 - 9. Start using JIRA
- Troubleshooting

Before you begin

Before you install JIRA, there's a few questions you need to answer.

<p>Are you using a supported operating system?</p>	<p>Tell me more...</p> <p>Check the Supported platforms page for the version of JIRA you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • We don't support installing JIRA on OSX or mac OS. • The JIRA installer includes Java (JRE) and Tomcat, so you don't need to install these seperately.
<p>Do you want to run JIRA as a Windows Service?</p>	<p>Tell me more...</p> <p>Running JIRA as a service in Windows means that your JIRA application will automatically start up when Windows is started.</p> <p>If you choose to run JIRA as a service:</p> <ul style="list-style-type: none"> • You must run the installer as administrator to be able to install JIRA as a service. • The JIRA service will be run as the Windows 'SYSTEM' user account. • We strongly recommend creating a dedicated user account (e.g. with username 'jira') on Windows for running JIRA. <p>See Running JIRA applications as a Window's service for more information.</p> <p>If you choose not to run JIRA as a service:</p> <ul style="list-style-type: none"> • You will start and stop JIRA by running the <code>start-jira.bat</code> file in your JIRA installation directory. • JIRA will be run as the Windows user account that was used to install JIRA, or you can choose to run as a dedicated user. • JIRA will need to be restarted manually if your server is restarted.
<p>Is your database set up and ready to use?</p>	<p>Tell me more...</p> <p>To run JIRA in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support. If you don't already have a database, PostgreSQL is free, easy to set up and has been extensively tested with JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, MySQL, and SQL Server. • Use UTF-8 character encoding. • If you're using Oracle or MySQL you'll need to download the driver for your database. • The embedded H2 database can be used for evaluating JIRA, but you'll need to migrate to another database before running in production. You may find it easier to use external database from the start.

Do you have a JIRA license?	<p>▼ Tell me more...</p> <p>You'll need a valid JIRA Software Server, JIRA Core Server or JIRA Service Desk Server license to use JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you have not yet purchased a JIRA application license you'll be able to create an evaluation license during setup. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from JIRA Cloud, you'll need a new license.
-----------------------------	---

Install a JIRA application

1. Download JIRA

Download the installer for your operating system:

- JIRA Core at <https://www.atlassian.com/software/jira/core/download>
- JIRA Software at <https://www.atlassian.com/software/jira/download>
- JIRA Service Desk at <https://www.atlassian.com/software/jira/service-desk/download>

2. Run the installer

1. Run the installer. We recommend using a Windows administrator account.
2. Follow the prompts to install JIRA. You'll be asked for the following info:
 - a. **Destination directory** – this is where JIRA will be installed.
 - b. **Home directory** – this is where JIRA data like logs, search indexes and files will be stored.
 - c. **TCP ports** – these are the HTTP connector port and control port JIRA will run on. Stick with the default unless you're running another application on the same port.
 - d. **Install as service** – this option is only available if you ran the installer as administrator.
3. JIRA will start up in your browser once installation is complete.

Set up your JIRA application

3. Choose set up method

Choose **I'll set it up myself**.

4. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.
2. Choose **My own database**.
3. Choose your database type then enter the details for your database.

▼ [Show me how to do this...](#)

JIRA connects to your database using a standard JDBC database connection. Connection pooling is handled within JIRA, you can change this using [JIRA configuration tool](#) later.

If you're using Oracle or MySQL there's an extra step:

- Download and extract the appropriate database JDBC drivers. See [Supported platforms](#) to get the right version.
- Drop the JAR file into your `<jira-installation>/lib` folder before continuing with the setup wizard.

In the setup wizard:

- **Driver Class Name** – the Java class name for your database driver. If you're not sure, check the documentation for your database.
- **Database URL** – the JDBC URL for your database. If you're not sure, check the documentation for your database.
- **Username** and **Password** – A valid username and password that JIRA can use to access your database.

5. Set application properties

1. Give your JIRA site a name.
2. Choose whether your site should be private or if anyone can sign up. You can change this later.
3. Enter your base URL - this is the address people will use to access your JIRA site.

6. Enter your license

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

7. Create your administrator account

Enter details for the administrator account. You can add more administrators after set up is complete.

8. Set up email notifications

Enter details of your mail server. This will allow JIRA to send notifications when issues change.

9. Start using JIRA

That's it! Your JIRA site is accessible from your base URL or a URL like this: `http://<computer_name_or_IP_address>:<port>`

Here's a few things that will help you get your team up and running:

- [Add and invite users](#) to get your team on board, or [configure user directories](#) for slightly bigger teams.
- [Create your first project](#) to have something to work on.
- [Configure SSL or HTTPS](#) to keep JIRA and your team more secure.

Troubleshooting

▼ [Running into problems installing JIRA?](#)

Some anti-virus or other Internet security tools may interfere with the JIRA installation process and prevent the process from completing successfully. If you experience or anticipate experiencing such an issue with your anti-virus/Internet security tool, disable this tool first before proceeding with the JIRA installation.

Head to [Installation Troubleshooting](#) in our Knowledge Base for more help.

Uninstalling JIRA applications from Windows

This page describes the procedure for uninstalling JIRA, which had been installed using the [Windows Installer](#).

i If you wish to re-install JIRA in 'unattended mode', do not uninstall your previous installation of JIRA just yet. See [Using the silent installation feature](#) for more information.

To uninstall JIRA from Windows:

1. Log in to Windows as the same user that was used to install JIRA with the [Windows Installer](#).
2. Start the uninstaller by doing one of the following:
 - Click the Windows **'Start'** menu > **'All Programs'** > 'JIRA X.Y' > **'Uninstall JIRA X.Y'** (where 'X.Y' refers to the installed version of JIRA that you are about to uninstall)

OR

- Open the Windows Control Panel, choose '**Add or Remove Programs**' (on Windows XP) or '**Programs and Features**' on (Windows 7/Vista), and then uninstall 'JIRA X.Y' from the list of applications
 - OR**
 - Open the Windows command prompt, and do the following:
 - a. Change directory `cd` to your JIRA installation directory
 - b. Run the `uninstall.exe` file
3. Follow the prompts to uninstall JIRA from your computer.

i Please note:

- The uninstaller will not delete the JIRA home directory.
- All log files that were generated while JIRA was running will not be deleted.
- All files within the JIRA installation directory will be deleted (with the exception of the Tomcat `log` folder located in the JIRA installation directory).
- The uninstaller can be made to operate in unattended mode by specifying the `-q` option at the Windows command prompt — i.e. `uninstall.exe -q`

Installing JIRA applications on Windows from Zip File

In this guide we'll run you through installing a JIRA application in a production environment, with an external database, manually using a zip file.

This method gives you the most control of the installation process.

Other ways to install JIRA:

- [Evaluation](#) - get your free trial up and running in no time.
- [Installer](#) – install JIRA using the Windows installer.
- [Linux](#) – install JIRA on a Linux operating system.

On this page:

Before you begin

Install a JIRA application

1. Download JIRA
2. Create the installation directory
3. Create the home directory
4. Check the ports
5. Start JIRA

Set up your JIRA application

6. Choose set up method
7. Connect to your database
8. Set application properties
9. Enter your license
10. Create your administrator account
11. Set up email notifications
12. Start using JIRA

Troubleshooting

Before you begin

Before you install JIRA, there's a few questions you need to answer.

<p>Are you using a supported operating system and Java version?</p>	<p>Tell me more about this...</p> <p>Check the Supported platforms page for the version of JIRA you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • We don't support installing JIRA on OSX. • We don't support OpenJDK. You'll need to install Oracle Java. • You can use either the JDK (Java Development Kit) or JRE (Java Runtime Environment). • We only support the version of Apache Tomcat that is bundled with JIRA.
---	--

<p>Do you want to run JIRA as a Windows Service?</p>	<p>▼ Tell me more about this...</p> <p>Running JIRA as a service in Windows means that your JIRA application will automatically start up when Windows is started.</p> <p>You should use the Windows installer if you want to run JIRA as a Service.</p> <p>If you choose not to run JIRA as a service:</p> <ul style="list-style-type: none"> • You will start and stop JIRA by running the <code>start-jira.bat</code> file in your JIRA installation directory. • JIRA will be run as the Windows user account that was used to install JIRA, or you can choose to run as a dedicated user (this user must have full read and write access to the installation directory and home directory). • JIRA will need to be restarted manually if your server is restarted.
<p>What database do you plan to use?</p>	<p>Tell me more about this...</p> <p>To run JIRA in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support. If you don't already have a database, PostgreSQL is free, easy to set up and has been extensively tested with JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, MySQL, and SQL Server. • Use UTF-8 character encoding. • If you're using Oracle or MySQL you'll need to download the driver for your database. • The embedded H2 database can be used for evaluating JIRA, but you'll need to migrate to another database before running in production. You may find it easier to use external database from the start.
<p>Do you have a JIRA license?</p>	<p>Tell me more about this...</p> <p>You'll need a valid JIRA Software Server, JIRA Core Server or JIRA Service Desk Server license to use JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you have not yet purchased a JIRA application license you'll be able to create an evaluation license during setup. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from JIRA Cloud, you'll need a new license.
<p>Is your JAVA_HOME variable set correctly?</p>	<p>Tell me more about this...</p> <p>Before you install JIRA, check that you're running a supported Java version and that the <code>JAVA_HOME</code> environment variable is set correctly.</p> <p>JIRA applications can only run with Oracle JDK or JRE.</p> <p>To check the <code>JAVA_HOME</code> variable:</p> <p>Open a command prompt and type <code>echo %JAVA_HOME%</code> and hit Enter.</p> <ul style="list-style-type: none"> • If you see a path to your Java installation directory, the <code>JAVA_HOME</code> environment variable has been set correctly. • If nothing is displayed, or only <code>%JAVA_HOME%</code> is returned, you'll need to set the <code>JAVA_HOME</code> environment variable manually.

Install a JIRA application

1. Download JIRA

Download the zip file for your operating system:

- JIRA Core at <https://www.atlassian.com/software/jira/core/download>
- JIRA Software at <https://www.atlassian.com/software/jira/download>
- JIRA Service Desk at <https://www.atlassian.com/software/jira/service-desk/download>

2. Create the installation directory

1. Create your installation directory (with full control permission) – this is where JIRA will be installed. Avoid using spaces or special characters in the path. We'll refer to this directory as your <installation-directory>.
2. Extract the JIRA zip file to your <installation-directory>. We recommend using [7zip](#) or [Winzip](#).

3. Create the home directory

1. Create your home directory (with full control permission) – this is where JIRA data like logs, search indexes and files will be stored. This should be separate to your installation directory. We'll refer to this directory as your <home-directory>.
2. Tell JIRA where to find your <home-directory> when it starts up. There are two ways to do this:

▼ Edit the `jira-application.properties` file...

Edit <installation-directory>\atlassian-jira\WEB-INF\classes\jira-application.properties file in any text editor.

After `jira.home` add the absolute path to your home directory. You will need to escape the backslashes, for example:

```
jira.home=X:\\path\\to\\jira-home
```

If you define an UNC path you will need to double escape the leading backslash, for example:

```
jira.home=\\\\\\machinename\\path\\to\\jira-home
```

▼ Set an environment variable...

You can set an environment variable named `JIRA_HOME` in your operating system with the absolute path to your <home-directory>.

Open Command Prompt and execute the following:

```
set JIRA_HOME=X:\path\to\jira-home
```

where `x` is the drive where you created your <home-directory>.

You can then specify the command above in a script used to start JIRA.

4. Check the ports

By default JIRA listens on port 8080. If you have another application running on your server that uses the same ports, you'll need to tell JIRA to use a different port.

▼ Show me how to do this...

To change the ports:

1. Edit <installation-directory>\conf\server.xml
2. Change the **Server** port (8005) and the **Connector** port (8080) to free ports on your server.

In the example below we've changed the **Server** port to 5005 and the **Connector** port to 5050.

```
<Server port="5005" shutdown="SHUTDOWN">
...
  <Service name="Catalina">
    <Connector port="5050"
      maxThreads="150"
      minSpareThreads="25"
      connectionTimeout="20000"
      enableLookups="false"
      maxHttpHeaderSize="8192"
      protocol="HTTP/1.1"
      useBodyEncodingForURI="true"
      redirectPort="8443"
      acceptCount="100"
      disableUploadTimeout="true" />
```

5. Start JIRA

1. Run `<installation-directory>/start-jira.bat` to start the install process.

A command prompt will open. Closing this window will stop JIRA.

2. Go to <http://localhost:8080/> to launch JIRA in your browser (change the port if you've updated the Connector port).

▼ Trouble starting JIRA?

- If the command prompt window closes immediately, your `JAVA_HOME` variable may not be set correctly.

Set up your JIRA application

6. Choose set up method

Choose **I'll set it up myself**.

7. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.
2. Choose **My own database**.
3. Choose your database type then enter the details for your database.

▼ Show me how to do this...

JIRA connects to your database using a standard JDBC database connection. Connection pooling is handled within JIRA, you can change this using [JIRA configuration tool](#) later.

If you're using Oracle or MySQL there's an extra step:

- Download and extract the appropriate database JDBC drivers. See [Supported platforms](#) to get the right version.
- Drop the JAR file into your `<jira-installation>/lib` folder before continuing with the setup wizard.

In the setup wizard:

- **Driver Class Name** – the Java class name for your database driver. If you're not sure, check the documentation for your database.
- **Database URL** – the JDBC URL for your database. If you're not sure, check the documentation for your database.
- **Username** and **Password** – A valid username and password that JIRA can use to access your database.

8. Set application properties

1. Give your JIRA site a name.
2. Choose whether your site should be private or if anyone can sign up. You can change this later.
3. Enter your base URL - this is the address people will use to access your JIRA site.

9. Enter your license

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

10. Create your administrator account

Enter details for the administrator account. You can add more administrators after set up is complete.

11. Set up email notifications

Enter details of your mail server. This will allow JIRA to send notifications when issues change.

12. Start using JIRA

That's it! Your JIRA site is accessible from your base URL or a URL like this: `http://<computer_name_or_IP_address>:<port>`

Here's a few things that will help you get your team up and running:

- [Add and invite users](#) to get your team on board, or [configure user directories](#) for slightly bigger teams.
- [Create your first project](#) to have something to work on.
- [Configure SSL or HTTPS](#) to keep JIRA and your team more secure.

Troubleshooting

▼ [Running into problems installing JIRA?](#)

- If your web browser window shows an error the first time you try to access JIRA, wait for 30 seconds or so and then refresh the page.
- If the command prompt window closes immediately, your `JAVA_HOME` variable may not be set correctly.

Head to [Installation Troubleshooting](#) in our Knowledge Base for more help.

Installing JIRA applications on Linux

In this guide we'll run you through installing a JIRA application in a production environment, with an external database, using the Linux installer.

This is the most straightforward way to get your production site up and running on a Linux server.

Other ways to install JIRA:

- [Evaluation](#) - get your free trial up and running in no time.
- [TAR.GZ](#) – install JIRA manually from an archive file.
- [Windows](#) – install JIRA on a Windows server.

On this page:

- Before you begin
- Install a JIRA application
 1. Download JIRA
 2. Run the installer
- Set up your JIRA application
 3. Choose set up method
 4. Connect to your database
 5. Set application properties
 5. Enter your license
 6. Create your administrator account
 7. Set up email notifications
 8. Start using JIRA
- Troubleshooting

Before you begin

Before you install JIRA, there are a few questions you need to answer.

<p>Are you using a supported operating system?</p>	<p>Tell me more...</p> <p>Check the Supported Platforms page for the version of JIRA you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • We don't support installing JIRA on OSX or mac OS for production sites. • The JIRA installer includes Java (JRE) and Tomcat, so you don't need to install these separately. • JIRA can't run on OpenJDK.
<p>Do you want to run JIRA as a service?</p>	<p>Tell me more...</p> <p>Running JIRA as a service means that JIRA will automatically start up when Linux is started.</p> <p>If you choose to run JIRA as a service:</p> <ul style="list-style-type: none"> • You must use <code>sudo</code> to run the installer to be able to install JIRA as a service. • The installer will create a dedicated user account, <code>jira</code>, that will run the service. <p>If you choose not to run JIRA as a service:</p> <ul style="list-style-type: none"> • You will start and stop JIRA by running the <code>start-jira.sh</code> file in your JIRA installation directory. • JIRA will be run as the user account that was used to install JIRA, or you can choose to run as a dedicated user. • JIRA will need to be restarted manually if your server is restarted.

Is your database set up and ready to use?	<p>▼ Tell me more...</p> <p>To run JIRA in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support. If you don't already have a database, PostgreSQL is free, easy to set up and has been extensively tested with JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, MySQL, and SQL Server. • Use UTF-8 character encoding. • If you're using Oracle or MySQL you'll need to download the driver for your database. • The embedded H2 database can be used for evaluating JIRA, but you'll need to migrate to another database before running in production. You may find it easier to use external database from the start.
Do you have a JIRA license?	<p>Tell me more...</p> <p>You'll need a valid JIRA Software Server, JIRA Core Server or JIRA Service Desk Server license to use JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you have not yet purchased a JIRA application license you'll be able to create an evaluation license during setup. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from JIRA Cloud, you'll need a new license.
Check some known issues	<p>Tell me more...</p> <p>For Linux installations, we've noticed some problems when displaying certain system text in the application (CAPTCHA and gadgets). Instead of showing regular alphanumeric letters, the text will appear to be garbled and look like symbols. To avoid this problem, you should install several fonts that are required by JIRA. For more info, see JIRA UI shows unreadable text.</p>

Install a JIRA application

1. Download JIRA

Download the installer for your operating system:

- JIRA Core at <https://www.atlassian.com/software/jira/core/download>
- JIRA Software at <https://www.atlassian.com/software/jira/download>
- JIRA Service Desk at <https://www.atlassian.com/software/jira/service-desk/download>

2. Run the installer

1. Make the installer executable.

▼ [Show me how to do this...](#)

Change to the directory where you downloaded JIRA then execute this command:

```


JIRA Core


$ chmod a+x atlassian-jira-core-X.X.X-x64.bin
```

JIRA Software

```
$ chmod a+x atlassian-jira-software-X.X.X-x64.bin
```

JIRA Service Desk

```
$ chmod a+x atlassian-servicedesk-X.X.X-x64.bin
```

Where `-x.x.x` is the JIRA version you downloaded.

2. Run the installer, we recommend using `sudo` to run the installer as this will create a dedicated account to run JIRA and allow you to run JIRA as a service.

▼ [Show me how to do this...](#)

To use `sudo` to run the installer execute this command:

JIRA Core

```
$ sudo ./atlassian-jira-core-X.X.X-x64.bin
```

JIRA Software

```
$ sudo ./atlassian-jira-software-X.X.X-x64.bin
```

JIRA Service Desk

```
$ sudo ./atlassian-servicedesk-X.X.X-x64.bin
```

Where `-x.x.x` is the JIRA version you downloaded.

You can also choose to run the installer as with root user privileges.

3. Follow the prompts to install JIRA. You'll be asked for the following info:
 - **Install type** – choose option 2 (custom) for the most control.
 - **Destination directory** – this is where JIRA will be installed.
 - **Home directory** – this is where JIRA data like logs, search indexes and files will be stored.
 - **TCP ports** – these are the HTTP connector port and control port JIRA will run on. Stick with the default unless you're running another application on the same port.
 - **Install as service** – this option is only available if you ran the installer as `sudo`.
4. Once installation is complete head to <http://localhost:8080> in your browser to begin the setup process. (Replace 8080 if you chose a different port during installation) .

Set up your JIRA application

3. Choose set up method

Choose **I'll set it up myself.**

4. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.

2. Choose **My own database**.
3. Choose your database type then enter the details for your database.

▼ [Show me how to do this...](#)

JIRA connects to your database using a standard JDBC database connection. Connection pooling is handled within JIRA, you can change this using [JIRA configuration tool](#) later.

If you're using Oracle or MySQL there's an extra step:

- Download and extract the appropriate database JDBC drivers. See [Supported platforms](#) to get the right version.
- Drop the JAR file into your `<jira-installation>/lib` folder before continuing with the setup wizard.

In the setup wizard:

- **Driver Class Name** – the Java class name for your database driver. If you're not sure, check the documentation for your database.
- **Database URL** – the JDBC URL for your database. If you're not sure, check the documentation for your database.
- **Username** and **Password** – A valid username and password that JIRA can use to access your database.

5. Set application properties

1. Give your JIRA site a name.
2. Choose whether your site should be private or if anyone can sign up. You can change this later.
3. Enter your base URL - this is the address people will use to access your JIRA site.

5. Enter your license

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

6. Create your administrator account

Enter details for the administrator account. You can add more administrators after setup is complete.

7. Set up email notifications

Enter details of your mail server. This will allow JIRA to send notifications when issues change.

8. Start using JIRA

That's it! Your JIRA site is accessible from your base URL or a URL like this: `http://<computer_name_or_IP_address>:<port>`

Here's a few things that will help you get your team up and running:

- [Add and invite users](#) to get your team on board, or [configure user directories](#) for slightly bigger teams.
- [Create your first project](#) to have something to work on.
- [Configure SSL or HTTPS](#) to keep JIRA and your team more secure.

Troubleshooting

▼ [Running into problems installing JIRA?](#)

- Some anti-virus or other Internet security tools may interfere with the JIRA installation process and prevent the process from completing successfully. If you experience or anticipate experiencing such an issue with your anti-virus/Internet security tool, disable this tool first before proceeding with the JIRA installation.
- The [Linux OOM Killer](#) can sometimes kill JIRA processes when memory on the server becomes too low. See [How to Configure the Linux Out-of-Memory Killer](#).

Head to [Installation Troubleshooting](#) in our Knowledge Base for more help.

Uninstalling JIRA applications from Linux

This page describes the procedure for uninstalling JIRA, which had been installed using the [Linux installer](#).

i If you wish to re-install JIRA in 'unattended mode', do not uninstall your previous installation of JIRA just yet. See [Using the silent installation feature](#) for more information.

To uninstall JIRA from Linux:

1. Open a Linux console.
2. Change directory (`cd`) to your JIRA installation directory. For example:

```
cd /opt/atlassian/jira/
```

3. Execute the command `uninstall`

i This command must be executed as the same user account that was used to install JIRA with the [Linux installer](#).

4. Follow the prompts to uninstall JIRA from your computer.

i Please note:

- All files within the JIRA installation directory will be deleted (with the exception of the Tomcat log folder located in the JIRA installation directory).
- The uninstaller will **NOT** delete:
 - The JIRA database
 - The JIRA home directory
 - Log files that were generated while JIRA was running
- The uninstaller can be made to operate in unattended mode by specifying the `-q` option — i.e. `uninstall -q`

Installing JIRA applications on Linux from Archive File

In this guide we'll run you through installing a JIRA application in a production environment, with an external database, manually using a `tar.gz` file.

This method gives you the most control over the installation process.

Other ways to install JIRA:

- [Evaluation](#) - get your free trial up and running in no time.
- [Installer](#) – install JIRA using the Linux installer.
- [Windows](#) – install JIRA on a Windows server.

On this page:

Before you begin

Install a JIRA application

1. Download JIRA
2. Create the installation directory
3. Create the home directory
4. Check the ports
5. Start JIRA

Set up your JIRA application

6. Choose set up method
7. Connect to your database
8. Set application properties
9. Enter your license
10. Create your administrator account
11. Set up email notifications
12. Start using JIRA

Troubleshooting

Before you begin

Before you install JIRA, there are a few questions you need to answer.

<p>Are you using a supported operating system and Java version?</p>	<p>▼ Tell me more...</p> <p>Check the Supported platforms page for the version of JIRA you are installing. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • We don't support installing JIRA on OS X or mac OS for production environments. • JIRA can't run on OpenJDK. You'll need to install Oracle Java. • You can use either the JDK (Java Development Kit) or JRE (Java Runtime Environment). • We only support the version of Apache Tomcat that is bundled with JIRA.
<p>Do you want to run JIRA as a service?</p>	<p>Tell me more...</p> <p>Running JIRA as a service means that your JIRA application will automatically start up when your Linux server is started.</p> <p>You should use the Linux installer if you want to run JIRA as a service.</p> <p>If you choose not to run JIRA as a service:</p> <ul style="list-style-type: none"> • You will start your JIRA application by running the <code>start-jira.sh</code> file in your JIRA installation directory. • We recommend creating a dedicated user to run JIRA. This user must have full read, write and execute access to the installation directory and home directory. • JIRA will need to be restarted manually if your server is restarted.
<p>What database do you plan to use?</p>	<p>Tell me more...</p> <p>To run JIRA in production you'll need an external database. Check the Supported platforms page for the version you're installing for the list of databases we currently support. If you don't already have a database, PostgreSQL is free, easy to set up and has been extensively tested with JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • Set up your database before you begin. Step-by-step guides are available for PostgreSQL, Oracle, MySQL, and SQL Server. • Use UTF-8 character encoding. • If you're using Oracle or MySQL you'll need to download the driver for your database. • The embedded H2 database can be used for evaluating JIRA, but you'll need to migrate to another database before running in production. You may find it easier to use external database from the start.
<p>Do you have a JIRA license?</p>	<p>Tell me more...</p> <p>You'll need a valid JIRA Software Server, JIRA Core Server or JIRA Service Desk Server license to use JIRA.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you have not yet purchased a JIRA application license you'll be able to create an evaluation license during setup. • If you already have a license key you'll be prompted to log in to my.atlassian.com to retrieve it, or you can enter the key manually during setup. • If you're migrating from JIRA Cloud, you'll need a new license.

<p>Is your <code>JAVA_HOME</code> variable set correctly?</p>	<p>▼ Tell me more...</p> <p>Before you install JIRA, check that you're running a supported Java version and that the <code>JAVA_HOME</code> environment variable is set correctly.</p> <p>JIRA applications can only run with Oracle JDK or JRE.</p> <p>To check your Java version:</p> <pre style="border: 1px dashed #ccc; padding: 5px;">\$ java -version</pre> <p>To check your <code>JAVA_HOME</code> variable is set correctly:</p> <pre style="border: 1px dashed #ccc; padding: 5px;">\$ echo \$JAVA_HOME</pre> <p>If you see a path to your Java installation directory, the <code>JAVA_HOME</code> environment variable has been set correctly. If a path is not returned you'll need to set your <code>JAVA_HOME</code> environment variable manually before installing JIRA.</p>
<p>Have you created a dedicated user to run JIRA?</p>	<p>Tell me more...</p> <p>We strongly recommend running JIRA as a dedicated user.</p> <p>You should create this user before you begin, so that when creating the installation and home directories, you can give this user appropriate read and write permissions.</p> <p>In this example, we'll create a user called <code>jira</code>:</p> <pre style="border: 1px dashed #ccc; padding: 5px;">\$ sudo /usr/sbin/useradd --create-home --comment "Account for running JIRA Software" --shell /bin/bash jira</pre>

Install a JIRA application

1. Download JIRA

Download the `tar.gz` file for your operating system:

- JIRA Core at <https://www.atlassian.com/software/jira/core/download>
- JIRA Software at <https://www.atlassian.com/software/jira/download>
- JIRA Service Desk at <https://www.atlassian.com/software/jira/service-desk/download>

2. Create the installation directory

1. Create your installation directory – this is where JIRA will be installed. Avoid using spaces or special characters in the path. We'll refer to this directory as your `<installation-directory>`.

▼ [Show me how to do this...](#)

In this example we'll call our installation directory `jirasoftware`:

```
$ mkdir jirasoftware
```

2. Extract the JIRA `tar.gz` file to your `<installation-directory>`. We recommend using a GNU `v`

ersion of the archive utility, especially on Solaris.

▼ [Show me how to do this...](#)

Change to the directory where you downloaded JIRA then execute these commands:

```
$ tar -xzf atlassian-jira-software-X.X.X.tar.gz -C
<installation-directory>
$ cd <installation-directory>
$ tar -xf atlassian-jira-software-X.X.X.tar
```

Replace `x.x.x` with your JIRA version and `<installation-directory>` with the full path to the directory you created in the last step.

3. Give your dedicated JIRA user read, write and execute permission to your `<installation-directory>`.

▼ [Show me how to do this...](#)

In this example we're changing ownership of the installation directory and giving the user `jira` read, write and execute permissions.

```
$ chown -R jira <installation-directory>
$ chmod -R u=rwx,go-rwx <installation-directory>
```

3. Create the home directory

1. Create your home directory – this is where JIRA application data like logs, search indexes and files will be stored. This should be separate to your installation directory, with no spaces or special characters in the path. Each JIRA application needs its own home directory.

We'll refer to this directory as your `<home-directory>`.

▼ [Show me how to do this...](#)

In this example we'll call our home directory `jirasoftware-home`:

```
$ mkdir jirasoftware-home
```

2. Give your dedicated JIRA user read, write and execute permissions to the `<home-directory>`.

▼ [Show me how to do this...](#)

In this example we're changing ownership of the home directory and giving the user `jira` read, write and execute permissions.

```
$ chown -R jira <home-directory>
$ chmod -R u=rwx,go-rwx <home-directory>
```

3. Tell JIRA where to find your `<home-directory>` when it starts up. There are two ways to do this:

▼ [Edit the jira-application.properties file...](#)

Edit `<installation-directory>\atlassian-jira\WEB-INF\classes\jira-application.properties` file in any text editor.

After `jira.home` add the absolute path to your home directory (not a symlink), for example:

```
jira.home=/var/jirasoftware-home
```

▼ Set an environment variable...

You can set an environment variable named `JIRA_HOME` in your operating system with the absolute path to your <home-directory>.

In Terminal, execute the following:

```
export JIRA_HOME=/path/to/home-directory
```

You can then specify the command above in a script used to start JIRA.

4. Check the ports

By default JIRA listens on port 8080. If you have another application running on your server that uses the same ports, you'll need to tell JIRA to use a different port.

▼ Show me how to do this...

To change the ports:

1. Edit <installation-directory>\conf\server.xml
2. Change the **Server** port (8005) and the **Connector** port (8080) to free ports on your server.

In the example below we've changed the **Server** port to 5005 and the **Connector** port to 5050.

```
<Server port="5005" shutdown="SHUTDOWN">
...
  <Service name="Catalina">
    <Connector port="5050"
      maxThreads="150"
      minSpareThreads="25"
      connectionTimeout="20000"
      enableLookups="false"
      maxHttpHeaderSize="8192"
      protocol="HTTP/1.1"
      useBodyEncodingForURI="true"
      redirectPort="8443"
      acceptCount="100"
      disableUploadTimeout="true"/>
```

If you are running on a Unix server and bind the ports below 1024 (such as port 80 for example), you will **need to start JIRA as root** in order to successfully bind to the port.

5. Start JIRA

1. Run <installation-directory>/bin/start-jira.sh to start the setup process.

▼ Show me how to do this...

We recommend running JIRA as your dedicated user.

```
$ su -u <user>
$ ./start-jira.sh
```

If you're using Ubuntu the command is a little different:

```
$ sudo su <user>
$ ./start-jira.sh
```

2. Go to <http://localhost:8080/> to launch JIRA in your browser (change the port if you've updated the Connector port).

▼ [Trouble starting JIRA?](#)

- Check your JAVA_HOME variable is set correctly.

Set up your JIRA application

6. Choose set up method

Choose **I'll set it up myself**.

7. Connect to your database

1. If you've not already done so, it's time to create your database. See the 'Before you begin' section of this page for details.
2. Choose **My own database**.
3. Choose your database type then enter the details for your database.

▼ [Show me how to do this...](#)

JIRA connects to your database using a standard JDBC database connection. Connection pooling is handled within JIRA, you can change this using [JIRA configuration tool](#) later.

If you're using Oracle or MySQL there's an extra step:

- Download and extract the appropriate database JDBC drivers. See [Supported platforms](#) to get the right version.
- Drop the JAR file into your `<jira-installation>/lib` folder before continuing with the setup wizard.

In the setup wizard:

- **Driver Class Name** – the Java class name for your database driver. If you're not sure, check the documentation for your database.
- **Database URL** – the JDBC URL for your database. If you're not sure, check the documentation for your database.
- **Username** and **Password** – A valid username and password that JIRA can use to access your database.

8. Set application properties

1. Give your JIRA site a name.
2. Choose whether your site should be private or if anyone can sign up. You can change this later.
3. Enter your base URL - this is the address people will use to access your JIRA site.

9. Enter your license

Follow the prompts to log in to my.atlassian.com to retrieve your license, or enter a license key.

10. Create your administrator account

Enter details for the administrator account. You can add more administrators after set up is complete.

11. Set up email notifications

Enter details of your mail server. This will allow JIRA to send notifications when issues change.

12. Start using JIRA

That's it! Your JIRA site is accessible from your base URL or a URL like this: `http://<computer_name_or_IP_address>:<port>`

Here's a few things that will help you get your team up and running:

- [Add and invite users](#) to get your team on board, or [configure user directories](#) for slightly bigger teams.
- [Create your first project](#) to have something to work on.
- [Configure SSL or HTTPS](#) to keep JIRA and your team more secure.

Troubleshooting

▼ [Running into problems installing JIRA?](#)

- Check your `JAVA_HOME` is set correctly.
- Use a [GNU](#) version of the `unzip` utility. There are known issues extracting the `tar.gz` file on Solaris and AIX. See '[extractBundledPlugins Couldn't find atlassian-bundled-plugins.zip on classpath](#)' Due to Solaris TAR Utility.

Head to [Installation Troubleshooting](#) in our Knowledge Base for more help.

Unattended installation

If you've previously installed JIRA using the Windows or Linux installer, you can use a configuration file from your existing JIRA installation (`response.varfile`) to re-install JIRA in unattended mode, no user input required.

This can be useful when you have installed JIRA on a test server and are ready to install on your production server with the same configuration.

Good to know

- The `response.varfile` file contains the options specified during the installation wizard steps of your previous JIRA installation. Don't uninstall your previous JIRA installation until after you've copied this file to your new install location.
- If you decide to modify the `response.varfile` file, make sure all directory paths specified are absolute, for example, `sys.installationDir=C:\\Program Files\\Atlassian\\jira` (Windows) or `sys.installationDir=/opt/atlassian/jira` (Linux).

Unattended installations will fail the file contains relative directory paths.

Install a JIRA application in unattended mode

1. Download the installer for your operating system:
 - JIRA Core at <https://www.atlassian.com/software/jira/core/download>
 - JIRA Software at <https://www.atlassian.com/software/jira/download>
 - JIRA Service Desk at <https://www.atlassian.com/software/jira/service-desk/download>
2. Copy `<installation-directory>/install4j/response.varfile` from your existing JIRA installation to where you downloaded the installer.
3. In command prompt or terminal change directory (`cd`) to where you downloaded the installer.
4. Run the following command to install JIRA:

Windows

```

JIRA Core
> atlassian-jira-core-X.X.X-x64.exe -q -varfile response.varfile

```

JIRA Software

```
> atlassian-jira-software-X.X.X-x64.exe -q -varfile
response.varfile
```

JIRA Service Desk

```
> atlassian-servicedesk-X.X.X-x64.exe -q -varfile
response.varfile
```

Linux**JIRA Core**

```
$ atlassian-jira-core-X.X.X-x64.bin -q -varfile
response.varfile
```

JIRA Software

```
$ atlassian-jira-software-X.X.X-x64.bin -q -varfile
response.varfile
```

JIRA Service Desk

```
$ atlassian-servicedesk-X.X.X-x64.bin -q -varfile
response.varfile
```

Where `-x.x.x` is the JIRA version you downloaded.

`-q` instructs the installer to run in unattended mode (quietly). `-varfile` specifies the location and name of the configuration file containing the options used by the installer.

5. JIRA will start automatically once the silent installation finishes.

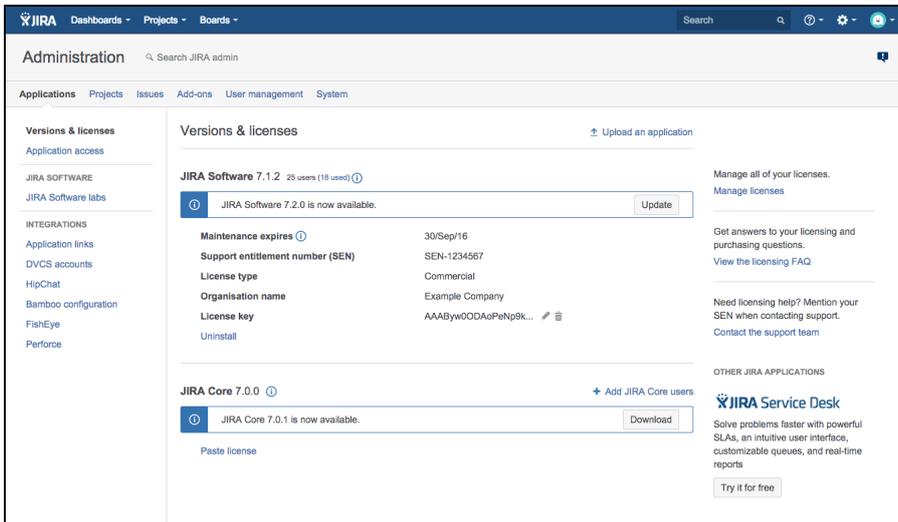
Finally, head to <http://localhost:<port>> to finish setting up JIRA.

See the **Set up a JIRA application** section on [Installing JIRA applications on Windows](#) or [Installing JIRA applications on Linux](#) for more info.

Installing additional applications and version updates

After you have [installed your first JIRA application](#) and have it running, you can install additional applications and update existing applications through the **Versions & licenses** page.

The image below shows you a typical installation of JIRA Software. Note that only JIRA Software is licensed. All updates for installed applications, in this instance JIRA Core and JIRA Software, are displayed. JIRA Service Desk is not installed, and you can see it's available to try on the right hand side.



On this page:

- Before you begin
- Discovering and installing additional applications
- Updating installed applications to the latest available update
- Updating installed applications to different version
- Updating JIRA Core
- Options when you have no internet connection

Before you begin

- You need to have the **JIRA Administrator** global permission to install additional applications or updates.
- To get a trial license for an application, you need your [Atlassian account](#) login details.
- If your JIRA instance is not connected to the internet, you can download the additional application or required updates manually from the [Atlassian website](#). You can then get a trial license for your additional application at [my.atlassian.com](#). You may be asked to provide your [Server ID](#).

Discovering and installing additional applications

If your JIRA instance is connected to the internet, JIRA will list additional applications available on the **Versions & licenses** page. You can download, install and license these applications directly on this page.

1. Choose



> **Applications**. The list of additional application will display on the right-hand side of your **Versions & licensing** page.

2. Select the application you'd like to install, click on **Try it for free** and follow the prompts.
3. Your application is installed with a trial license and you're ready to go! For an overview of what additional functionality and features you'll be using, you can review the [applications and project types overview](#) page.

Updating installed applications to the latest available update

1. Choose



> **Applications**. Any applications that have updates available will display a message informing you of

- the latest available update.
- 2. Click the **Download** button in the message. A progress bar for your download will display, and confirm when it's completed.
- 3. Your application is up to date!

JIRA Software and JIRA Service Desk can be updated in this way, while your server is running. JIRA Core updates work differently and are described [below](#).

Updating installed applications to different version

Sometimes you may need to update an application to a version that is not the latest available. This could be due to compatibility requirements with your JIRA Core version, or due to your license restricting you to updates prior to maintenance expiring.

When you want to update an installed application to a version that is not the latest available, you first need to download the version update file. You can browse available versions, along with their compatibility, on the Atlassian website:

- [JIRA Software available versions](#)
- [JIRA Service Desk available versions](#)

Make sure the version you download is compatible with your JIRA Core version. Once you've downloaded the update file, you can manually install it:

1. Choose  **> Applications.**
2. Select the **Upload an application** link.
3. Browse to the update file you downloaded.
4. Click the **Upload** button. A progress bar for your upload will display, and confirm when it's been uploaded and installed.
5. Your application is now updated to the version you selected.

Updating JIRA Core

The **Versions & licenses** page will notify you when an updated version of JIRA Core is available. However, unlike version updates for JIRA Software and JIRA Service Desk, updates for JIRA Core cannot be applied while your server is running. Instead, **Versions & licenses** page will prompt you to download the installer for the new version.

If you want to download something other than the latest installer for JIRA Core, you can download it from the Atlassian website:

- [JIRA Core available versions](#)

To update your JIRA Core installation, you should follow our [upgrade documentation](#), as this is an important step that requires planning and preparation.

Options when you have no internet connection

If your JIRA server is not connected directly to the Internet, or your firewall blocks connections to the [Atlassian Marketplace website](#), the Versions & licenses page will not be able to check for or apply version updates.

There are several scenarios that you may need to cover:

- To update your JIRA Core installation, you should follow our [upgrade documentation](#), as this is an important step that requires planning and preparation.
- To update any other existing JIRA applications, you can follow the steps set out in [Updating installed applications to a different version](#), making sure that the version you download is compatible with your JIRA Core version.
- To install new applications, download the application file as described in [Updating installed applications to a different version](#). For this option, you'll also need to obtain a trial license for your additional application at [my.atlassian.com](#) so that you can update the license key manually after

you've installed the application.

Connecting JIRA applications to a database

JIRA requires a relational database to store its issue data.

If you are setting up a completely new JIRA installation, the [JIRA setup wizard](#) will configure a database connection for you to either JIRA's internal [H2](#) or an external database.

i JIRA's internal H2 database is suitable for evaluation purposes. For production installations of JIRA, we strongly recommend that you connect JIRA to another [supported database](#). This allows you to take advantage of your database system's own backup and recovery features.

The following are more detailed instructions for configuring a connection to a JIRA database:

- [Connecting JIRA applications to PostgreSQL](#)
- [Connecting JIRA applications to MySQL](#)
- [Connecting JIRA applications to Oracle](#)
- [Connecting JIRA applications to SQL Server 2012](#)
- [Connecting JIRA applications to SQL Server 2014](#)
- [Tuning database connections](#)

Which database?

Your choice of database can significantly affect your subsequent experience of JIRA administration. If you have a choice of databases, please first read [our list of supported databases](#).

If you are looking for a low-cost solution, consider using [MySQL](#) or [PostgreSQL](#), as both of these are open source (free) software.

Upgrading JIRA or migrating JIRA to another server?

If you are [upgrading JIRA manually](#) or [migrating JIRA to another server](#), and do not have access to a pre-existing `dbconfig.xml` file, you will need to re-configure your database connection. This results in a `dbconfig.xml` file (being created in the [JIRA home directory](#) of your new JIRA installation), whose content defines your JIRA database connection.

You can re-configure your database connection with either the [JIRA configuration tool](#), or you can do it manually.

✓ Specific instructions for configuring database connections, either using the JIRA configuration tool or manually, are provided in the specific instructions for each database (listed above).

Data Migration

To transfer your issue data from one database to another, please refer to the instructions for [Switching databases](#).

Connecting JIRA applications to PostgreSQL

These instructions will help you connect JIRA to a [PostgreSQL](#) database.

Before you begin

- Check whether your version of PostgreSQL is supported. See [Supported platforms](#).
- If you are [migrating JIRA to another server](#), create an export of your data as an [XML backup](#). You will then be able to transfer data from your old database to your new database, as described in [Switching databases](#).
- Shut down JIRA before you begin, unless you are running the setup wizard.

On this page:

- Before you begin
- 1. Create and configure the PostgreSQL database
- 2. Configure your JIRA server to connect to your PostgreSQL database
- 3. Start JIRA

1. Create and configure the PostgreSQL database

Accept remote TCP connections (remote PostgreSQL server only)

If you are connecting JIRA to a remote PostgreSQL server (i.e. if your PostgreSQL server is not installed locally on your JIRA server host system), you will need to configure your `data/postgresql.conf` and `data/pg_hba.conf` files to accept remote TCP connections from your JIRA server's IP address.

The following PostgreSQL documentation contains information on the appropriate `listen_addresses` value in the `postgresql.conf` file as well as the `pg_hba.conf` file:

- [PostgreSQL 9.6 documentation — Connections and Authentication](#)
- [PostgreSQL 9.5 documentation — Connections and Authentication](#)
- [PostgreSQL 9.4 documentation — Connections and Authentication](#)
- [PostgreSQL 9.3 documentation — Connections and Authentication](#)

After you modify the `data/postgresql.conf` and `data/pg_hba.conf` files, restart PostgreSQL for the changes to take effect.

Create users and databases for your version of PostgreSQL

You can find information on creating users and databases for your version of PostgreSQL on their [website](#).

1. Create a database user (login role) which JIRA will connect as (e.g. `jiradbuser`).

Remember this database user name, as it will be used to configure JIRA's connection to this database in subsequent steps.

2. Create a database for JIRA to store issues in (e.g. `jiradb`) with Unicode collation.

Remember this database name, as it will be used to configure JIRA's connection to this database in subsequent steps.

```
CREATE DATABASE jiradb WITH ENCODING 'UNICODE' LC_COLLATE 'C'
LC_CTYPE 'C' TEMPLATE template0;
```

Or from the command-line:

```
$ createdb -E UNICODE -l C -T template0 jiradb
```

3. Ensure that the user has permissions to connect to the database, and to create and write to tables in the database.

```
GRANT ALL PRIVILEGES ON DATABASE <Database Name> TO <Role Name>
```

- To verify that the privileges were granted successfully, connect to the database and run the `\z` command.

2. Configure your JIRA server to connect to your PostgreSQL database

There are two ways to configure your JIRA server to connect to your PostgreSQL database:

- **Using the JIRA setup wizard** — Use this method if you have just installed JIRA, and you are setting it up for the first time. Your settings will be saved to the `dbconfig.xml` file in your [JIRA home directory](#).
- **Using the JIRA configuration tool** — Use this method if you have an existing JIRA instance. Your settings will be saved to the `dbconfig.xml` file in your [JIRA home directory](#).

Instructions for each configuration method

JIRA setup wizard

The [JIRA setup wizard](#) will display when you access JIRA for the first time in your browser.

- In the first screen, 'Configure Language and Database', set **Database Connection to My own database**.
- Set **Database Type to PostgreSQL**.
- Fill out the fields, as described in the [Database connection fields](#) section below.
- Test your connection and save.

JIRA configuration tool

- Run the JIRA configuration tool as follows:
 - **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
 - **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).  This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set](#) error KB article. Please refer to it for the workaround.
- Navigate to the **Database** tab, and set **Database type to PostgreSQL**.
- Fill out the fields, as described in the [Database connection fields](#) section below.
- Test your connection and save.
- Restart JIRA.

Database connection fields

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:postgresql://dbserver:5432/jiradb</url></code>	The name or IP address of the machine that the PostgreSQL server is installed on.

Port	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:postgresql://dbserver:5432/jiradb</url></code>	The TCP/IP port that the PostgreSQL server is listening on. You can leave this blank to use the default port.
Database	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:postgresql://dbserver:5432/jiradb</url></code>	The name of your PostgreSQL database (into which JIRA will save its data). You should have created this in Step 1 above.
Username	Located in the <code><username></code> tag (see bold text in example below): <code><username>jiradbuser</username></code>	The user that JIRA uses to connect to the PostgreSQL server. You should have created this in Step 1 above.
Password	Located in the <code><password></code> tag (see bold text in example below): <code><password>jiradbuser</password></code>	The user's password — used to authenticate with the PostgreSQL server.
Schema	Located in the <code><schema-name></code> tag (see bold text in example below): <code><schema-name>public</schema-name></code>	The name of the schema that your PostgreSQL database uses. PostgreSQL 7.2 and later require a schema to be specified in the <code><schema-name/></code> element. If your PostgreSQL database uses the default 'public' schema, this should be specified in the <code><schema-name/></code> element as shown below. Ensure that your database schema name is lower-case, as JIRA cannot work with PostgreSQL databases whose schema names contain upper-case characters.

Sample dbconfig.xml file

For more information about the child elements of `<jdbc-datasource/>` beginning with `pool` in the `dbconfig.xml` file above, see [Tuning database connections](#).

```
<?xml version="1.0" encoding="UTF-8"?>

<jira-database-config>
  <name>defaultDS</name>
  <delegator-name>default</delegator-name>
  <database-type>postgres72</database-type>
  <schema-name>public</schema-name>
  <jdbc-datasource>
    <url>jdbc:postgresql://dbserver:5432/jiradb</url>
    <driver-class>org.postgresql.Driver</driver-class>
    <username>jiradbuser</username>
    <password>password</password>
    <pool-min-size>20</pool-min-size>
    <pool-max-size>20</pool-max-size>
    <pool-max-wait>30000</pool-max-wait>
    <pool-max-idle>20</pool-max-idle>
    <pool-remove-abandoned>true</pool-remove-abandoned>

    <pool-remove-abandoned-timeout>300</pool-remove-abandoned-timeout>

    <validation-query>select version();</validation-query>

    <min-evictable-idle-time-millis>60000</min-evictable-idle-time-millis>
  >

  <time-between-eviction-runs-millis>300000</time-between-eviction-runs-
  -millis>

    <pool-test-on-borrow>false</pool-test-on-borrow>
    <pool-test-while-idle>true</pool-test-while-idle>

  </jdbc-datasource>
</jira-database-config>
```

3. Start JIRA

You should now have JIRA configured to connect to your PostgreSQL database. The next step is to start it up!

 **Congratulations, you now have JIRA connected to your PostgreSQL database.**

Connecting JIRA applications to MySQL

These instructions will help you connect JIRA to a supported [MySQL](#) database.

Before you begin

- Check whether your version of MySQL is supported. See [Supported platforms](#).
- Check [known issues](#) below.
- If you are [migrating JIRA to another server](#), create an export of your data as an [XML backup](#). You will then be able to transfer data from your old database to your new database, as described in [Switching databases](#).

- If you plan to set up Confluence and JIRA on the same MySQL server, read the [Confluence MySQL setup guide](#). Confluence requirements are more strict than JIRA's, so you should configure MySQL to suit Confluence. This configuration will work for JIRA, too.
- Shut down JIRA before you begin, unless you are running the setup wizard.

On this page:

- Before you begin
- 1. Create and configure the MySQL database
- 2. Copy the MySQL JDBC driver to your application server
- 3. Configure your JIRA server to connect to your MySQL database
- 4. Start JIRA
- Known issues

1. Create and configure the MySQL database

1. Create a database user which JIRA will connect as (e.g. **jiradbuser**).
Remember this database user name, as it will be used to configure JIRA's connection to this database in subsequent steps.
2. Create a database for JIRA to store issues in (e.g. **jiradb**). The database must have a character set of UTF8. Enter the following command from within the MySQL command client.
Remember this database name, as it will be used to configure JIRA's connection to this database in subsequent steps.

```
CREATE DATABASE jiradb CHARACTER SET utf8 COLLATE utf8_bin;
```

(if you want your database to be named **jiradb**).

3. Ensure that the user has permission to connect to the database, and permission to create and populate tables. These can be provided with the following -
For MySQL 5.5, MySQL 5.6, and MySQL 5.7.0 to MySQL 5.7.5:

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER, INDEX on
<JIRADB>.* TO '<USERNAME>'@'<JIRA_SERVER_HOSTNAME>' IDENTIFIED
BY '<PASSWORD>';
flush privileges;
```

For MySQL 5.7.6 and above, you must also include the REFERENCES permission:

```
GRANT
SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, ALTER, INDEX
on <JIRADB>.* TO '<USERNAME>'@'<JIRA_SERVER_HOSTNAME>'
IDENTIFIED BY '<PASSWORD>';
flush privileges;
```

Tip:

To confirm if the permissions were granted successfully, log into the DB server with the JIRA DB user and run the command below:

```
SHOW GRANTS FOR <USERNAME>@<JIRA_SERVER_HOSTNAME>;
```

4. Edit the `my.cnf` file (`my.ini` on Windows operating systems) in your MySQL server. (Refer to [MySQL L Option Files](#) for detailed instructions on editing `my.cnf` and `my.ini`.)

Locate the `[mysqld]` section in the file, and add or modify the following parameters:

- Set the default storage engine to InnoDB:

```
[mysqld]
...
default-storage-engine=INNODB
...
```

- Specify the value of `max_allowed_packet` to be at least 256M:

```
[mysqld]
...
max_allowed_packet=256M
...
```

- Specify the value of `innodb_log_file_size` to be at least 256M for MySQL 5.5 and below:

```
[mysqld]
...
innodb_log_file_size=256M
...
```

NB: This should be set to at least 2G for MySQL 5.6 and above.

- Ensure the `sql_mode` parameter does not specify `NO_AUTO_VALUE_ON_ZERO`

```
// remove this if it exists
sql_mode = NO_AUTO_VALUE_ON_ZERO
```

5. Restart your MySQL server for the changes to take effect:

- On Windows, use the Windows Services manager to restart the service.
- On Linux:
 - Run one of the following commands, depending on your setup: `"/etc/init.d/mysqld stop"` or `"/etc/init.d/mysql stop"` or `"service mysqld stop"`.

- Then run the same command again, replacing 'stop' with 'start'.

2. Copy the MySQL JDBC driver to your application server

If you are **upgrading JIRA and you are using the recommended MySQL driver** (Connector/J JDBC driver v5.1), you can skip the instructions in this section. The JIRA upgrade task will automatically copy over your existing driver to the upgraded installation.

To copy the MySQL JDBC driver to your application server:

1. Get the MySQL driver:
 - If you are **installing JIRA**, download the recommended MySQL driver [JDBC Connector/J 5.1](#). You can download either the `.tar.gz` or the `.zip` file by selecting the *'Platform Independent'* option. Extract the jar for the driver (e.g. `mysql-connector-java-5.x.x-bin.jar`) from the archive.
 - If you are **upgrading JIRA and you are not using the recommended MySQL driver** (JDBC Connector/J 5.1), back up the driver from your JIRA installation before you upgrade. The driver will be in the `<JIRA installation directory>/lib/` directory.
2. Copy the MySQL JDBC driver jar to the `<JIRA installation directory>/lib/` directory for your new/upgraded installation. If you are installing JIRA using the Windows installer, you will need to do this step after running the Windows installer, but **before running the setup wizard**.
3. Restart JIRA / JIRA service.
4. If you are installing JIRA, skip the rest of the instructions on this page and access JIRA in your browser to [run the setup wizard](#) instead.

Please note:

- We recommend the Connector/J driver from MySQL (linked above). A user has reported experiencing problems with the Resin JDBC driver for MySQL.

3. Configure your JIRA server to connect to your MySQL database

There are two ways to configure your JIRA server to connect to your MySQL database:

- **Using the JIRA setup wizard** — Use this method if you have just installed JIRA, and are setting it up for the first time. Your settings will be saved to the `dbconfig.xml` file in your [JIRA home directory](#).
- **Using the JIRA configuration tool** — Use this method, if you have an existing JIRA instance. Your settings will be saved to the `dbconfig.xml` file in your [JIRA home directory](#).

Instructions for each configuration method

JIRA setup wizard

The [JIRA setup wizard](#) will display when you access JIRA for the first time in your browser.

1. In the first screen, 'Configure Language and Database', set **Database Connection to My own database**.
2. Set **Database Type to MySQL**.
3. Fill out the fields, as described in the [Database connection fields](#) section below.
4. Test your connection and save.

JIRA configuration tool

1. Run the JIRA configuration tool as follows:
 - **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
 - **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).
 - This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.
2. Navigate to the **Database** tab and set **Database type to MySQL**.
3. Fill out the fields, as described in the [Database connection fields](#) section below.
4. Test your connection and save.
5. Restart JIRA.

Database connection fields

Setup wizard / Configuration tool	dbconfig.xml
Hostname	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:mysql://dbserver:3306/jiradb?useUnicode=true&characterE:UTF8&sessionVariables=default_storage_engine=InnoDB</url></code>
Port	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:mysql://dbserver:3306/jiradb?useUnicode=true&characterE:UTF8&sessionVariables=default_storage_engine=InnoDB</url></code>
Database	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:mysql://dbserver:3306/jiradb?useUnicode=true&characterE:UTF8&sessionVariables=default_storage_engine=InnoDB</url></code>
Username	Located in the <code><username></code> tag (see bold text in example below): <code><username>jiradbuser</username></code>
Password	Located in the <code><password></code> tag (see bold text in example below): <code><password>jiradbuser</password></code>

Sample dbconfig.xml file

- For more information about the child elements of `<jdbc-datasource/>` beginning with `pool` in the

dbconfig.xml file above, see [Tuning database connections](#).

- Both the JIRA setup wizard and database configuration tool also add the element `<validation-query>select 1</validation-query>` to this file, which is usually required when running JIRA with default MySQL installations. See [Surviving connection closures](#) for more information.
- The database URL in the example below assumes a UTF-8 database — i.e. that your database was created using a command similar to `create database jiradb character set utf8`; If you do not specify `character set utf8` when creating this database, you risk getting 'Data truncation: Data too long for column' errors when importing data or corruption of non-supported characters.
- The database URL in the example below contains the `sessionVariables=default_storage_engine=InnoDB` parameter. We strongly recommend adding this parameter to avoid data corruption.

```
<?xml version="1.0" encoding="UTF-8"?>

<jira-database-config>
  <name>defaultDS</name>
  <delegator-name>default</delegator-name>
  <database-type>mysql</database-type>
  <jdbc-datasource>

  <url>jdbc:mysql://dbserver:3306/jiradb?useUnicode=true&characterE
ncoding=UTF8&sessionVariables=default_storage_engine=InnoDB</url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <username>jiradbuser</username>
  <password>password</password>
  <pool-min-size>20</pool-min-size>
  <pool-max-size>20</pool-max-size>
  <pool-max-wait>30000</pool-max-wait>
  <pool-max-idle>20</pool-max-idle>
  <pool-remove-abandoned>true</pool-remove-abandoned>

  <pool-remove-abandoned-timeout>300</pool-remove-abandoned-timeout>

  <validation-query>select 1</validation-query>

  <min-evictable-idle-time-millis>60000</min-evictable-idle-time-millis
>

  <time-between-eviction-runs-millis>300000</time-between-eviction-runs
-millis>

  <pool-test-while-idle>true</pool-test-while-idle>
  <pool-test-on-borrow>false</pool-test-on-borrow>
  <validation-query-timeout>3</validation-query-timeout>
  </jdbc-datasource>
</jira-database-config>
```

4. Start JIRA

You should now have JIRA configured to connect to your MySQL database. The next step is to start it up!

 **Congratulations, you now have JIRA connected to your MySQL database.**

Known issues

Here's a list of known issues for this database. Expand each of them for more details.

▼ Hostnames in permissions are compared as strings...

If you grant permissions in MySQL to a hostname such as `localhost`, then you'll need to use the same string when connecting to the database from JIRA. Using `127.0.0.1` won't work, even though it resolves to the same place. This mistake will result in warnings about tables not being found, because the JDBC connection didn't have permissions to create the new tables when JIRA was set up.

▼ Connection problems...

If you are using a MySQL database with any of the following, you may experience problems with your connections dropping out (see [JIRA-15731](#) for details):

- JIRA 3.13 or later,
- version 5.5.25 or higher of Tomcat 5,
- version 6.0.13 or higher of Tomcat 6,

For more info on how to address this, see [Surviving connection closures](#).

▼ Special characters for database password are not supported...

Special characters for database password are not supported, because JIRA can't interpret them.

▼ Using the InnoDB storage engine...

The default storage engine used by MySQL Server versions prior to 5.5 is MyISAM. Because of that, a JIRA database running on a default configuration of a MySQL Server earlier than version 5.5 could experience problems with creating tables ([JIRA-24124](#)), which may result in data corruption in JIRA.

To avoid this problem, we strongly recommend specifying the `sessionVariables=default_storage_engine=InnoDB` parameter in your database URL (as stated [above](#)). This will ensure that tables written to JIRA's MySQL database use the InnoDB storage engine, which supports 'database transactions' required by JIRA.

▼ Binary logging...

JIRA uses the READ-COMMITTED transaction isolation level with MySQL, which currently supports only row-based binary logging.

If you require MySQL's binary logging features, you must configure MySQL's binary logging format to be 'row-based'. Otherwise, you may encounter problems when creating issues in JIRA.

▼ 4-byte characters are not supported...

JIRA doesn't support using MySQL with 4-byte characters.

Connecting JIRA applications to Oracle

These instructions will help you connect JIRA to an [Oracle](#) database.

Before you begin

- Check whether your version of Oracle is supported. See [Supported platforms](#).
- Check [known issues](#) below.
- Download the Oracle JDBC 12.2.0.1 driver from the [Oracle website](#). You'll need to later copy it to the JIRA installation directory.
- If you are [migrating JIRA to another server](#), create an export of your data as an [XML backup](#). You will then be able to transfer data from your old database to your new database, as described in [Switching databases](#).
- Shut down JIRA before you begin, unless you are running the setup wizard.

On this page:

- [Before you begin](#)
- [1. Configure Oracle](#)
- [2. Download the Oracle JDBC driver](#)
- [3. Configure your JIRA Server to connect to your Oracle database](#)
- [4. Start JIRA](#)
- [Known issues](#)

1. Configure Oracle

1. Ensure that you have a database instance available for JIRA (either create a new one or use an existing one).
2. Within that database instance, create a user which JIRA will connect as (e.g. `jiradbuser`).
Remember this database user name, as it will be used to configure JIRA's connection to this database in subsequent steps.

```
create user <user> identified by <user_pass> default tablespace
<tablespace_name> quota unlimited on <tablespace_name>;
```

Note:

- When you create a user in Oracle, Oracle will create a 'schema' automatically.
 - When you create a user, the tablespace for the table objects must be specified.
3. Ensure that the user has the following privileges:

```
grant connect to <user>;
grant create table to <user>;
grant create sequence to <user>;
grant create trigger to <user>;
```

It is **very important** that *the user is granted the exact privileges indicated above*. JIRA requires only these privileges — if either less or more than these privileges are applied, some JIRA functions may not work properly.

Simply put, for JIRA functions to work as expected, we advise that you *grant specific privileges* to the user, and *not assign a role* to the user.

For example, if you grant the `RESOURCE` role to a user, and the `RESOURCE` role grants the `SELECT ANY TABLE` privilege, then JIRA functions may not work as expected.

We recommend that you grant the exact privileges indicated above to the user instead.

4. Ensure your database is configured to use the same character encoding as JIRA. The recommended encoding is `AL32UTF8` (the Oracle equivalent of Unicode UTF-8).

2. Download the Oracle JDBC driver

JIRA requires the Oracle JDBC 12.2.0.1 driver. Download it from the Oracle website, and copy to your installation directory.

1. [Download the Oracle JDBC driver](#).
2. Copy the downloaded **ojdbc8.jar** file to the `lib/` directory in the JIRA installation directory.

3. Configure your JIRA Server to connect to your Oracle database

There are two ways to configure your JIRA server to connect to your Oracle database:

- **Using the JIRA setup wizard** — Use this method if you have just installed JIRA, and are setting it up for the first time. Your settings will be saved to the `dbconfig.xml` file in your [JIRA home directory](#).
- **Using the JIRA configuration tool** — Use this method if you have an existing JIRA instance. Your settings will be saved to the `dbconfig.xml` file in your [JIRA home directory](#).

Instructions for each configuration method

JIRA setup wizard

JIRA configuration tool

The [JIRA setup wizard](#) will display when you access JIRA for the first time in your browser.

1. In the first screen, 'Configure Language and Database', set **Database Connection to My own database**.
2. Set **Database Type to Oracle**.
3. Fill out the fields, as described in the [Database connection fields](#) section below.
4. Test your connection and save.

1. Run the JIRA configuration tool as follows:
 - **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
 - **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).
 This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.
2. Navigate to the **Database** tab and set **Database type to Oracle**.
3. Fill out the fields, as described in the [Database connection fields](#) section below.
4. Test your connection and save. Any custom settings specified while manually configuring JIRA with Oracle (e.g., adding the `<connection-properties>SetBigStringTryClob=true</connection-properties>`) will be deleted. You will need to reinstate them manually.
5. Restart JIRA.

Database connection fields

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:oracle:thin:@dbserver:1521/ORCL</url></code>	The name or IP address of the machine that the Oracle server is installed on.
Port	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:oracle:thin:@dbserver:1521/ORCL</url></code>	The TCP/IP port that the Oracle server is listening on. The default port number for Oracle is '1521'.
SID	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:oracle:thin:@dbserver:1521/ORCL</url></code>	The Oracle "System Identifier". The default value for most Oracle servers is 'ORCL'. If you are using the Oracle Express Edition, this will be 'XE'.
Username	Located in the <code><username></code> tag (see bold text in example below): <code><username>jiradbuser</username></code>	The user that JIRA uses to connect to the Oracle server. You should have created this in Step 1 above.
Password	Located in the <code><password></code> tag (see bold text in example below): <code><password>jiradbuser</password></code>	The user's password — used to authenticate with the Oracle server.

Sample dbconfig.xml file

For more information about the child elements of `<jdbc-datasource/>` beginning with `pool` in the `dbconfig.xml` file above, see [Tuning database connections](#).

```

<?xml version="1.0" encoding="UTF-8"?>

<jira-database-config>
  <name>defaultDS</name>
  <delegator-name>default</delegator-name>
  <database-type>oracle10g</database-type>
  <jdbc-datasource>
    <url>jdbc:oracle:thin:@dbserver:1521/ORCL</url>
    <driver-class>oracle.jdbc.OracleDriver</driver-class>
    <username>jiradbuser</username>
    <password>password</password>
    <pool-min-size>20</pool-min-size>
    <pool-max-size>20</pool-max-size>
    <pool-max-wait>30000</pool-max-wait>
    <pool-max-idle>20</pool-max-idle>
    <pool-remove-abandoned>true</pool-remove-abandoned>

    <pool-remove-abandoned-timeout>300</pool-remove-abandoned-timeout>

    <validation-query>select 1 from dual</validation-query>

    <min-evictable-idle-time-millis>60000</min-evictable-idle-time-millis
  >

  <time-between-eviction-runs-millis>300000</time-between-eviction-runs
  -millis>

    <pool-test-while-idle>true</pool-test-while-idle>
    <pool-test-on-borrow>false</pool-test-on-borrow>
  </jdbc-datasource>
</jira-database-config>

```

4. Start JIRA

You should now have JIRA configured to connect to your Oracle database. The next step is to start it up!

 **Congratulations, you now have JIRA connected to your Oracle database.**

Known issues

- If you start experiencing problems when dealing with custom workflows or working with issues that have long descriptions, comments or custom field values, try adding the element `<connection-properties>SetBigStringTryClob=true</connection-properties>` as a child of the `</jdbc-datasource>` element in your `dbconfig.xml` file. This connection property may solve these problems. You'll need to restart JIRA afterwards.

Connecting JIRA applications to SQL Server 2012

These instructions will help you connect JIRA to a [Microsoft SQL Server 2012](#) database.

Before you begin

- If you're [Migrating JIRA applications to another server](#), create an export of your data as an [XML backup](#). You will then be able to transfer data from your old database to your new database, as described in [Switching databases](#).

- Stop JIRA before you begin, unless you just started the installation and are running the Setup Wizard.

1. Create and Configure the SQL Server Database

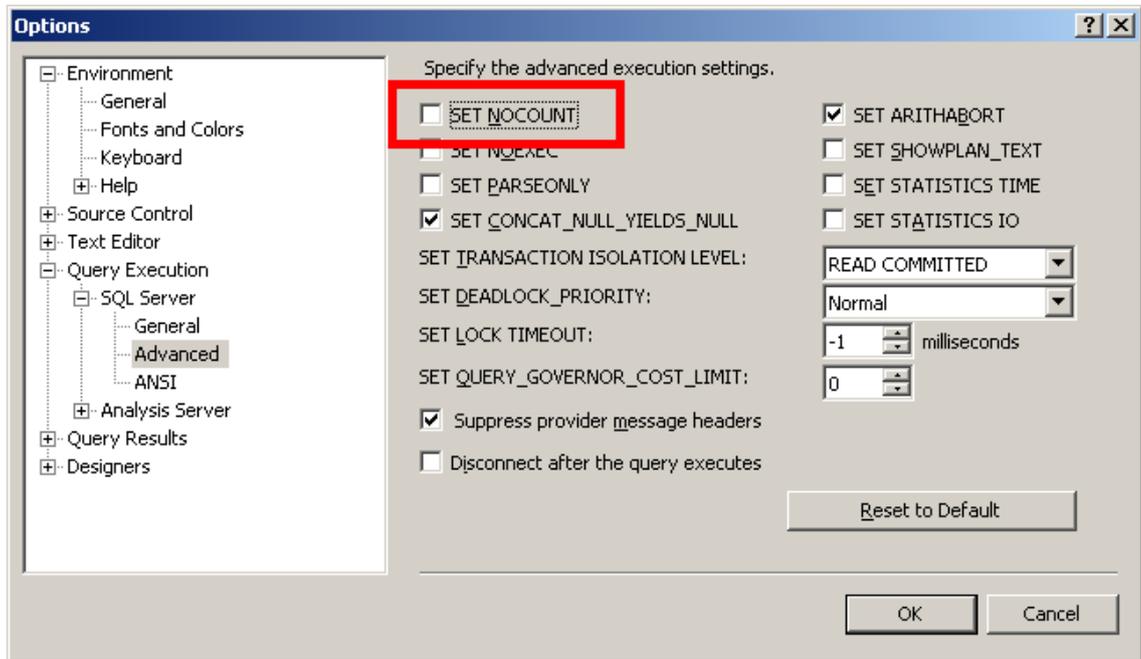
When creating the database, remember your **database name**, **user name**, **schema name**, and **port number**, because you'll need them later to connect JIRA to your database.

1. Create a database for JIRA (e.g. `jiradb`).
 - Make sure the collation type is **case-insensitive**.
 - ▼ [Supported collation types...](#)

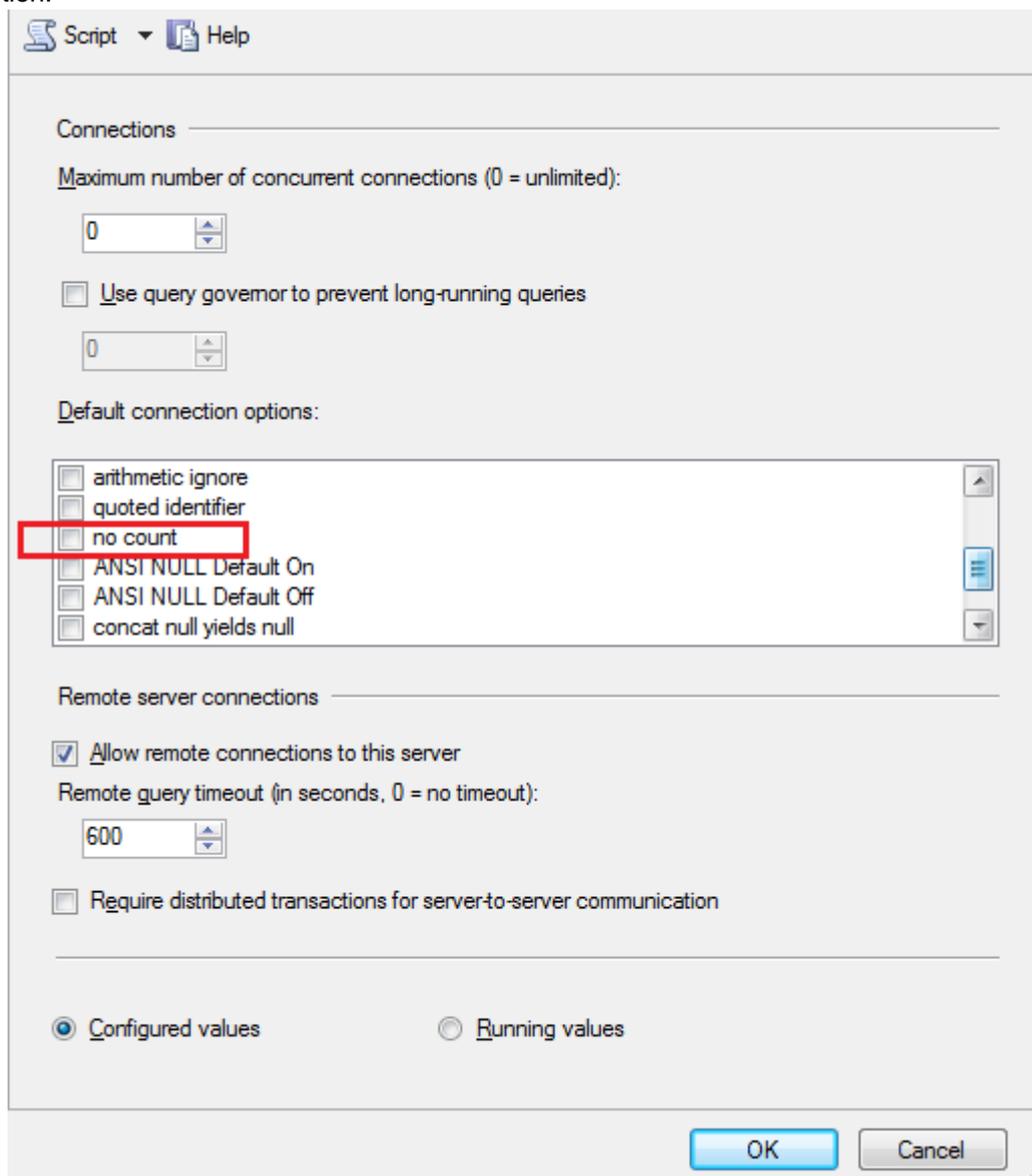
We support `SQL_Latin1_General_CP437_CI_AI` and `Latin1_General_CI_AI` as case-insensitive, accent-insensitive, and language neutral collation types. If your SQL Server installation's collation type settings have not been changed from their defaults, check the collation type settings.
 - SQL Server uses Unicode encoding to store characters. This is sufficient to prevent any possible encoding problems.
2. Create a database user which JIRA will connect as (e.g. `jiradbuser`). This user should *not* be the database owner, but *should* be in the `db_owner` role.
3. Create an empty 'schema' in the database for the JIRA tables (e.g. `jiraschema`).
 - ▼ [Tell me more...](#)

A 'schema' in SQL Server 2012 is a distinct namespace used to contain objects and is *different* from a traditional database schema. You are not required to create any of JIRA's tables, fields or relationships (JIRA will create these objects in your empty schema when it starts for the first time). You can read more on SQL Server 2012 schemas in the relevant [Microsoft documentation](#).
4. Make sure that the database user has permission to connect to the database, and to create and populate tables in the newly-created schema.
5. Make sure that TCP/IP is enabled on SQL Server and is listening on the correct port. A default SQL Server installation uses port number 1433.
6. Make sure that SQL Server is operating in the appropriate authentication mode.
 - ▼ [Tell me more...](#)

By default, SQL Server operates in 'Windows Authentication Mode'. However, if your user is not associated with a trusted SQL connection, i.e. 'Microsoft SQL Server, Error: 18452' is received during JIRA startup, you will need to change the authentication mode to 'Mixed Authentication Mode'. Read the Microsoft documentation on authentication modes and changing the authentication mode to 'Mixed Authentication Mode'.
7. Turn off the SET NOCOUNT option.
 - a. Open SQL Server Management Studio.
 - b. Go to **Tools > Options > Query Execution > SQL Server > Advanced**, and clear the **SET NOCOUNT** check box.



- c. Go to **Server > Properties > Connections > Default Connections**, and clear the **no count** option.



8. Access the **Query Console** by right clicking on the newly created database and selecting 'New Query'. Run the following command to set the isolation level.

```
ALTER DATABASE THE-NEW-DATABASE-CREATED-FOR-JIRA SET
READ_COMMITTED_SNAPSHOT ON
```

2. Configure JIRA to connect to the database

There are two ways to configure your JIRA server to connect to your SQL Server database.

- **Using the JIRA setup wizard** — Use this method, if you have just installed JIRA and are setting it up for the first time. Your settings will be saved to the `dbconfig.xml` file in your [JIRA application home directory](#).
 - ▼ [Show me how to do this...](#)

The JIRA setup wizard will display when you access JIRA for the first time in your browser.

 1. In the first screen, 'Configure Language and Database', set **Database Connection to My own database**.
 2. Set **Database Type to SQL Server**.
 3. Fill out the fields, as described in the [Database connection fields](#) section below.
 4. Test your connection and save.
- **Using the JIRA configuration tool** — Use this method, if you have an existing JIRA instance. Your settings will be saved to the `dbconfig.xml` file in your [JIRA application home directory](#).
 - ▼ [Show me how to do this...](#)
 1. Run the JIRA configuration tool as follows:
 - **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
 - **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).
 - ⓘ This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.
 2. Navigate to the **Database** tab and set **Database type to SQL Server**.
 3. Fill out the fields, as described in the [Database connection fields](#) section below.
 4. Test your connection and save.
 5. Restart JIRA.

Database connection fields

The table shows the fields you'll need to fill out when connecting JIRA to your database. You can also refer to them, and the sample `dbconfig.xml` file below, if you'd like to create or edit the `dbconfig.xml` file manually.

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url></code>	The name or IP address of the machine that the SQL Server server is installed on.
Port	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url></code>	The TCP/IP port that the SQL Server server is listening on. You can leave this blank to use the default port.

Database	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url></code>	The name of your SQL Server database (into which JIRA will save its data). You should have created this in Step 1 above.
Username	Located in the <code><username></code> tag (see bold text in example below): <code><username> jiradbuser </username></code>	The user that JIRA uses to connect to the SQL Server server. You should have created this in Step 1 above.
Password	Located in the <code><password></code> tag (see bold text in example below): <code><password> jiradbuser </password></code>	The user's password — used to authenticate with the SQL Server server.
Schema	Located in the <code><schema-name></code> tag (see bold text in example below): <code><schema-name> dbo </schema-name></code>	The name of the schema that your SQL Server database uses. You should have created this in Step 1 above.

Sample dbconfig.xml file

For more information about the child elements of `<jdbc-datasource/>` beginning with `pool` in the `dbconfig.xml` file above, see [Tuning database connections](#).

```
<jira-database-config>
<name>defaultDS</name>
<delegator-name>default</delegator-name>
<database-type>mssql</database-type>
<schema-name>jiraschema</schema-name>
<jdbc-datasource>
  <url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url>

  <driver-class>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-class>
  <username>jiradbuser</username>
  <password>password</password>
  <pool-min-size>20</pool-min-size>
  <pool-max-size>20</pool-max-size>
  <pool-max-wait>30000</pool-max-wait>
  <pool-max-idle>20</pool-max-idle>
  <pool-remove-abandoned>true</pool-remove-abandoned>
  <pool-remove-abandoned-timeout>300</pool-remove-abandoned-timeout>

  <validation-query>select 1</validation-query>

  <min-evictable-idle-time-millis>60000</min-evictable-idle-time-millis>
  <time-between-eviction-runs-millis>300000</time-between-eviction-runs-millis>

  <pool-test-while-idle>true</pool-test-while-idle>
  <pool-test-on-borrow>>false</pool-test-on-borrow>
</jdbc-datasource>
</jira-database-config>
```

3. Start JIRA

You should now have JIRA configured to connect to your SQL Server database. The next step is to start it up!

Connecting JIRA applications to SQL Server 2014

These instructions will help you connect JIRA to a [Microsoft SQL Server 2014](#) database.

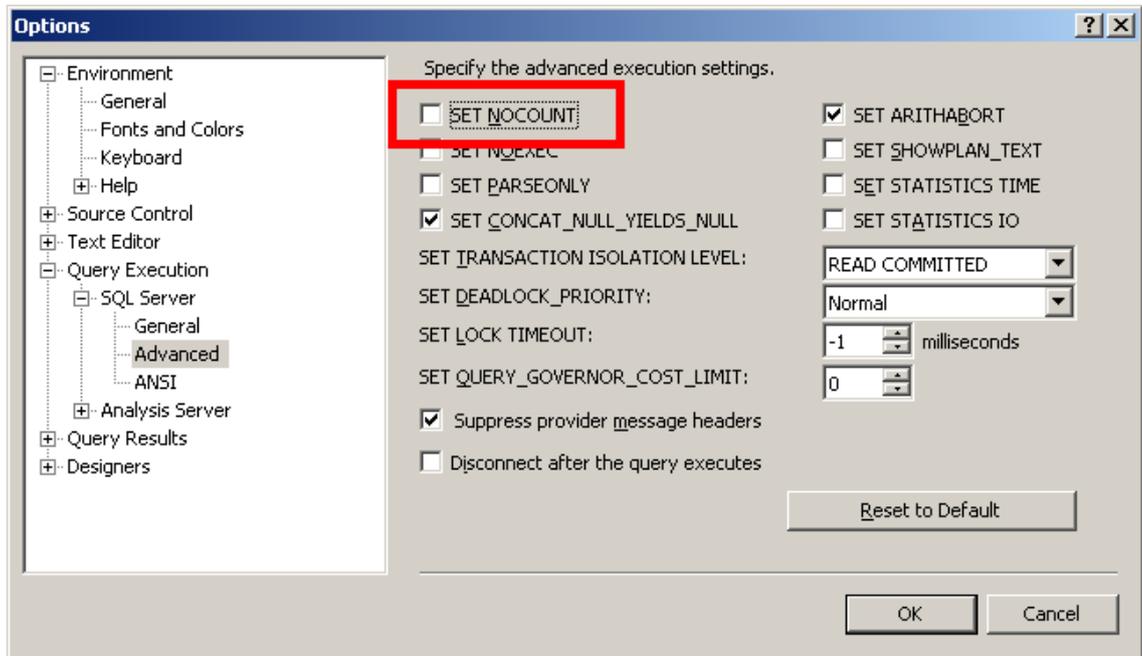
Before you begin

- If you're [Migrating JIRA applications to another server](#), create an export of your data as an [XML backup](#). You will then be able to transfer data from your old database to your new database, as described in [Switching databases](#).
- Stop JIRA before you begin, unless you just started the installation and are running the Setup Wizard.

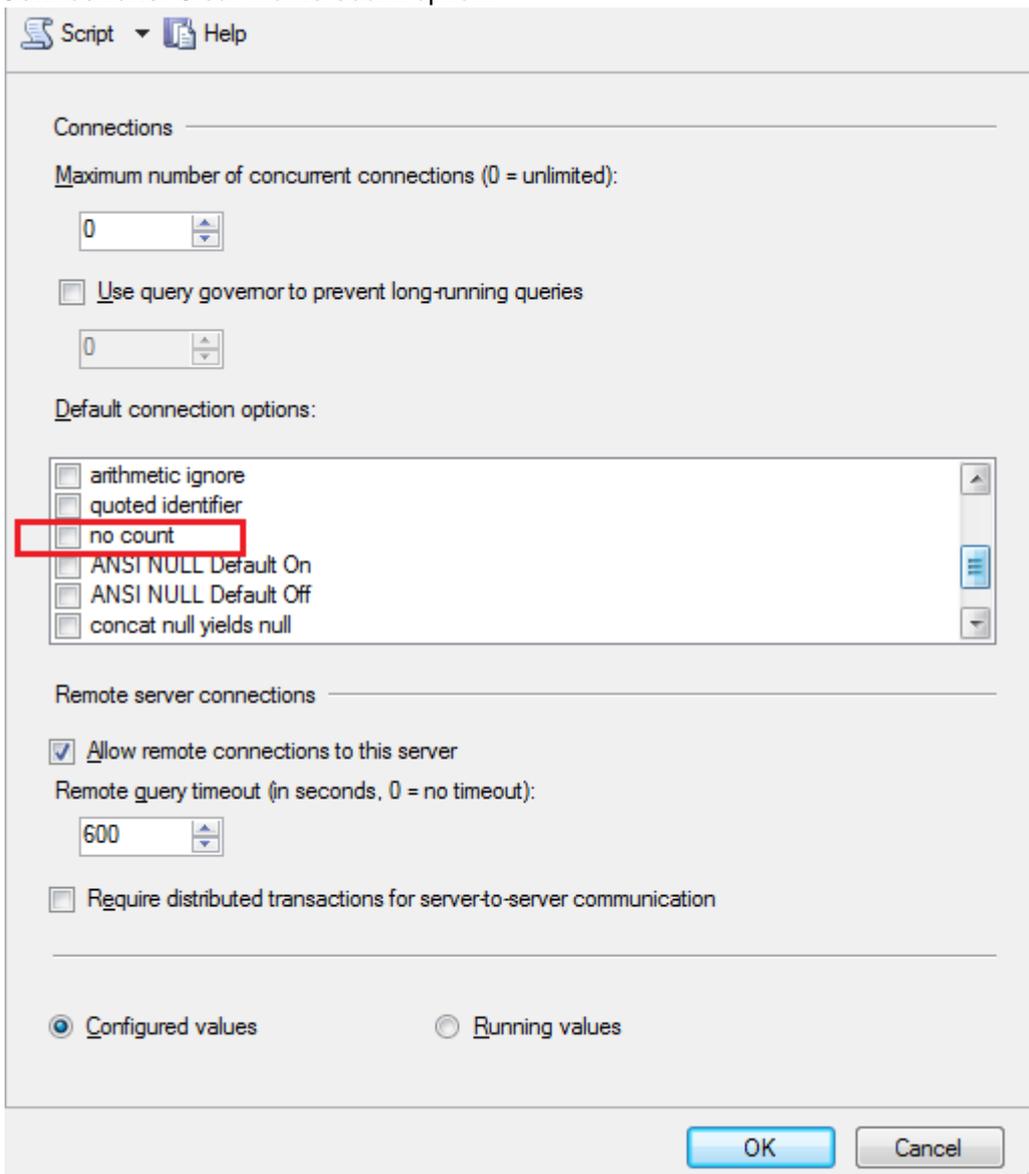
1. Create and configure the SQL Server database

When creating the database, remember your **database name**, **user name**, **schema name**, and **port number**, because you'll need them later to connect JIRA to your database.

1. Create a database for JIRA (e.g. `jiradb`).
 - Make sure the collation type is **case-insensitive**.
 - ▼ [Supported collation types...](#)
We support `SQL_Latin1_General_CP437_CI_AI` and `Latin1_General_CI_AI` as case-insensitive, accent-insensitive, and language neutral collation types. If your SQL Server installation's collation type settings have not been changed from their defaults, check the collation type settings.
 - SQL Server uses Unicode encoding to store characters. This is sufficient to prevent any possible encoding problems.
2. Create a database user which JIRA will connect as (e.g. `jiradbuser`). This user should *not* be the database owner, but *should* be in the `db_owner` role.
3. Create an empty 'schema' in the database for the JIRA tables (e.g. `jiraschema`).
 - ▼ [Tell me more...](#)
A 'schema' in SQL Server 2016 is a distinct namespace used to contain objects and is *different* from a traditional database schema. You are not required to create any of JIRA's tables, fields or relationships (JIRA will create these objects in your empty schema when it starts for the first time). You can read more on SQL Server 2016 schemas in the relevant [Microsoft documentation](#).
4. Make sure that the database user has permission to connect to the database, and to create and populate tables in the newly-created schema.
5. Make sure that TCP/IP is enabled on SQL Server and is listening on the correct port. A default SQL Server installation uses port number 1433.
6. Make sure that SQL Server is operating in the appropriate authentication mode.
 - ▼ [Tell me more...](#)
By default, SQL Server operates in 'Windows Authentication Mode'. However, if your user is not associated with a trusted SQL connection, i.e. 'Microsoft SQL Server, Error: 18452' is received during JIRA startup, you will need to change the authentication mode to 'Mixed Authentication Mode'. Read the Microsoft documentation on authentication modes and changing the authentication mode to 'Mixed Authentication Mode'
7. Turn off the SET NOCOUNT option.
 - a. Open SQL Server Management Studio.
 - b. Go to **Tools > Options > Query Execution > SQL Server > Advanced**, and clear the **SET NOCOUNT** check box.



- c. Right-click your server in the Object Explorer, and go to **Properties > Connections > Default Connections**. Clear the **no count** option.



- Access the **Query Console** by right clicking on the newly created database and selecting 'New Query'. Run the following command to set the isolation level.

```
ALTER DATABASE THE-NEW-DATABASE-CREATED-FOR-JIRA SET
READ_COMMITTED_SNAPSHOT ON
```

2. Configure JIRA to connect to the database

There are two ways to configure your JIRA server to connect to your SQL Server database.

- Using the JIRA setup wizard** — Use this method, if you have just installed JIRA and are setting it up for the first time. Your settings will be saved to the `dbconfig.xml` file in your [JIRA application home directory](#).
 - ▼ [Show me how to do this...](#)

The JIRA setup wizard will display when you access JIRA for the first time in your browser.

 - In the first screen, 'Configure Language and Database', set **Database Connection to My own database**.
 - Set **Database Type to SQL Server**.
 - Fill out the fields, as described in the [Database connection fields](#) section below.
 - Test your connection and save.
- Using the JIRA configuration tool** — Use this method, if you have an existing JIRA instance. Your settings will be saved to the `dbconfig.xml` file in your [JIRA application home directory](#).
 - ▼ [Show me how to do this...](#)
 - Run the JIRA configuration tool as follows:
 - Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
 - Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).
 - i** This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.
 - Navigate to the **Database** tab and set **Database type to SQL Server**.
 - Fill out the fields, as described in the [Database connection fields](#) section below.
 - Test your connection and save.
 - Restart JIRA.

Database connection fields

The table shows the fields you'll need to fill out when connecting JIRA to your database. You can also refer to them, and the sample `dbconfig.xml` file below, if you'd like to create or edit the `dbconfig.xml` file manually.

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url></code>	The name or IP address of the machine that the SQL Server server is installed on.
Port	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url></code>	The TCP/IP port that the SQL Server server is listening on. You can leave this blank to use the default port.

Database	Located in the <code><url></code> tag (bold text in example below): <code><url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url></code>	The name of your SQL Server database (into which JIRA will save its data). You should have created this in Step 1 above.
Username	Located in the <code><username></code> tag (see bold text in example below): <code><username> jiradbuser </username></code>	The user that JIRA uses to connect to the SQL Server server. You should have created this in Step 1 above.
Password	Located in the <code><password></code> tag (see bold text in example below): <code><password> jiradbuser </password></code>	The user's password — used to authenticate with the SQL Server server.
Schema	Located in the <code><schema-name></code> tag (see bold text in example below): <code><schema-name> dbo </schema-name></code>	The name of the schema that your SQL Server database uses. You should have created this in Step 1 above.

Sample dbconfig.xml file

For more information about the child elements of `<jdbc-datasource/>` beginning with `pool` in the `dbconfig.xml` file above, see [Tuning database connections](#).

```
<jira-database-config>
<name>defaultDS</name>
<delegator-name>default</delegator-name>
<database-type>mssql</database-type>
<schema-name>jiraschema</schema-name>
<jdbc-datasource>
  <url>jdbc:sqlserver://dbserver:1433;databaseName=jiradb</url>

  <driver-class>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-class>
  <username>jiradbuser</username>
  <password>password</password>
  <pool-min-size>20</pool-min-size>
  <pool-max-size>20</pool-max-size>
  <pool-max-wait>30000</pool-max-wait>
  <pool-max-idle>20</pool-max-idle>
  <pool-remove-abandoned>>true</pool-remove-abandoned>
  <pool-remove-abandoned-timeout>300</pool-remove-abandoned-timeout>

  <validation-query>select 1</validation-query>

  <min-evictable-idle-time-millis>60000</min-evictable-idle-time-millis>
  <time-between-eviction-runs-millis>300000</time-between-eviction-runs-millis>

  <pool-test-while-idle>true</pool-test-while-idle>
  <pool-test-on-borrow>>false</pool-test-on-borrow>
</jdbc-datasource>
</jira-database-config>
```

3. Start JIRA

You should now have JIRA configured to connect to your SQL Server database. The next step is to start it up!

Tuning database connections

JIRA uses a database connection pool, based on Apache Commons DBCP (DataBase Connection Pool), to manage JIRA's access to its underlying database.

In earlier JIRA versions, the database connection pool was handled purely through the Apache Tomcat application server running JIRA. However, from JIRA version 4.4, JIRA's `dbconfig.xml` file provides a set of database connection pool settings to Tomcat, which in turn are used by Tomcat to manage JIRA's database connection pool. From JIRA version 5.1, the number database connection pool settings defined in JIRA's `dbconfig.xml` file substantially increased.

The information on this page can help you tweak JIRA's database connection pool settings. You can do this by using the [JIRA configuration tool](#) or by directly editing JIRA's `dbconfig.xml` file, as described [below](#).

The **Advanced** tab of the JIRA Configuration Tool makes it easier to both configure and control JIRA's database connection pool. The [Database monitoring](#) page (accessible to JIRA system administrators) provides a visual tool for monitoring JIRA's database connection usage.

On this page:

- [Connection pool architecture](#)
- [Tuning JIRA's database connections](#)
 - [Connection pool settings](#)
 - [Monitoring the connection pool](#)

Connection pool architecture

Whenever JIRA needs to access (i.e. read from or write to) its database, a database connection is required.

A database connection is a large and complex object that handles all communication between JIRA and its database. As such, database connections are time consuming to establish and consume a significant amount of memory on both the client (the JIRA application) and database server.

To avoid the impact of creating a new database connection for each database access request made by JIRA, a pool of pre-established database connections is maintained. Each new database access request made by JIRA uses a connection from this pool of pre-established connections, as required. Hence:

1. When JIRA starts up, a minimum number of database connections are established in the pool between JIRA and its database.
2. When JIRA needs to access its database, JIRA:
 - a. requests a database connection from the pool
 - b. uses this database connection to read from and/or write to its database
 - c. returns the database connection to the pool when finished.

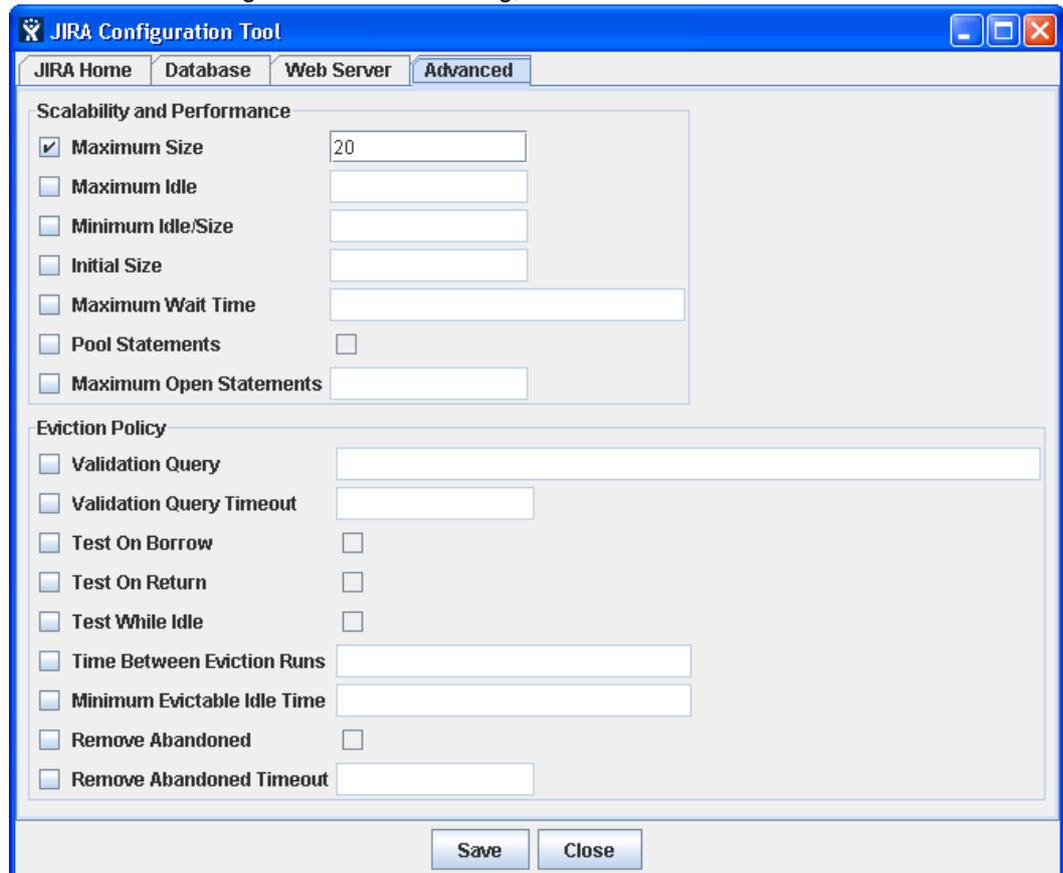
If the frequency of JIRA's database access requests begin to exceed the number of available database connections in the pool, extra connections are automatically created to handle the load.

Conversely, if the frequency of JIRA's database access requests begin to drop below the number of available database connections in the pool, connections can be automatically closed to release resources back to the system.

Modern databases can handle large numbers of connections relatively easily and with sufficient memory, many hundred. On the client side, however, these connections can consume a significant amount memory. Hence, it is generally best to limit the number of connections to a much smaller number while having a sufficient number for the application to rarely need to wait for a connection when it needs one.

Tuning JIRA's database connections

1. Shut down your JIRA installation.
2. Do either of the following:
 - Use the [JIRA configuration tool](#) to tune JIRA's database connections.
 - a. Start the JIRA configuration tool:
 - **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
 - **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).
 - **Information:** This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.
 - **Please Note:** You may need to set the `JAVA_HOME` environment variable to run the JIRA configuration tool. See [Installing Java](#) for details.
 - b. Once the JIRA configuration tool is running, click its **Advanced** tab.



- c. Refer to [Connection pool settings](#) below for more information about the options on this tab. To specify a value for one of these options, ensure its leftmost checkbox has been selected first.
 - **Information:** Some options above are simple checkboxes (i.e. in the centre of the JIRA configuration tool). Selecting these checkboxes sets the values of their associated options to 'true'. Conversely, clearing these checkboxes sets the values of their associated options to 'false'.
 - d. Click the **Save** button to save your changes, which will be stored as elements in your `dbconfig.xml` file.
 - Alternatively, edit the `dbconfig.xml` file at the root of your [JIRA home directory](#).
 - a. Refer to [Connection pool settings](#) below for more information about the elements you can add to your `dbconfig.xml` file to fine tune JIRA's database connection.
 - b. Save your edited `dbconfig.xml` file.
3. Restart your JIRA installation.

Connection pool settings

JIRA configuration tool 'Advanced' tab option	Element in dbconfig.xml	Explanation	Recommendations / Notes	Default
Maximum Size	pool-max-size	The maximum number of database connections that can be opened at any time.	This value should be sufficiently large enough that JIRA rarely needs to wait for a database connection to become available when JIRA requires one. See Monitoring below for suggestions on how to set this parameter.	20
Maximum Idle	pool-max-idle	The maximum number of database connections that are allowed to remain idle in the pool.	Specifying a negative number sets no limit on the number of database connections that can remain idle. If the value of Minimum Idle/Size (below) is the same as that of Maximum Size (above), which is the case by default, then this setting has no effect.	Value of Maximum Size
Minimum Idle/Size	pool-min-size (min-idle)	The minimum number of idle database connections that are kept open at any time.	Having this value set to that of Maximum Size (above), which is the case by default, means the pool will always have a fixed number of connections and idle connections will never be closed. On very large JIRA installations, there may be some benefit in specifying a lower value for this setting than that of Maximum Size , to conserve resources.	Value of Maximum Size
Initial Size	pool-initial-size	The initial number of database connections opened in the pool.	This setting is not usually configured (other than the default value of 0), since a number of database connections are quickly created when JIRA starts up.	0 (when not specifically configured)

Maximum Wait Time	pool-max-wait	The length of time (in milliseconds) that JIRA is allowed to wait for a database connection to become available (while there are no free ones available in the pool), before returning an error.	Specifying a value of '-1' means that Tomcat will wait indefinitely. You should specify a time here which is long enough to allow for any contention spikes, but short enough that users will receive a meaningful error rather than just getting no response or a browser time out.	30000
Advanced settings				
Generally, changing the settings below are not usually required. Refer to the Apache DBCP documentation if r				
Pool Statements	pool-prepared-statements	Enable the pooling of prepared statements for the database connection pool.	Do not amend the default value of false, as it will cause exceptions. For more information see JRASERVER-44908 - DBPC configuration pool-prepared-statements leads to Statement Leak CLOSED	false (when i specific onfig
Maximum Open Statements	max-open-prepared-statements	The maximum number of open statements that can be allocated from the statement pool at the same time.	Do not amend the default value, as it will cause exceptions.	0 (when i specific onfig
Validation Query	validation-query	The SQL query that will be used to validate connections from this pool. If specified, this query MUST be an SQL SELECT statement that returns at least one row.	See Surviving connection closures fo r more information.	select ' (for My (otherw specific onfig

Validation Query Timeout	validation-query-timeout	The length of time (in seconds) that the system should wait for a validation query to succeed before it considers the database connection broken.	<p>The length of time should be quite short as the validation query should be designed to do a minimum amount of work.</p> <p>If you specify a Validation Query above, then you should specify a value for the Validation Query Timeout too. If not, a value of '-1' is assumed, which results in the system waiting indefinitely until a validation query succeeds against a broken database connection, which it never will.</p> <p> This should only be done for MySQL. Using a Validation Query Timeout on any database other than MySQL will cause significant problems with the JIRA instance.</p>	3 (for MySQL) (other specific configurations)
Test On Borrow	pool-test-on-borrow	<p>Tests if the database connection is valid when it is borrowed from the database connection pool by JIRA.</p> <p>If the database connection is broken, it is removed from the pool.</p>	<p>This value should always be 'false' as JIRA borrows a connection for each database operation.</p> <p>If you continue to have problems with database connections closing, try setting this option to 'true'. However, this should only be used as a last resort and only in the event that decreasing the value of Time Between Eviction Runs has not reduced or prevented problems with database connections closing.</p>	True (specific configurations however does not affect Validation Query explicit specific except MySQL has a Validation Query, will the have a

Test On Return	pool-test-on-return	<p>Tests if the database connection is valid when it is returned to the database connection pool by JIRA.</p> <p>If the database connection is broken, it is removed from the pool.</p>	<p>This value should always be 'false' as JIRA returns borrowed connections for each database operation.</p>	<p>false (when specified in config)</p>
Test While Idle	pool-test-while-idle	<p>Periodically tests if the database connection is valid when it is idle.</p> <p>If the database connection is broken, it is removed from the pool.</p>	<p>This should be set to 'true' for MySQL.</p> <p>By default, MySQL database servers close database connections if they are not used for an extended period of time. This causes problems with JIRA installations (which use MySQL databases) that are largely inactive for long periods, e.g. overnight. Setting this to 'true' will work around this behavior.</p> <p>Test While Idle only needs to be specified if you have specified a Validation Query above.</p>	<p>true (for MySQL)</p> <p>false (when specified in config)</p>

Time Between Eviction Runs	<code>time-between-eviction-runs-millis</code>	<p>The number of milliseconds to sleep between runs of the idle object eviction thread. When non-positive, no idle object eviction thread will be run.</p> <p>The eviction thread will remove idle database connections when the number of idle connections exceeds Minimum Idle/Size (above).</p>	<p>This should be set to a positive but largish value for MySQL so the evictor runs and tests connections. A reasonable value would be 300000 (5 minutes).</p> <p>✔ If you continue to have problems with database connections closing, try setting this option to a lower value.</p>	<p>300000 (for MySQL)</p> <p>5000 (for HS)</p> <p>(otherwise specific onfig)</p>
Minimum Evictable Idle Time	<code>min-evictable-idle-time-millis</code>	<p>The minimum amount of time an object may sit idle in the database connection pool before it is eligible for eviction by the idle object eviction (if any).</p>		<p>60000 (for MySQL)</p> <p>4000 (for HS)</p> <p>(otherwise specific onfig)</p>

Remove Abandoned	pool-remove-abandoned	<p>Flag to remove abandoned database connections if they exceed the Removed Abandoned Timeout (be low).</p> <p>If an internal failure occurs, it is possible that JIRA may borrow a connection and never return it. If this happens too often, then the pool may run short of database connections, causing JIRA's performance to degrade or JIRA to fail altogether.</p>	<p>This value should be set to 'true'.</p> <p>This will allow the pool to recover any abandoned connections and prevent this affecting system performance.</p>	true
Remove Abandoned Timeout	pool-remove-abandoned-timeout	The length of time (in seconds) that a database connection can be idle before it is considered abandoned.		300

*  **Please note:**

- JIRA writes elements with their default values (in the right-hand column of the table above) to the `dbconfig.xml` file after:
 - You have run through the [JIRA setup wizard](#) or
 - You use the **Advanced** tab of the JIRA configuration tool to configure/tune your database connection — even when the leftmost checkboxes of options associated with these elements have not been selected.
- The exception to this are elements whose values have '(when not specified in `dbconfig.xml`)' indicated below them. These elements are:
 - Not written to the `dbconfig.xml` file after running through the [JIRA setup wizard](#).
 - Only written to the `dbconfig.xml` file by:
 - Manually writing them into this file.
 - Using the Advanced tab of the JIRA configuration tool, selecting the leftmost checkboxes of the options associated with these elements and specifying values for these options.

- When '(when not specified in `dbconfig.xml`)' is indicated below a default value in the right-hand column of the table above, then this default value is assumed, even when it is not present in the `dbconfig.xml` file.

Monitoring the connection pool

JIRA provides a view of its database connection usage via the 'Database Monitoring' page. See [Monitoring database connection usage](#) for more information.

Surviving connection closures

When a database server reboots or a network failure has occurred, all connections in the database connection pool are broken. To overcome this issue, JIRA would normally need restarting.

However, database connections in the database connection pool can be validated by running a simple SQL query. If a broken database connection is detected in the pool, a new one is created to replace it.

To do this, you need to specify an optional `<validation-query/>` element (in the `dbconfig.xml` file of your [JIRA home directory](#)), whose content is the query which validates connections in the database connection pool. See the following procedure for details.

Ensuring JIRA validates connections to its database

1. Shut down JIRA (or the Tomcat installation running JIRA).
2. Edit the `dbconfig.xml` file at the root of your [JIRA home directory](#) or use the **Advanced** tab of the [JIRA configuration tool](#) to configure the relevant settings.
3. Configure the validation query for your type of database:
 - If editing the `dbconfig.xml` file, add the `<validation-query/>` element with the appropriate validation query for your type of database, as shown in the example below for MySQL. (See [Determining the validation query](#) below for details.)

```

<?xml version="1.0" encoding="UTF-8"?>

<jira-database-config>
  <name>defaultDS</name>
  <delegator-name>default</delegator-name>
  <database-type>mysql</database-type>
  <jdbc-datasource>

    <url>jdbc:mysql://dbserver:3306/jiradb?useUnicode=true&characterEncoding=UTF8&sessionVariables=storage_engine=InnoDB</url>

    <driver-class>com.mysql.jdbc.Driver</driver-class>
    <username>jiradbuser</username>
    <password>password</password>
    <pool-min-size>20</pool-min-size>
    <pool-max-size>20</pool-max-size>
    <pool-max-wait>30000</pool-max-wait>

    <validation-query>select 1</validation-query>

    <min-evictable-idle-time-millis>60000</min-evictable-idle-time-millis>

    <time-between-eviction-runs-millis>300000</time-between-eviction-runs-millis>

    <pool-max-idle>20</pool-max-idle>
    <pool-remove-abandoned>true</pool-remove-abandoned>

    <pool-remove-abandoned-timeout>300</pool-remove-abandoned-timeout>

    <pool-test-while-idle>true</pool-test-while-idle>
    <pool-test-on-borrow>false</pool-test-on-borrow>
    <validation-query-timeout>3</validation-query-timeout>

  </jdbc-datasource>
</jira-database-config>

```

- If using the [JIRA configuration tool](#), on the **Advanced** tab, select the **Validation Query** checkbox and enter the appropriate validation query for your type of database. (See [Determining the validation query](#) below for details.)
4. Specify a validation query timeout for your validation query, whose value is the appropriate length of time (in seconds) that the system should wait for a validation query to succeed before the system considers the database connection broken:
 - If editing the `dbconfig.xml` file, add the `<validation-query-timeout/>` element with the appropriate length of time (in seconds). ⚠ This should **only** be done for MySQL.
 - If using the [JIRA configuration tool](#), on the **Advanced** tab, select the **Validation Query Timeout** checkbox and enter the appropriate length of time (in seconds).
 5. You may wish to specify the following options, which relate to the above validation query options (see [Tuning database connections - connection pool settings](#) section for details):

JIRA configuration tool 'Advanced' tab option	Element in <code>dbconfig.xml</code>
Test While Idle	<code>pool-test-while-idle</code>

Time Between Eviction Runs	<code>time-between-eviction-runs-millis</code>
Minimum Evictable Idle Time	<code>min-evictable-idle-time-millis</code>

6. Save your edited `dbconfig.xml` file (or click the **Save** button if using the [JIRA configuration tool](#)).
7. Restart JIRA (or the Tomcat installation running JIRA).

i Please Note: If you continue to have problems with connections closing, you may need to set the `time-between-eviction-runs-millis` parameter to a lower value or as a last resort, set `test-on-borrow` to `true`. For more information about `test-on-borrow`, see [Tuning database connections - connection pool settings](#) section.

Determining the validation query and timeout

Different database types have slightly different SQL syntax requirements for their validation query. The validation query should be as simple as possible, as this is run every time a connection is retrieved from the pool. The validation query timeout should only be set for MySQL.

The following validation queries are recommended for the following types of databases:

Database type	Validation query	Validation query timeout
MySQL	<code>select 1</code>	3
Microsoft SQL Server	<code>select 1</code>	N/A
Oracle	<code>select 1 from dual</code>	N/A
PostgreSQL	<code>select version();</code>	N/A

⚠ If the **Validation query timeout** is used on any database other than MySQL it will cause significant problems with the JIRA instance.

Result

You should now be able to recover from a complete loss of all connections in the database connection pool without the need to restart JIRA or the application server running JIRA.

⚠ Performance considerations:

- Setting this option has a performance impact. The overall decrease in performance should be minimal, as the query itself is quick to run. In addition, the query will only execute when you make a connection. Thus, if the connection is kept for the duration of a request, the query will only occur once per request.
- If you are running a large JIRA installation, you may wish to assess the performance impact of this change before implementing it.

Switching databases

JIRA's data can be migrated from one database to:

1. A different database on the same database server,
2. The same database type on a different server (e.g. from one PostgreSQL server to another PostgreSQL server) or
3. A different type of database server (e.g. from a MySQL server to a PostgreSQL server).

⚠ For migrating JIRA to another server, please refer to the [Migrating JIRA to another server](#) document instead.

To do this, follow the appropriate procedure:

- [Migrating JIRA's data to the same type of database](#) (covers scenarios 1 and 2 above)
- [Migrating JIRA's data to a different type of database server](#) (covers scenario 3 above)

Migrating JIRA's data to the same type of database

Use this procedure to migrate JIRA's data to:

- A different database on the same database server, or

- The same database type on a different database server (e.g. from one PostgreSQL server to another PostgreSQL server).
1. Use your database server's native tools to either:
 - Copy your JIRA database to a new database on the same database server installation, or
 - Copy/migrate your JIRA database to a new database of the same type on a different database server installation.
- Please Note:**
- If you are unable to do either of these tasks, use the [Migrating JIRA's database to a different type of database server](#) procedure (below) instead.
 - You could use this procedure to migrate JIRA's data to a different type of database server (e.g. MySQL to PostgreSQL). However, you would need to find tools that support these processes. Furthermore, Atlassian does not provide support for this strategy.
2. Once your new database has been populated with JIRA's data, shut down your JIRA server.
 3. Make a backup of your [JIRA home directory](#) and [JIRA installation directory](#).
 4. Reconfigure your JIRA server's connection to your database:
 - If you installed a 'Recommended' distribution of JIRA, you can use the [JIRA configuration tool](#) (by running `bin/config.sh` (for Linux/Solaris) or `bin\config.bat` (for Windows) in your [JIRA installation directory](#)), which provides a convenient GUI that allows you to reconfigure JIRA's database connection settings.
 - If any of the following points applies to your situation, you need to manually configure the `dbconfig.xml` file in your [JIRA home directory](#). Refer to the appropriate database configuration guide in the [Connecting JIRA to a database](#) section for the manual configuration instructions.
 - You have a console-only connection to your JIRA server
 - You would prefer to configure your database connection manually (for custom configuration purposes).

Migrating JIRA's data to a different type of database server

Use this procedure to migrate JIRA's data to a different type of database server (e.g. from a MySQL server to a PostgreSQL server).

✔ You can also use this procedure if your JIRA installation is currently using the internal H2 database (which is only supported for evaluating JIRA) and you need to switch your JIRA installation across to using a [supported database](#) (which are supported for JIRA installations used in a production environment).

1. Create an export of your data as an XML backup. See [Backing up data](#) for details.
2. Create a new database on your new database server to house JIRA's data. See the appropriate database configuration guide in the [Connecting JIRA to a database](#) section for the database creation instructions.
3. Shut down your JIRA server.
4. Make a backup of your [JIRA home directory](#) and [JIRA installation directory](#).
5. Delete the `dbconfig.xml` file in your [JIRA home directory](#).
6. Restart JIRA and you should see the first step of the [JIRA setup wizard](#) for configuring your database connection.
7. Configure JIRA's connection to your new database (created in step 2 above) and click the 'Next' button.
8. On the 'Application Properties' setup page, click the '**import your existing data**' link and [restore your data](#) from the XML backup created in step 1 above.

Known issues

- [Database migration to SQL Server fails because of duplicate entries](#)

Installing JIRA Data Center

These instructions are applicable for installing JIRA Software Data Center or JIRA Service Desk Data Center on your own hardware. You can also [install Data Center in Amazon Web Services](#).

Learn more about what Jira Software Data Center and Jira Service Desk Data Center provide on our website.

Before you begin

Before you install JIRA Data Center, you need to answer a few questions.

What is JIRA Data Center?	<p>▼ Tell me more...</p> <p>We've created a handy quick start guide to help you understand what's going on with DC, and what you need to do to get it up and running in no time!</p> <p>Take a look at Quick start for JIRA Data Center.</p>
How do I get it?	<p>Tell me more...</p> <p>You'll need two things to get started – an installer, and a license.</p> <ul style="list-style-type: none"> • To download the installer, see JIRA Software, or JIRA Service Desk. • As for the license, get the JIRA Software Data Center license or JIRA Service Desk Data Center license, or create an evaluation license. <p>If you're installing JIRA from scratch, you'll first need to apply the evaluation license so make sure you create it.</p>
What are the prerequisites?	<p>Tell me more...</p> <p>Supported platforms</p> <p>Supported operating systems, databases, etc., are the same as for the Server installation, and you can see them here: Supported platforms.</p> <p>Node requirements</p> <p>Requirements specific to Data Center include requirements for nodes that create the cluster:</p> <ul style="list-style-type: none"> • Each node is a separate machine (physical or virtual). They don't need to be identical, but should be as similar as possible for consistent performance. • All nodes are running the same version of JIRA. You'll be copying JIRA from one node to another, so this shouldn't be a problem. They use the same timezone, and have the current time synced. You can use <code>ntpdate</code> to set this up. • All nodes share a common database, also installed on a separate machine. • All nodes can access the shared home directory. You can set it up using NFS, or a similar solution. We'll mention it in this guide.
Do I need a load balancer?	<p>Tell me more...</p> <p>Yes. JIRA Data Center relies on a load balancer to balance the traffic between the nodes, and this guide assumes that you already have one set up. You can use a load balancer of your choice, just make sure it meets these requirements:</p> <ul style="list-style-type: none"> • Supports "cookie based session affinity", also known as "sticky sessions". • Can route HTTP/HTTPS traffic to one of the available nodes. • Can determine whether a node is available or not, and route requests to other nodes if needed. • All Atlassian applications and other REST clients must access your nodes through the load balancer. <p>Or you can just turn your proxy into a load balancer.</p> <p>Many bigger installations of JIRA already have a reverse proxy configured, and many reverse proxies can do load balancing as well. We've provided some examples on how to use your proxy as a load balancer. See Load balancer examples.</p>

1. Install or upgrade your JIRA instance

JIRA Data Center is available for JIRA 7.0, or later. If you're not on this version yet, install or upgrade your JIRA instance.

[JIRA installation and upgrade guide](#)

Few things to know about:

- If you're installing your instance from scratch, generate a non-Data Center evaluation license at the setup stage, and update to the Data Center license when you're adding the `cluster.properties` file at step 3.
- If you're upgrading your instance, update to the Data Center license when you're adding the `cluster.properties` file at step 3.

2. Set up the shared directory

You'll need to create a remote directory that is readable and writable by all nodes in the cluster. There are multiple ways to do this, but the simplest is to use an NFS share.

1. Create a remote directory, accessible by all nodes in the cluster, and name it e.g. `sharedhome`.
2. Stop your JIRA instance.
3. Copy the following directories from the JIRA local home directory to the new `sharedhome` directory (some of them may be empty).
 - `data`
 - `plugins`
 - `logos`
 - `import`
 - `export`
 - `caches`

3. Configure your existing JIRA instance to work in a cluster

1. In the JIRA local home directory, create a `cluster.properties` file, with contents as follows:

Example cluster.properties file:

```
# This ID must be unique across the cluster
jira.node.id = node1
# The location of the shared home directory for all JIRA nodes
jira.shared.home = /data/jira/sharedhome
```

For more information and some additional parameters, see [Cluster.properties file parameters](#).

2. Start your instance, and apply the Data Center license.

4. Add the first node to the load balancer

The load balancer distributes the traffic between the nodes. If a node stops working, the remaining nodes will take over its workload, and your users won't even notice it.

1. Add the first node to the load balancer.
2. Restart the node, and then try opening different pages in JIRA. If the load balancer is working properly, you should have no problems with accessing JIRA.

5. Add the remaining nodes to the cluster

1. Copy the JIRA **installation directory** and the **local home directory** from the first node to this new node.
2. Ensure the new node can access (read and write) the shared home directory.
3. Edit the `cluster.properties` file, and change the node ID. All node IDs must be unique among nodes.
4. Start JIRA. It will read the configuration from the shared home directory, and start without any extra setup.
5. Take a look around the new JIRA instance. Ensure that issue creation, search, attachments, and customizations work as expected.
6. If everything looks fine, you can configure your load balancer to start routing traffic to the new node. Once you do this, you can make a couple of changes in one JIRA instance to see if they're visible in

other instances as well.

While adding your nodes to the cluster, you can check their status by going to



> **System** > **System info**. Your nodes will be listed in the **Cluster nodes** section.

Cluster.properties file parameters

In addition to the required parameters, the cluster.properties file allows you to configure some additional options, mostly related to EhCache.

▼ [Tell me more...](#)

Parameter	Required	Description/value
<code>jira.node.id</code>	Yes	This unique ID must match the username and the <code>BalancerMember</code> entry in the Apache configuration.
<code>jira.shared.home</code>	Yes	The location of the shared home directory for all JIRA nodes.
<code>ehcache.peer.discovery</code>	No	Describes how nodes find each other: <code>default</code> – JIRA will automatically discover nodes (recommended) <code>automatic</code> – JIRA will use the EhCache's multicast discovery. This is the historical method used by EhCache, but it can be difficult to configure, and is not recommended by Atlassian. If you choose automatic... If you set <code>ehcache.peer.discovery</code> <code>automatic</code> then you need to set the following parameters: <ul style="list-style-type: none"> • <code>ehcache.multicast.address</code> • <code>ehcache.multicast.port</code> • <code>ehcache.multicast.timeToLive</code> • <code>ehcache.multicast.hostName</code> For more info on these parameters, see Ehcache documentation .
<code>ehcache.listener.hostName</code>	No	The hostname of the current node for each communication. JIRA Data Center will resolve this internally if the parameter isn't set. If you have problems resolving the hostname of the network you can set this parameter.
<code>ehcache.listener.port</code>	No	The port that the node is going to be listening to (default is 40001). If multiple nodes are on the same host, or if this port is unavailable, you might need to set this parameter manually.

<code>ehcache.object.port</code>	No	The port on which the remote objects bound in the registry receive calls. Make sure you also open this port on your firewall. We recommend that you specify this parameter, as otherwise a random free port will be used.
<code>ehcache.listener.socketTimeoutMillis</code>	No	By default, this is set to the EhCache default.

Monitoring the health of your Data Center

Now that you got your Data Center up and running, we recommend that you keep monitoring its health right from the start. This will help you keep any problems from getting bigger and messing with your work, and you'll always know what's going on in the cluster.

JIRA Data Center is equipped with a set of health checks tools that let you monitor the whole cluster and each node individually, including all important settings. To access the health check tools, go to



> **System > Support Tools**. All health checks are listed in the **Instance health** tab.

Running the setup wizard

The JIRA setup wizard allows you to either set up a JIRA application for evaluation and demonstration purposes, or for production and testing.

To get started, access your new JIRA application in a browser, after you have [installed it](#). Your server will be available at the following URL, if you are using the default port: `http://<jira-server-name>:8080`.

i *The JIRA application setup wizard will only display the first time after you install your JIRA application. Once you have completed it, you cannot run it again. However, every setting configured in the setup wizard can be configured via the JIRA administration console.*

Evaluation and demonstration

If you want to evaluate or demonstrate a JIRA application, let us do most of the set up for you. We will help you set up an Atlassian account if you don't have one, and will generate an evaluation license for you. We'll also set up a H2 database for evaluation purposes (see [Supported Platforms](#)). The only requirement is you have a connection to the internet, as we'll need this to validate and generate Atlassian account details and your evaluation license.

Follow the steps [here](#).

Production and testing

If you want to set up a JIRA application for production or testing purposes before you upgrade, we recommend you follow the custom installation path. This will allow you to connect to your own database if required, and set up your email SMTP server. This path can also be followed if you don't have a connection to the internet. You'll be able to manually paste in a license key.

Follow the steps [here](#).

Evaluation and demonstration setup

The screenshot shows the 'JIRA setup' screen. At the top right, there is a 'Language' dropdown menu. Below the title, a line of text reads: 'Select whether you'd like to set up JIRA in demonstration or production mode.' There are two main options presented in rounded rectangular boxes. The first option, 'Set it up for me', is highlighted with a light blue background and features a laptop icon. Its text states: 'This is the quick setup for demonstration and evaluation environments. We'll do most of the JIRA configuration for you, but you **need to be online with a working internet connection** so we can generate an evaluation license for you. You can change the configuration later if you need to.' The second option, 'I'll set it up myself', is in a white box with a light blue border and features a wrench and screwdriver icon. Its text states: 'Set up and configure your JIRA instance manually. This is recommended for production environments, or if you don't have a working internet connection.' At the bottom right of the screen is a blue 'Next' button.

1. Choose the language you would like the JIRA application setup and user interface to appear in by selecting the preferred **Language**. Note:

- As soon as you choose a language from the **Language** drop-down list, the JIRA application user interface will switch to that language.
- Be aware that some languages may have more comprehensive translations than others.

2. Select **Set it up for me** and click **Next**.

3. Enter your Atlassian ID email address, or if you don't have an Atlassian ID account, enter an email address you'd like to use and have access to, and click **Next**.

4. We'll validate your account, and create a new one if needed. Remember the details you use for your Atlassian ID account, as these will be the same credentials for your JIRA system administrator. You can change the system administrator details within JIRA when you're up and running. Check out [Managing Global Permissions](#) and [Managing Users](#) for more information. Select **Next** to generate your license.

5. Select **Next** to finish the setup process. This may take a minute or two.

6. Once the setup is complete, you're ready to get started! Select **Launch JIRA** to get going!

✔ That's it!

Production and testing setup

This screenshot is identical to the one above, showing the 'JIRA setup' screen with the 'Set it up for me' option highlighted in light blue and the 'I'll set it up myself' option in white with a light blue border. The 'Next' button is visible at the bottom right.

1. Choose the language you would like the JIRA application user interface to appear in by selecting the preferred **Language**. Note:

- As soon as you choose a language from the **Language** dropdown list, the JIRA application user interface will switch to that language.
- Be aware that some languages may have more comprehensive translations than others.

2. Select **I'll set it up myself** and click **Next**.

3. Configure a database for JIRA.

Choose between connecting JIRA to the bundled database or your own database.

Database Connection	Recommended for	Instructions	Notes
Bundled database	Evaluations only	Go to the next step. The bundled H2 database will be automatically configured by the setup wizard.	<ul style="list-style-type: none"> The H2 database is suitable for evaluation and demonstration purposes only. We recommend connecting to a supported database for production environments.
Your own database	Production use	<ol style="list-style-type: none"> Choose a database. See our list of supported databases first. Configure the database connection. If you need help, see the guides on Connecting JIRA to a database. Note, the fields displayed on this screen are identical to those on the JIRA configuration tool. 	<ul style="list-style-type: none"> Your external database must be a newly-created (or empty) database. Database connection pool — You cannot configure your database connection pool size through the setup wizard. You can do this subsequently using the JIRA configuration tool or manually (described on each specific database configuration guide). MySQL database — The MySQL driver is not bundled with JIRA (see Connecting JIRA to MySQL). You need to copy the driver into the lib folder of your JIRA installation and restart JIRA/JIRA service before completing the setup wizard.

4. If you're connecting to your own database, click **Test connection** to make sure JIRA can connect. Click **Next** when you're ready to proceed.

5. You need to configure the Title, Mode, and Base URL for your instance:

Setting	Instructions	Notes
Application Title	Choose a title that helps identify your installation and its purpose.	<ul style="list-style-type: none"> The application title will be displayed on the login page and the dashboard. After you have completed the setup wizard, you may also want to configure the logo and color scheme of your installation.
Mode	Choose a mode that suits how you use your issue tracker.	<ul style="list-style-type: none"> Setting the mode to public enables public signup. Note, that allowing anyone to sign up can cause you to exceed the user limit on your JIRA application license. A public issue tracker can be useful for gathering feedback and bug reports directly from customers. A private issue tracker may be more suitable for tracking the development progress of your team.

Base URL	Specify the base URL that users will use to access your instance.	<ul style="list-style-type: none"> You can only configure JIRA to respond to a single URL and this setting must match the URL that your users request for accessing your JIRA instance. You cannot (for example) have a different hostname or URL for internal and external users. Any mismatch between this Base URL setting and the URL requested by your JIRA application users will cause problems with dashboard gadgets. This URL is also used in outgoing email notifications as the prefix for links to JIRA issues.
----------	---	---

Further information:

- If you need to change these settings after setting up your application, you can configure them via the JIRA administration console. For details, see [Configuring JIRA options](#).
- JIRA will store your automated backups, file attachments and indexes in your [JIRA home directory](#).

Click **Next** when you've configured all the application properties to your liking.

6. You are required to enter a JIRA application license key before you can use your application. If you don't have a JIRA application license key, you can get the setup wizard to create an evaluation license for you. Evaluation license keys will allow you to use a fully functional installation for 30 days.

License keys for Atlassian applications are linked to your account at my.atlassian.com. If you don't have a my.atlassian.com account, you can create one and get the setup wizard to create an evaluation license for you.

7. Enter the details for the administrator account for the installation. The account will be granted the [JIRA system administrator permission](#).

You can create additional JIRA system administrator and JIRA administrator accounts after you have set up JIRA. Click **Next** when you've entered the details.

8. Set up your email SMTP server. This step is optional. You can configure email notifications after you have set up JIRA if you wish.

If you want to configure email notifications at this stage, you will need to set up a connection to a mail server. See this page for further instructions: [Configuring JIRA's SMTP Mail Server to Send Notifications](#). Click **Finish** to complete the setup.

 **Congratulations, you have completed setting up your new JIRA application installation!**

Detailed information on using and administering JIRA and your JIRA applications can be found in the rest of the Administering JIRA applications documentation.

Licensing and application access

To grant users log in access to a JIRA application, the application must first be [licensed](#), and secondly, the application must have at least one default group assigned to it. Any users added to this group will be able to log in to the application. This is called **application access**. Your JIRA application may have more than one group assigned to it, and a user may be a member of more than one group assigned to the application, but they will only count as one licensed user for that application. This is covered in more detail on [Managing users access to JIRA applications](#).

On this page:

- [Installing your first application and application access](#)
- [Adding additional JIRA applications](#)
- [Running multiple JIRA applications](#)

Installing your first application and application access

When you [install your first application](#) and license it (you may obtain a license as part of the installation process, or directly from my.atlassian.com), JIRA will create two user groups, and add you to both of them. The first group is the **jira-administrators** group, and this is the group that grants you the **JIRA Administrator global permission** and grants you administrative privileges. The second group created depends on the JIRA application you have installed. They are listed below:

JIRA application	User group created when the product is licensed
JIRA Core	jira-core-users
JIRA Software	jira-software-users
JIRA Service Desk	jira-servicedesk-users

Both of these groups are assigned to the application you installed on the **Application access** page, and the second group is also assigned as the [default group](#). This means any subsequent users you create for the application will be added automatically to this group.

Adding additional JIRA applications

You may have a requirement to add another JIRA application to your instance. You can [install additional applications](#) through your **Version & licensing** page. This allows you to locate the most up-to-date version of the application and install it. Once installed, you'll still need to ensure the new application is licensed. Once licensed, JIRA will create a default group for the application, but you will not be added to this group automatically. To gain full access to the application, you should add yourself to a group associated with the application.

Running multiple JIRA applications

Each JIRA application comes complete with a specific set of features and functions, which tailors the experience delivered to its users. Every user in JIRA will have access to an application based on their [membership of groups](#). A user may have access to all the applications, or only one application. If a user has access to an application, they will count as a licensed user for that application. For example, if a user belongs to a group for JIRA Software and a group for JIRA Service Desk, they will count as a licensed user for both JIRA Software and JIRA Service Desk.

When you have multiple applications installed, by default, all users will be able to view all projects (unless there are specific project permissions set up that prohibit this). This means a JIRA Core user will be able to see all JIRA Software and JIRA Service Desk projects. However, as they are not licensed for these applications, they will not be able to see any features or functions that are specific to that application. For example, a JIRA Core user viewing a JIRA Software project would be able to see the project and its issues, but would not be able to see any JIRA Software specific features, like Agile boards, development information, or release information. These features can only be viewed by a JIRA Software user. It's important to note that JIRA Core does not have any specific features or functions that cannot be viewed and/or actioned by other users. This means that if you are a JIRA Software or JIRA Service Desk user, you can already view and work on a JIRA Core project. You do not need to have specific application access for JIRA Core, and therefore do not need to consume a license. View the [JIRA applications and project types overview](#) page for more information on what licensed users can and cannot view and action on projects from other applications.

License compatibility

Each JIRA application you install must have a unique license. There are various license types available. Some of these license types are incompatible with each other. If you try to install incompatible license types, JIRA will present you with an error. To resolve this, you should select compatible license types, obtain them and install them. You should make sure you remove the incompatible license type first.

You can manage your JIRA licenses on the [Versions & licenses](#) page.

Commercial licenses

A **commercial** license is a paid license that allows you to run a JIRA application and add users.

- All commercial licenses will work with each other if you have more than one JIRA application installed.
- You can mix commercial licenses with evaluation licenses.
- You cannot mix commercial licenses with other license types (e.g. Data Center or Academic licenses).

Data Center licenses

Data Center is a deployment option providing high availability and performance at scale for your JIRA applications.

- If you have installed a Data Center license for an application and configured the application for Data Center, all subsequent licenses must be Data Center licenses.
- If you have any other type of license and want to install a Data Center license, this can be done.
- If you have more than one JIRA application, and you want to set them up for Data Center, all the applications must have Data Center licenses. You cannot mix a Data Center license with any other type.

Evaluation licenses

An evaluation (or "trial") license lets you try the full functionality of a JIRA application for a fixed period of time (typically 30 days). When the trial ends, the application stops functioning until you install a paid license.

Unpaid licenses

Unpaid licenses are available for evaluators, not for profit organizations, charities and students.

- All unpaid licenses will work with each other if you have more than one JIRA application installed.
- You cannot mix unpaid licenses with commercial (paid) licenses when you have more than one JIRA application installed.

Extending JIRA applications

JIRA is very flexible, and has a number of extension points where JIRA's data can be queried or its functionality extended. This page provides an overview of the mechanisms available for extending JIRA.

i JIRA add-ons: For information on installing or enabling existing add-ons, please read the [Managing add-ons](#) documentation. To learn about creating your own add-ons, see [developing add-ons with the Atlassian Plugin SDK](#).

Note that an add-on that specifically plugs into the architecture of an Atlassian application such as JIRA is sometimes called a **plugin**, although the terms 'plugin' and 'add-on' are often used interchangeably.

Custom field types	JIRA comes with various custom field types defined. New types can be written and plugged into JIRA. See the How to create a new Custom Field Type 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 customized by add-ons. You can write your own user format add-on to change the display of user details in JIRA, e.g. display a profile picture. See the User Format Plugin Module for more information.
Gadgets	New gadgets can be created by writing an XML descriptor file, packaged as an Atlassian plugin . See Tutorial - Writing gadgets for JIRA 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 summarizing JIRA's data.

Workflow functions and conditions	<p>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 provides hooks where you can plug in your own behavior:</p> <ul style="list-style-type: none"> • 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. <p>See the Working with workflows for details on workflow post-functions, conditions, and validators. Once written, these can be packaged as plugins and reused.</p>
Issues and projects	<p>On the 'View Issue' page, some issue information (comments, change history) is displayed. Likewise, the 'Browse Project' page contains separate sections, listed on the far left, for different types of project information.</p> <p>By writing a plugin, you can add new issue or project sections (that will be listed in the left panel) to JIRA. For instance, you may wish to display project/issue data pulled in from an external source. This is how the JIRA Subversion plugin works.</p>
Listeners (Note this is not configurable in JIRA Cloud applications)	<p>JIRA has a complete event subsystem, which fires events whenever anything happens. For example, an <code>ISSUE_CREATED</code> event is fired whenever an issue is created. A listener is just a class that implements a <code>JiraListener</code> 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 <code>MailListener</code>. This is useful when you want to drive or affect external systems from events, which occur within JIRA — usually used to <i>push</i> data into outside systems. For more information, read the listeners documentation.</p>
Services	<p>Services are classes that implement the <code>JiraService</code> interface. When installed, you specify an update period, and JIRA will call the <code>run()</code> method of your service periodically. A sample service is <code>POPCommentService</code>. 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 <i>pull</i> data into JIRA from outside systems. For more information, see the services guide.</p>

Integrating with development tools

Connecting JIRA Software to compatible development tools provides your

team with a range of functionality and information related to your development work. You can connect to multiple instances of the same development tool, but it's recommended you set up one of these instances as the primary link, meaning JIRA Software will query that instance first when looking for that sort of information. For example, if you connect to Bitbucket Server instance A and Bitbucket Server instance B, and you make Bitbucket Server instance A the primary instance, when JIRA Software needs info relating to Bitbucket Server it will query instance A.

Integration features

These are the features that become available when you connect JIRA Software to the development tools listed below. We recommend that you use the latest version of each application – if you're using earlier versions, see the [version matrix](#) to find out which features are available.

On this page:

- [Integration features](#)
- [How it works](#)
- [Supported versions](#)
- [Development tools configuration for a project](#)
- [Set up JIRA Software with development tools](#)
- [Troubleshooting](#)

Development panel on issues

The Development panel is shown on the View Issue screen and provides the following functionality:

- [Bitbucket Cloud and Bitbucket Server](#): view and create branches, view commits, and view and create pull requests
- [FishEye/Crucible](#): view commits and branches, view and create reviews
- [Bamboo](#): view the status of builds and deployments
- [GitHub and GitHub Enterprise](#): view commits, branches and pull requests

For more information about using the Development panel, see the [JIRA Software documentation](#).

Development

2 branches	Updated 14/Jan/14 11:51 AM
4 commits	Latest 14/Jan/14 8:44 PM
1 pull request MERGED	Updated 14/Jan/14 8:44 PM
3 builds ✔	Latest 21/Jan/14 11:51 AM

[Deployed to Production](#)

[Create branch](#)

Workflow triggers

Workflow triggers can help keep JIRA Software issues synchronized with the information in your development tools – FishEye/Crucible, Bitbucket, and GitHub.

Instead of relying on developers to manually update the status of an issue after committing code, completing reviews, or creating branches, you can configure triggers in your workflow to automatically transition issues when these events occur in your development tools. For example, you could configure a trigger to automatically transition an issue from 'To Do' to 'In Progress' when a branch is created.

See [Configuring workflow triggers](#) for instructions on setting up workflow triggers.

There is a known issue where the 'Branch created' event isn't supported for GitHub, which is being tracked under

JSWSERVER-144

73 - Implement 'Create Branch' feature in DVCS connector plugin for Github integration **RESOLVED**

— please keep this in mind when configuring trigger events.

Add trigger

<div style="border: 1px solid #ccc; background-color: #e6f2ff; padding: 5px; margin-bottom: 5px;">  <p>Pull request created Automatically transitions the issue when a related pull request is created in a...</p> </div> <div style="border: 1px solid #ccc; background-color: #e6f2ff; padding: 5px; margin-bottom: 5px;">  <p>Pull request declined Automatically transitions the issue when a related pull request is declined in a...</p> </div> <div style="border: 1px solid #ccc; background-color: #e6f2ff; padding: 5px; margin-bottom: 5px;">  <p>Branch created Automatically transitions the issue when a related branch is created in a connected...</p> </div>	<div style="border: 1px solid #ccc; background-color: #e6f2ff; padding: 5px; margin-bottom: 5px;">  <p>Pull request merged Automatically transitions the issue when a related pull request is merged in a...</p> </div> <div style="border: 1px solid #ccc; background-color: #e6f2ff; padding: 5px; margin-bottom: 5px;">  <p>Pull request reopened Automatically transitions the issue when a related pull request is reopened in a...</p> </div> <div style="border: 1px solid #ccc; background-color: #e6f2ff; padding: 5px; margin-bottom: 5px;">  <p>Commit created Automatically transitions the issue when a related commit is made in a connected...</p> </div>
---	--



Tell us what other triggers you'd like to see
We are very interested in how you'd like to automate your workflow.

Next
Cancel

Release Hub

The Release Hub shows the progress of a version, so you can determine which issues are likely to ship at a glance. The commits, builds, and deployments related to each issue are shown, helping you to spot potential development issues that could cause problems for a release.

When you are ready, you can also release the version from the Release Hub. Doing this marks the version as complete, moves incomplete issues to other versions, and triggers release builds (if JIRA Software is integrated with Bamboo).

Read more about the Release Hub here: [Checking the progress of a version](#)

Version 2.0 UNRELEASED Release
 Start: 03Nov14 Release: 15Feb15 [Release Notes](#)

48 Warnings 273 Issues in Version 262 Issues Done 7 Issues in Progress 4 Issues to Go

Warnings indicate when the status of a JIRA issue doesn't reflect related development activity. For example: an issue marked complete that has an open pull request should be marked as still being in progress. Manage Warnings

Unreviewed Code
 These issues have been marked complete but the commits are not part of a pull request or review.

P	T	Key	Summary	Assignee	Status	Development
High	Defect	SSP-1563	UI is not loading in IE8	Andrew Swan	DONE	1 commit
High	Defect	SSP-1919	Incorrect permissions for new report	Andrew Swan	DONE	1 commit
Medium	Story	SSP-1555	As a developer, I want to view the build status for an issue	Bruce Templeton	DONE	5 commits
High	Defect	SSP-1600	Missing error message when Bamboo is unavailable	Eduardo Soares	DONE	1 commit

How it works

When the Atlassian development tools are integrated with JIRA Software, a user simply needs to supply an issue key for the issue to be automatically linked:

- Commits are linked automatically if the issue key is included in the commit message.
- Branches are linked automatically if the issue key is included in the branch name.
- Pull requests are linked automatically if the issue key is included in the pull request's title or in the source branch name.
- Reviews are linked automatically if the issue key is included in the title of the review, or if the [issue is linked](#) from the review.
- Builds and deployments are linked automatically if a commit involved in the build has the issue key in its commit message.

When triggers are configured in the workflow for your project, particular events published by the developer tools automatically transition issues.

There are some details and known issues:

- When a user attempts to access one of the details dialogs, for commits, reviews, builds or pull requests, JIRA Software checks that they have the appropriate permissions to view the information in the dialog. It does this using the user authentication that is configured in the Application Link.
- The details dialogs (ex: for commits) may display duplicates, although the number of unique items are reported at the top of the dialog and in the Development panel summary. Duplicate commits, for example, can arise from having both Bitbucket Server and FishEye linked to JIRA Software, and FishEye in turn connected with Bitbucket Server, so that FishEye indexes, and reports, commits.
- Users who can see summarized data in the Development panel may not have permission to see in the details dialogs (for example, for branches, commits and pull requests) all the information that contributed to the summaries. That is, the details dialogs respect the access permissions that users have in the connected applications.
- If commits linked to the issue are involved with a Bamboo build that fails, the first successful build that follows will be reported, even though the original commits are no longer involved with that build.
- The Development panel replaces the Source, Commits and Builds tabs, as well as the Deployment panel, in an issue. So, for example, you won't see the Source tab, and commits in Bitbucket Server will be accessible from the Development panel. However, if a connected application is older than the supported version, information from that application will continue to be displayed in those tabs rather than in the Development panel.

Supported versions

The table below shows the minimum development tool version required for each integration feature in JIRA Software.

JIRA	FishEye / Crucible	Bamboo	Bitbucket Cloud	Bitbucket Server	GitHub	GitHub Enterprise	Integration feature
6.4+ or JIRA Cloud	3.3+/3.3+	5.4+	Current	Bitbucket Server 4.0+ (Stash 2.10)	Current	11.10.290+	Release Hub: <ul style="list-style-type: none"> View issues and development information (from Bamboo, Bitbucket, or FishEye/Crucible) for a version. See warnings that help you reconcile what is happening in development with JIRA data. Release a version, manage scope between versions, trigger release builds from place in JIRA.
6.3.3+ or JIRA Cloud	3.5.2	N/A	Current	Bitbucket Server 4.0+ (Stash 3.2.0)	Current	11.10.290+	Workflow triggers <ul style="list-style-type: none"> Transition JIRA issues from within Bitbucket Server or FishEye/Crucible
6.2+	3.3+/3.3+	5.4+	Current	Bitbucket Server 4.0+ (Stash 2.10+)	Current	11.10.290+	Development panel: <ul style="list-style-type: none"> Bitbucket: create and view branches and pull requests from an issue, view commits FishEye/Crucible(Git/Subversion/Perforce/CVS): browse, search repos, view commits and branches, create and view reviews Bamboo: view the status of builds and deployments for the related issues
6.1.+	N/A	N/A	Current	Bitbucket Server 4.0+ (Stash 2.8.x)	N/A	N/A	Development panel: <ul style="list-style-type: none"> Create branches in Bitbucket

Development tools configuration for a project

The Development Tools section of the administration screen for a project gives you an overview of the development tools that are connected to your JIRA Software instance, and of the users who can use the integrations between JIRA and those tools. Users must have access to the JIRA Software application to be able to see the Development panel. By default, anonymous users (those who are not logged in) and users without explicit JIRA Software application access do not see the panel.

View Permission

The View Permission section lists the user groups that can see the Development panel in a JIRA Software issue. The Development panel displays the Create Branch link and summary information for your development process, such as the number and status of the related commits, pull requests, reviews and

builds. The visibility of the panel is controlled by the "View Development Tools" project permission.

Applications

The Applications section lists the development tools that are integrated with JIRA Software.

Set up JIRA Software with development tools

Check that you have a compatible version of a development tool on the [version matrix](#), then follow the instructions below to connect your code development tools to JIRA.

Link to BitBucket Server, Bamboo, FishEye or Crucible

JIRA must be connected with Bitbucket Server, Bamboo, FishEye or Crucible [using application links](#).

Note that for Bitbucket Server, the following system plugins are required (these are bundled and enabled by default in Bitbucket Server):

- Atlassian Navigation Links Plugin (com.atlassian.plugins.atlassian-nav-links-plugin)
- Bitbucket Server Dev Summary Plugin (bitbucket-jira-development-integration-plugin).

If your developer tools instances are running on the same machine as JIRA Software Server, you'll need to ensure that the applications uses distinct web contexts. This avoids authentication and session issues with OAuth and application links. For example, if you were running FishEye and JIRA, you would change the default paths to:

<http://localhost:8080/>
<https://localhost:8060/fisheye> (rather than <http://localhost:8060/>)

Instructions:

- [Moving Bitbucket Server to a different context path](#)
- [Changing Bamboo's root context path](#)
- [Linking FishEye to JIRA](#)

Troubleshooting

[JIRA Application Development panel displays error - Couldn't read data](#)

Getting started with Bitbucket and JIRA Cloud

Learn how you can connect the Bitbucket code hosting service with JIRA Cloud. Connecting Bitbucket to JIRA gives your team the power to see commits, branches, pull requests. You can also create branches and see the status of pull requests all from the development panel in JIRA or JIRA Agile.

Development

1 branch

14 commits Latest 22/Aug/16 9:47 AM

1 pull request OPEN Updated 2 hours ago

1 build ✔ Latest 22/Aug/16 10:00 AM

[Create branch](#)

You won't see the builds reference shown in the screenshot above unless you [Integrate JIRA with Bamboo](#).

On this page:

- Before you begin...
- Step 1. Sign up for Bitbucket and create a Bitbucket team
- Create a team
- Step 2. Invite team members to your team on Bitbucket
- Step 3. Create repositories in or move repositories to your Bitbucket account
- Step 4. Connect the team account in JIRA
- What to do next

Before you begin...

1. Bitbucket and JIRA Cloud are independent services: you will have both a JIRA Cloud account and a Bitbucket team each with their own set of users, permissions, and access rules.
2. Bitbucket teams are not accounts: they must be managed by an administrator (or administrators) who have individual Bitbucket accounts. However, the Bitbucket team can have its own payment plan.
3. Every member of the Bitbucket team must have their own individual Bitbucket account. When you invite new team members using their email they are also automatically invited to sign up for Bitbucket and are automatically added to your team when they complete sign up.
4. You can transfer Bitbucket [repository ownership to a team](#): this can be helpful if you want to create a team based upon existing repositories.

Step 1. Sign up for Bitbucket and create a Bitbucket team

If you don't already have JIRA Cloud, [set up](#) a trial or a paid instance.

Create a Bitbucket account

If you already have an account, you can skip this section and go to the [next](#). When you create a Bitbucket individual account you must supply the following fields:

Field	About what you are supplying
Username	Up to 30 character username. You can use letters, numbers, and underscores in your username. Your username must be unique across the entire Bitbucket site. Bitbucket appends this username to the URL for all the repositories you create. For example, the username <code>atlassian_tutorial</code> has a corresponding Bitbucket URL of <code>https://bitbucket.org/atlassian_tutorial</code> .
Email address	An email address that is unique across the entire Bitbucket site. The system sends you a confirmation email.
Password	A combination of up to 128 characters. If you are using a Google account to sign up the system uses that password. You are responsible for ensuring that your account password is sufficiently complex to meet your personal security standards.

To sign up for a Bitbucket account:

1. Open <https://bitbucket.org/account/signup/> in your browser.
2. Complete the fields in the sign up form.
3. Click **Sign up**.

When you are done signing in, Bitbucket places you in the **Dashboard** of your account. Take a second to look around the user interface. Across the side of each Bitbucket page is a series of options that let you navigate around Bitbucket. On the top bar is a link for **Teams**. Select **Teams > Create team** and move to the next

section.

Create a team

To better understand how teams work let's first take a look at how they fit into the Bitbucket environment.

Teams are comprised of	Which are
Users	Who develops the code and manages the team. Each user has an individual Bitbucket account which can be added to (or removed from) any group or team within the Bitbucket universe.
Groups	What users can do and where they can go. Groups provide permissions (administrator, read/write, read only) to groups of individual users, and are assigned to repositories for the team.
Repositories	Where the code lives. The repository is where you store, access, create, develop, modify, and share the code for a project

Create a team

The following process and infographic describe how to create a very simple team consisting of you, one member, and one repository. Feel free to follow along in your Bitbucket account, or just read through to get an idea of what goes into creating a team.

<p>Prerequisite:</p> <p>You have an individual Bitbucket account</p>	
<p>Create a team</p> <p>Two user groups, Administrator and Developer, are created by default when you create a team.</p> <p>To create a team:</p> <ol style="list-style-type: none"> 1. Select Teams > Create team. 2. Fill in the available fields: <ol style="list-style-type: none"> a. Team name (when adding a team to a JIRA Cloud instance, consider using the same name as the JIRA Cloud instance or one you can easily identify with a specific project. b. Team ID 3. Add additional team members by entering their Bitbucket username or email address and clicking Add for each person you want to invite to the new team. <p>Note: you are already a member of the team and the administrator by default.</p> 4. Click Create. <p>Congratulations you have a team! You are taken to the team overview page where you can create your first team repository or manage the team's settings.</p>	

Step 2. Invite team members to your team on Bitbucket

You can add team members to your linked Bitbucket team account. These may be the same users you added to

JIRA. It is your choice. Users that you add to Bitbucket need not have accounts on the JIRA instance.

To add a user to a team, you add the user to one of the team's groups, as follows:

1. Log in to Bitbucket as a user with administrative rights on the team.
2. Choose Your avatar > **View all teams**.
3. Choose the team you want to add new members to.
4. Choose **Settings > User groups**.
5. Click on the group you want to add the user to.
6. Enter the user's username or email address, then click **Add**:

Members
Repositories

Add

	matthewhunter	Matthew Hunter	×
	tw-bot	tw-bot	×

If you entered an email address that corresponds to a Bitbucket account, Bitbucket resolves the account for you. If Bitbucket can't resolve the address, it sends the user an invitation to create a Bitbucket account and join the team.

Step 3. Create repositories in or move repositories to your Bitbucket account

A repository (sometimes called a repo) contains your project code. Whether you have no files or many files, you'll first want to create a repository on Bitbucket Cloud. From there, you can clone your repository to your local system and start working on it.

If you name a repository with upper case letters, you'll see the name with upper case letters in Bitbucket, but Bitbucket converts the name to all lower case in the repository UR. As a result, you can't create two repositories with names that result in the same URL.

To create a repository

1. Click **+** in the global sidebar and select **Repository** under **Create a new**.
2. Choose a repository **Owner**.
You'll only see this option if you're a member of at least one team.
3. Enter a **Repository name** that will describe your repository and appear in its URL.
4. Keep **This is a private repository** selected unless you want to make your repository public so that anyone can see it.
5. If you already have files that that you want to add to your repository, select **No** from **Include a README?** Otherwise, go with the default option or select a one of the included README options.
6. Select the **Version control system**. If you don't know the difference, keep **Git** as the default system.
7. Click **Create repository**.

After you create a repository

What comes next depends on what you want to do with your repository:

- **Starting from scratch with no files?** — Clone the repository to your local system to connect Bitbucket repository to a local directory. [Learn how](#)
- **Working on existing files that aren't under version control?** — Add unversioned files to a repository before pushing them to Bitbucket. [Learn how](#)
- **Already have local files in a Git or Mercurial repository?** — Push versioned code to an empty repository, maintaining commit history. [Learn how](#)

Take a minute to explore what comes with your new repository.

The screenshot shows the Bitbucket web interface for a repository named "BitbucketStationLocations". The interface includes a sidebar with navigation options: Source, Commits, Branches, Pull requests, Pipelines, Downloads, Boards, and Settings. The main content area displays the repository name, an "Invite" button, and a "Clone" button. Below this, there's a section for source files with a table listing "README.md" (2.56 KB, 8 minutes ago). A "README.md" section follows with instructions on how to edit the file and create a new file.

The Bitbucket service allows you to create an unlimited number of public repositories. The number of private repositories is restricted by your plan.

Tips, Tricks, and Links to More Information

- You can transfer a Bitbucket repository from an individual Bitbucket account to your JIRA team account.
- You can [import a Git or Mercurial project from your local system into Bitbucket](#).
- To learn about Bitbucket's few restrictions on repositories, see [this page](#).
- Some users have security and backup concerns about code, see [this page](#) for details.
- See the Atlassian blog for information about [Centralized vs. Distribute Version Control System \(DVCS\)](#).

Step 4. Connect the team account in JIRA

Make sure you understand how JIRA connects to your Bitbucket account

The connector needs permission from your Bitbucket account to access your account data. The connector does this through an [OAuth](#) access token.

You create an OAuth access token from the Bitbucket account. You should create the access token on the team

that owns the repositories that you want to link. The values that make up the token are:

key	A string generated by the Bitbucket system.
secret	A string generated by the Bitbucket system.
authorizing account	The account that authorizes the token.

After you create a key and secret in Bitbucket, go back to JIRA. There, you can enter the account, the OAuth key, and secret data.

Bitbucket does not automatically trust the key and secret it will ask you to authorize the Bitbucket connection.

When you link your Bitbucket account with JIRA all the public and private repositories owned by the account. It adds a POST commit hook service to the repository on the Bitbucket system. The POST commit hook is a piece of code that sits on the repository waiting for users to commit changes.

On the JIRA Cloud side, the repositories owned by your Bitbucket account appear on the **Manage DVCS Accounts** page. A team member may create repositories under their individual Bitbucket account, but assign the team as the owner. These repositories also appear in Bitbucket under the list.

Procedure to link an account

It is a two step procedure to link a Bitbucket account to JIRA. To work through this procedure, you must have administrative rights on both the JIRA Cloud instance and on the Bitbucket account you want to link.

Step 1. Create an OAuth access token for your Bitbucket account

To link a Bitbucket account you create an OAuth access token on your Bitbucket account. If you are linking repositories under a team, you should generate this token under the team account.

Log in to Bitbucket as a user with administrative rights on the account.

1. Choose **Manage account**.
2. (Optional) If connecting a team, choose the team from the **Account** drop-down.
3. Select **OAuth**.
4. Click **Add consumer**.
5. Enter `JIRA DVCS` for the **Name**.
6. Leave the other fields blank.
7. Press **Add consumer**.

Keep your browser open to Bitbucket and go onto the next step.

Step 2. Link the account on JIRA

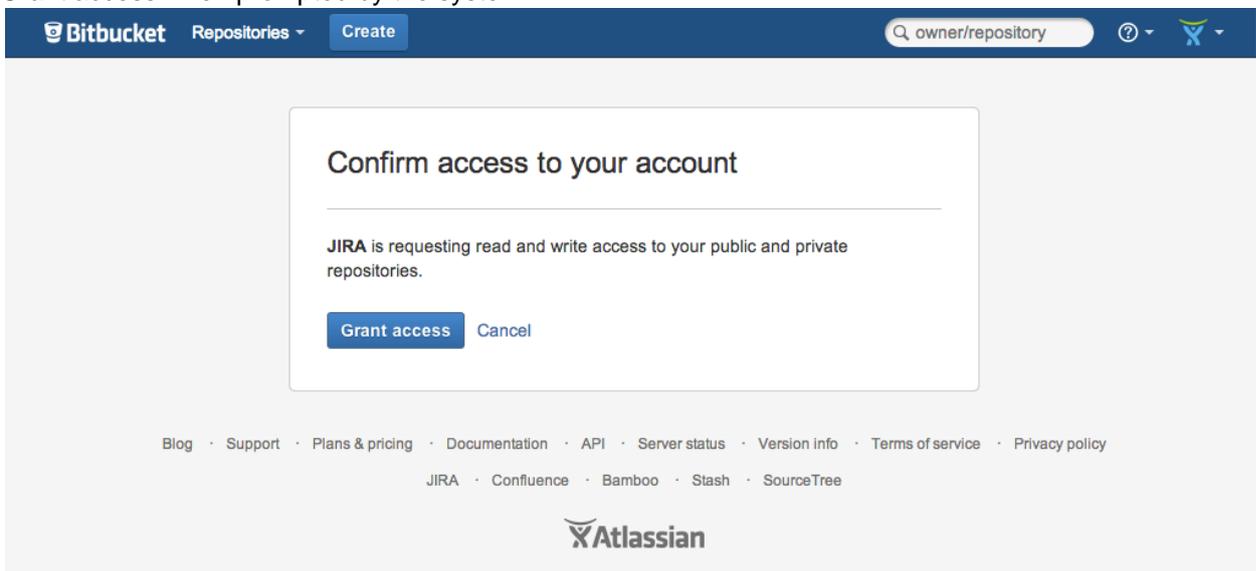
To complete the link between your DVCS and JIRA:

1. Log in to JIRA Cloud as a user with administrative rights.
2. From the JIRA dashboard click the  (settings) icon.
3. Choose **Applications** then **DVCS accounts** (under 'Integrations' in the left-hand panel).
4. Click **Link a Bitbucket Cloud or GitHub account**.
5. Choose **Bitbucket Cloud** as your **Host** value.
6. Enter a **Team or User Account**.

For example, if you want to link the account that owns the <https://bitbucket.org/tutorials/markdowndemo> repository then you would enter `tutorials` for the **Team or User Account** value. Linking the `tutorials` account links all of that account's repositories, not only the `markdowndemo` repository.

7. Copy the **OAuth Key** and **OAuth Secret** from your Bitbucket account into the dialog.
8. Leave the default auto link and smart commits (recommended) as is or change them.

9. Click **Add**.
10. Grant access when prompted by the system:



11. Upon success, the **DVCS accounts** page displays with your account.

The account you just connected and all of its repositories appear in the **DVCS accounts** page. The initial synchronization starts automatically. After that, the system continues to sync your repository automatically on a regular basis.

Automatic synchronization and temporarily disabling a link

After you link an account, JIRA automatically starts looking for commits that reference existing issue keys. The summary shows the synchronization results and errors, if any. A synchronization of commit data from the DVCS repository to JIRA can take some time. As the synchronization progresses, the commits appear in related issues. You can always enable and disable the linking of repositories with JIRA as needed.

How the link appears in Bitbucket

The DVCS Connector does two things:

- It adds an OAuth consumer to the linked account's list of integrated applications. To view the listing in Bitbucket, click your profile image and select **Manage Account**. Click **Integrated applications** and you'll see a listing similar to the following:

- The DVCS Connector programmatically adds a POST commit hook service to each of the account's repositories. To view this service, choose



Settings, then click **Hooks** to display the Hooks page. You'll see a listing similar to the following:

The DVCS Connector uses its link to check for new repositories on the account, then adds this service to those as well. You see the result of all this on the **Services** page.

What to do next

What you do after getting started depends on your team's own knowledge and needs:

- If your team is unfamiliar with code hosting using Bitbucket or brand new to DVCS, [you should work through the Bitbucket 101](#).
- If your team is comfortable with DVCS and Bitbucket, you might want to learn to [Link to a web service in Bitbucket](#) which will give you even more interaction between JIRA and Bitbucket.
- The DVCS Connector lets you update and move JIRA issues through a workflow, see [processing JIRA issues with smart commit messages](#) for commands and examples of how to do this.

Enable Smart Commits

When you manage your project's repositories in Bitbucket or GitHub, or use FishEye to browse and search your repositories, you can process your JIRA Software issues using special commands in your commit messages.

You can:

- comment on issues
- record time tracking information against issues
- transition issues to any status (for example 'Resolved') defined in the JIRA Software project's workflow.

Learn more about using Smart Commits: [Processing issues with Smart Commits](#). There are other commands available if you use Crucible for code reviews. See [Using Smart Commits](#) in the FishEye/Crucible documentation.

On this page:

- [Get Smart Commits working](#)
 - [First, link JIRA Software to the other application](#)
 - [Then enable Smart Commits in JIRA Software](#)
- [Forks and Smart Commits](#)

Get Smart Commits working

There are a couple of things you need to set up to get Smart Commits working.

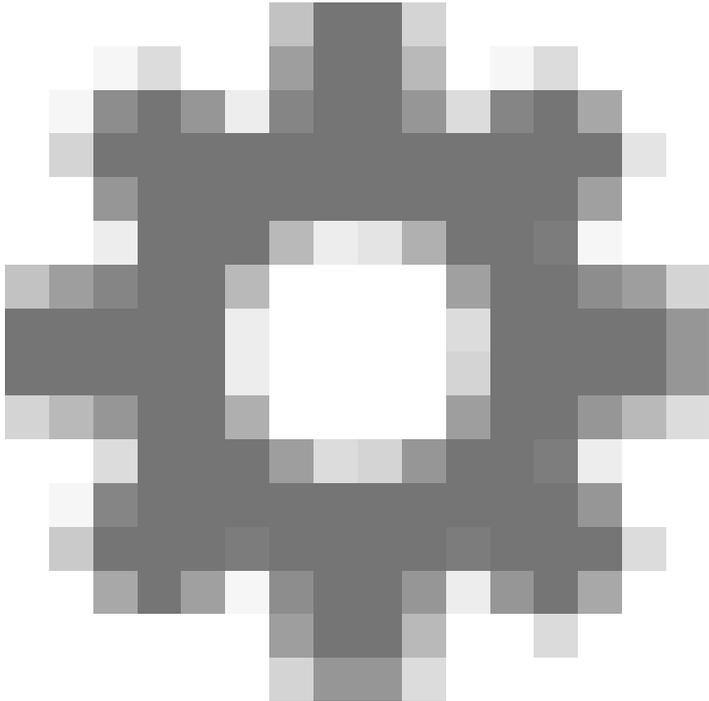
First, link JIRA Software to the other application

Smart Commits relies on *either* the JIRA DVCS Connector Plugin or an application link:

Application	Connect with	Notes
Bitbucket Cloud, GitHub	JIRA DVCS Connector	The JIRA DVCS Connector Plugin is bundled with JIRA Software, but if necessary, a JIRA administrator can install it directly from within the JIRA administration area. See Installing Marketplace apps for more information. A JIRA administrator with access to the Bitbucket Cloud or GitHub account must set up OAuth authentication with JIRA Software.
Bitbucket Server, FishEye, Crucible	Application link	See Linking a Bitbucket or GitHub repository with JIRA .

Then enable Smart Commits in JIRA Software

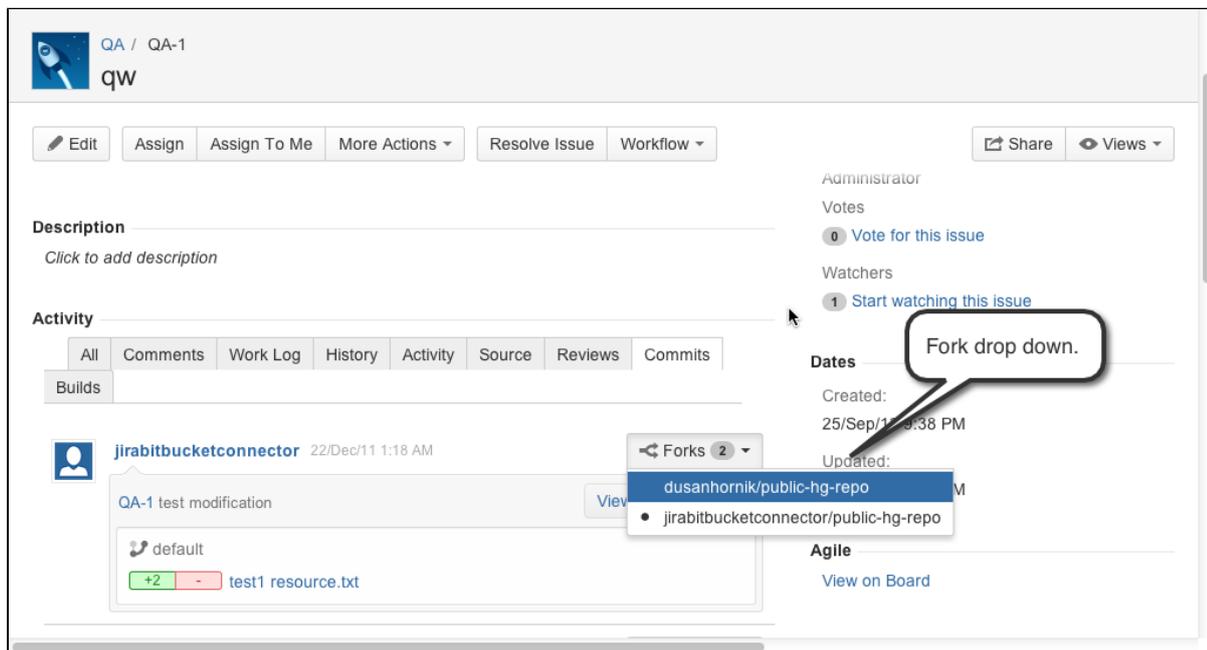
Smart Commits must be enabled in JIRA Software:

Application	Notes
Bitbucket Cloud, GitHub	<p>All new repositories added to your linked Bitbucket Cloud or GitHub account have Smart Commits enabled by default. However, a JIRA administrator can disable that if necessary, and can also enable or disable Smart Commits for individual repositories.</p> <p><i>Control whether Smart Commits are enabled for new repositories:</i></p> <ol style="list-style-type: none"> 1. Log in to JIRA Software as a user with administrative permissions. 2. Go to Administration > Applications > DVCS accounts. 3. Click the  (settings) icon for the account. 4. Click Enable Smart Commits on new repositories. <p><i>Enable or disable Smart Commits on individual repositories:</i></p> <ol style="list-style-type: none"> 1. Log in to JIRA Software as a user with administrative permissions. 2. Go to Administration > Applications > DVCS accounts. 3. Check (or clear) the Smart Commits option for a repository.
Bitbucket Server, FishEye, Crucible	<p>A JIRA administrator can control Smart Commits for each account in the connected application (Bitbucket Server, FishEye or Crucible).</p> <p><i>Enable or disable Smart Commits on individual accounts:</i></p> <ol style="list-style-type: none"> 1. Log in to JIRA as a user with the JIRA Administrator permissions. 2. Choose  > Applications. Select Application Links in the left menu. 3. Click Smart Commits for the application. 4. Select the checkbox for the account you want to enable Smart Commits for.

Note that elevated access rights in JIRA applications can result from the way that Git (and Mercurial) allow commits to be attributed to a user other than the user pushing a change to the repository. If this seems like a risk for your situation, then you should consider disabling Smart Commits in your JIRA application instance.

Forks and Smart Commits

If you use forks in your workflow, the DVCS Connector records each repository that contains a Smart Commit message. It actually processes the Smart Commit message only the first time it encounters it. When you view the commit tab in JIRA Software, you can see which forks include that particular commit:



Synchronize an account

When you have Jira Cloud [connected](#) to your Bitbucket Cloud accounts, Jira automatically updates as you work, so your team can see new branch, commit and pull request activity in Jira issues.

However, sometimes you may need to manually refresh Jira if you see inconsistencies between the information in Jira Software and the actual activity in your repository.

Synchronization can also fail if Jira does not recognize your OAuth settings. When this happens, you'll see an error message on the Jira 'DVCS accounts' page.

Manually synchronizing

The syncing operation compares the existing data in Jira Software with that in the linked repository. A sync does not affect the commits in Bitbucket. You can run a *soft sync* or a *full sync*:

- A soft sync retrieves changes made since the last sync. This is the recommended option.
- A full sync clears all existing data in Jira and resyncs everything. Only do this if your data or synchronization is broken. Note that a full sync can severely increase traffic on the repository provider, who may choose to temporarily restrict your account as a result.

You might consider manually syncing when:

- There are missing commit activities in Jira Software, for example if Jira Software was offline and there were commits to the repository.
- You haven't seen commits appearing in Jira Software for a longer period of time than usual.
- Bitbucket Cloud is back online after being unavailable and you want to retrieve the commit data immediately. Otherwise, you can wait for Bitbucket to perform the next sync.

Always try a soft sync first; only do full sync as a last resort.

Soft syncing

To do a soft sync:

1. Log in to Jira Software using an account with Jira admin or sysadmin permissions.
2. Go to **Applications > DVCS accounts**.
3. Locate the repository that you want to sync.

4. Click the sync icon beside the **Last activity** date (you need to hover over the row to see that):

Repository	Last activity
Brit2Am spelling	Fri Feb 19 2016 

Full syncing

To do a full sync:

1. Locate the repository that you want to sync on the 'DVCS accounts' page.
2. Hold down SHIFT and click the sync icon beside the **Last activity** date (you need to hover over the row to see that).
3. Choose **Full synchronization**.

Configure automatic team invitations

You can add users to a Bitbucket Cloud group through either Bitbucket or through Jira Software. These groups are defined on a Bitbucket account (team or individual). When you add users, they receive an email invitation to join Bitbucket and the account group. To join a group, users must already have their own individual Bitbucket accounts. Each invitation contains a link that takes a user to Bitbucket, where they finish joining by providing their account (or by creating an account if necessary).

When you add users through Bitbucket, you can supply a user's Bitbucket account name or their email. If you are using Jira Software Cloud or Jira Software Server, you have options for adding Jira Software users manually or automatically. Automatic user sign up can happen through a Google integration (Cloud) or by public signup (Jira Software Server). These Jira Software users receive invitations only from the Bitbucket accounts you've configured to send them.

This page explains how to customize the Jira Software invitation behavior.

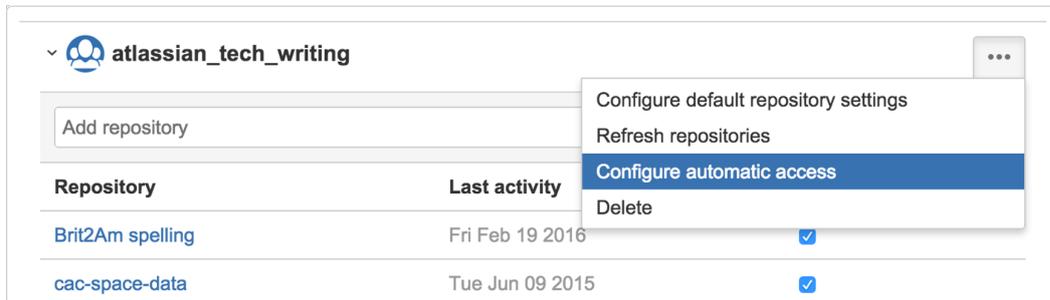
Decide on a configuration strategy for adding users to account groups

Before you configure automatic user sign up, decide which strategy for adding users to Bitbucket Cloud works best for your situation.

Which description best matches your situation?	The configuration we recommend you should use:
All of my Jira Software users should belong to one or all of my Bitbucket Cloud/group/team combinations.	Automatically invite new Jira Software users to a Bitbucket Cloud account's group when they become Jira Software users. See the following section on this page.
I add my users manually to my Jira Software instance.	Automatically invite new Jira Software users to a Bitbucket account's group. Optionally, change the Bitbucket team/groups for the user when you add them to Jira Software. See the following section on this page.
I've enabled automatic sign up on Jira Software but only a small set of Jira Software users should belong to my Bitbucket team.	Prevent the automatic invitation of new users. Manually add users to groups through Bitbucket. See the following section on this page.
I've enabled automatic sign up on Jira Software but some Jira Software users should belong to one Bitbucket team/group combination and others users belong to another.	

Set the groups that users are automatically invited to join

1. Log into Jira as a user with administrative rights.
2. Go to **Applications > DVCS accounts** in the Jira admin area.
3. Locate the account to configure. If you don't see your account, click **Link Bitbucket Cloud account** to start the connection process.
4. Choose **Configure automatic access** from the Actions menu:



5. Select the groups you want new Jira users assigned to, in the 'Configure automatic access' dialog. New Jira Software users will automatically be added to the groups you select, and will be invited to join Bitbucket. When the user joins, they have the group access to your project. To prevent users from being invited to join groups, deselect those groups.
6. Press **Save**.

Linking a Bitbucket or GitHub repository with JIRA

Use the JIRA DVCS connector to link a Bitbucket Cloud or GitHub (hosted or enterprise) account to JIRA Software Server. When linked to JIRA Software, branches, commit messages and pull requests are all seamlessly referenced in JIRA Software issues. This allows JIRA Software to display information about your development activity in the corresponding issue. See [Streamlining your development with JIRA](#).

This page explains how to link a Bitbucket or GitHub account to JIRA.

- [Make sure you understand how JIRA Software connects to your DVCS account](#)
- [Link a DVCS account to JIRA Software Server](#)
- [Example of how commit information appears in a JIRA Software project](#)

Make sure you understand how JIRA Software connects to your DVCS account

The JIRA DVCS connector links JIRA Software Server to an account on a DVCS hosting service (Bitbucket Cloud, GitHub, or GitHub Enterprise). For this reason, the connector needs permission from your DVCS account to access your account data. The connector does this through an [OAuth](#) access token.

You create an OAuth access token in the DVCS (Bitbucket, GitHub, or GitHub Enterprise). You should create the access token in the account that owns the repositories you want to link. How you create the token depends on the DVCS; the values that make up the token are:

key	A string generated by the DVCS.
secret	A string generated by the DVCS.
authorizing account	The account that authorizes the token.

After you create a key and secret in the DVCS, you go back to JIRA Software. There, you enter the account, the OAuth key, and secret data into JIRA Software.

The connector does not automatically trust the key and secret. It asks you to authorize the DVCS connection via the key and secret by supplying JIRA Software an account and password combination. Authorizing is the equivalent of telling the DVCS connector:

As a Bitbucket account holder, I know this service asking for a connection with a key and secret.

You are free to use them to get to this account data.

The *authorizing account* is not necessarily the account that created the key and secret. The authorizing account *should* have administrative access on all the repositories to be linked.

When you link an account through the JIRA DVCS connector, the connector locates all the public and private repositories owned by the account. It adds a post-commit service to the repository on the DVCS. The post-commit service is a piece of code that sits on the repository waiting for users to commit changes. When a commit happens, the DVCS connector collects the commit message for processing.

On the JIRA Software side, the repositories owned by your DVCS account appear on the **DVCS accounts** page. A team member may create repositories under their individual Bitbucket account, but assign the team as the owner. These repositories also appear in Bitbucket under the list.

Link a DVCS account to JIRA Software Server

When you link a Bitbucket or GitHub account to JIRA Software Server, you create an OAuth access token in the DVCS tool and then add that token to JIRA Software.

You'll need to have administrative rights on both the JIRA Software instance and on the DVCS account you want to link.

Step 1. Create an OAuth access token for your DVCS account

You create the OAuth access token on your DVCS account. If you are linking repositories for a team, you should generate this token using the team account.

Generate the new OAuth token in either Bitbucket, GitHub, or GitHub Enterprise, depending on which DVCS hosts your repositories.

Generate a new token in Bitbucket

Log in as a user with administrative rights on the account.

1. Choose **avatar > Bitbucket settings**.
2. (Optional) If connecting a team, choose the team from the **Manage** dropdown.
3. Click **OAuth** under 'Access Management'.
4. Click **Add consumer**.
5. Enter the following details:

Name	Enter 'JIRA DVCS' for this example.
Description	Enter a helpful reminder of the purpose of this token.
URL	Enter the URL for the JIRA Software instance (for example, https://example.atlassian.net)

6. Select the following permissions:
 - **Account: Write**
 - **Repositories: Admin** (but not Repository: Write)
 - **Pull requests: Read**

Permissions	
Account	<input checked="" type="checkbox"/> Email <input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write
Team membership	<input type="checkbox"/> Read <input type="checkbox"/> Write
Repositories	<input checked="" type="checkbox"/> Read <input type="checkbox"/> Write <input checked="" type="checkbox"/> Admin
Pull requests	<input checked="" type="checkbox"/> Read <input type="checkbox"/> Write
Issues	<input type="checkbox"/> Read <input type="checkbox"/> Write
Wikis	<input type="checkbox"/> Read and write
Snippets	<input type="checkbox"/> Read <input type="checkbox"/> Write
Webhooks	<input type="checkbox"/> Read and write

These are the minimum permissions required by the JIRA DVCS connector.

Selecting additional permissions will have no adverse affects on the integration.

- Click **Save**.
- Click the name of your new consumer to see the OAuth **Key** and **Secret** values.
- Keep your browser open to your DVCS and go to the next step.

Generate a new token in GitHub or GitHub Enterprise

Log in as a user with administrative rights on the account:

- Choose **Edit Your Profile**.
- Select **Applications**.
- Choose **Register new application**.
- Enter `JIRA DVCS` for the **Application Name**.
- Enter the JIRA Software URL for both the **URL** and **Callback URL** fields. Press **Register Application**.

Make sure you enter the **JIRA Software Base URL** (for example, <https://example.atlassian.net>) for *both* the **Homepage URL** and **Authorization callback URL** fields. *Don't* use the dashboard URL (<https://example.atlassian.net/secure/Dashboard.jspa>).

See

JSWSERVER-14228 - Entering mismatched URL for GIT causes error **CLOSED**

For JIRA 6.2, the URL to use is `https://example.atlassian.net/plugins/servlet/oauth/authorize`.

- Keep your browser open to your DVCS and go to the next step.

Step 2. Link the account on JIRA Software

Do the following to complete the link between your DVCS and JIRA Software:

- Log in to JIRA Software as a user with administrative rights.
- From the JIRA Software dashboard click the  (settings) icon.
- Choose **Applications**.
- From the **Integrations** section on the left, choose **DVCS accounts**.
- Click **Link Bitbucket Cloud or GitHub account**.
- Choose **Bitbucket Cloud** as your **Host** value.

7. Enter a **Team or User Account** name.

For example, if you want to link the account that owns the <https://bitbucket.org/tutorials/markdowndemo> repository, you would enter `tutorials` for the **Team or User Account** value. Linking the `tutorials` account links all of that account's repositories, not only the `markdowndemo` repository.

8. Copy the OAuth **Key** and **Secret** values from your DVCS site into the dialog.

GitHub's **Client ID** is equivalent to the OAuth **Key**. And the **Client Secret** is equivalent to the OAuth **Secret**.

9. Leave the default auto link and smart commits (recommended) as is or change them.

10. Click **Add**.

If you are connecting a GitHub account and get redirected to a blank page, see [DVCS connection to GitHub produces blank page](#).

11. Grant access when prompted:

The account you just connected and all of its repositories appears in the 'Managed DVCS Accounts' page. The initial synchronization starts automatically.

Automatic synchronization and temporarily disabling a link

After you link an account, JIRA Software automatically starts looking for commits that reference issue keys. The summary shows the synchronization results and errors, if any. A synchronization of commit data from the DVCS repository to JIRA Software can take some time. As the synchronization progresses, the commits appear in related issues. You can always enable and disable the linking of repositories with JIRA Software as needed.

How the link appears in Bitbucket

The DVCS Connector does two things:

- It adds an OAuth consumer to the linked account's list of integrated applications. To view the listing in Bitbucket, click your profile image and select **Bitbucket settings**. Click **OAuth** in the 'Access Management' section and you'll see a listing similar to the following:

Manage

Atlassian Technical Writing

GENERAL

[Team settings](#)

[Email addresses](#)

[Change team ID](#)

[Delete team](#)

PLANS AND BILLING

[Plan details](#)

[Billing details](#)

ACCESS MANAGEMENT

[User groups](#)

OAuth

[API key](#)

OAuth integrated applications

You have authorized the following applications to interact with your Bitbucket repositories.

Consumer	Description	Last updated

OAuth consumers

Generate your own OAuth consumer key and secret to [build your own custom integration with Bitbucket](#).

Add consumer

Name	Description
You have no consumers configured.	

- The DVCS Connector programmatically adds a post-commit service to each of the account's repositories. To view this service, choose



(Settings) on a repository, then click **Services**. You'll see a listing similar to the following:

Settings

GENERAL

[Repository details](#)

[Access management](#)

[Branch management](#)

[Username aliases](#)

[Deployment keys](#)

[Transfer repository](#)

[Delete repository](#)

INTEGRATIONS

[Services](#)

Webhooks

[Links](#)

Webhooks

Webhooks allow you to extend what Bitbucket does when the repository changes (for example, new code is pushed or a pull request is merged).

To learn more about how webhooks work, check out the [documentation](#).

Add webhook

Title	URL	Actions
No hooks		

The DVCS Connector uses its link to check for new repositories on the account, then adds this service to those as well. You see the result of all this on the 'Services' page.

Example of how commit information appears in a JIRA Software project

When a developer makes a commit, they should add a JIRA Software issue key to the commit message, like this:

```
hg commit -m "DVCS-2 add a README file to the project."
hg push
```

JIRA Software uses the issue key to associate the commit with an issue, and so the commit can be summarized in the Development panel for the JIRA Software issue:

The screenshot shows the JIRA Agile interface. At the top, there are navigation tabs: 'Agile', 'Capture', and 'Create Issue'. A search bar labeled 'Quick Search' is on the right. Below the navigation, there are tabs for 'My Issues' and 'Recently Updated'. The main content area is divided into two panels. The left panel shows a list of issues, with one issue highlighted: 'FUSE-113' with the description 'Populate builds detailed dialog'. The right panel is titled 'Teams in Space / FUSE-113' and contains a 'Development' section. This section lists several metrics: '8 branches' (Updated 13/Feb/14 5:49 PM), '62 commits' (Latest a day ago), '4 pull requests' (MERGED, Updated a day ago), and '1 build' (Latest 13/Feb/14 6:29 PM). Below these metrics, there is a 'Deployed to Production' button and a 'Create branch' link.

See "Integrating with Development tools" for more information.

Project permissions required

Project users must have the 'View Development Tools' permission to see commit information in the Development panel in a JIRA Software issue. A JIRA Software admin can edit a project's permission schema to grant this permission. See [Managing project permissions](#).

Integrating with other tools

Integrating with Flowdock

You can integrate [Flowdock](#) with JIRA Cloud and issues from your JIRA projects will be included in your Flowdock flows.

If you link a JIRA project to a Flowdock flow, **all JIRA comments will appear on FlowDock** regardless of the restriction level that is set when creating the comment. Please ensure that you only link JIRA projects to Flowdock flows when it is acceptable for all JIRA comments to be visible.

To enable Flowdock in JIRA:

1. Log in as an admin to your site.
2. Choose **Manage Plugins > Show System Plugins**.
3. Locate **Flowdock for JIRA** and click **Configure**. The Flowdock integration page will display all the JIRA projects that are set up.
4. Enter your Flowdock API key against the JIRA projects that you want to include in your Flowdock flow. To get your Flowdock API key, log in to Flowdock and view the [Integrating with variety of issue trackers](#) page. Your API key will be displayed in the JIRA instructions.
5. Click **Save**. The API key information will be saved and the Flowdock integration page will refresh.

You will now receive messages in your Flowdock flow for any issue activity (e.g. issue creation, issue comments added, issue fields updated, etc) in the configured JIRA projects.

Integrating with Zephyr

JIRA Cloud comes with the Zephyr Enterprise Connector plugin, and this plugin sends defect metrics to Zephyr Enterprise and Zephyr Community Editions. This plugin is a different plugin from [Zephyr for JIRA](#).

To enable Zephyr in JIRA:

1. Log in as an admin to your site.
2. Choose **Plugins**. You will see the list of user-installed plugins.
3. Near the bottom of the page, locate the **Zephyr Enterprise Connector** and click it to display the available options.
4. Click **Enable**. The Zephyr plugin will be enabled.

To connect to JIRA from Zephyr:

See Zephyr's [JIRA Overview & Setup](#) documentation.

Related topics

- http://www.getzephyr.com/zephyr/test_management_integrated_with_atlassian_ondemand.php
- <https://plugins.atlassian.com/plugin/details/18715>

Integrating with Subversion

JIRA's Subversion integration lets you see Subversion commit information relevant to each issue. Subversion integration can be implemented by using [Atlassian FishEye](#). The FishEye integration offers [greater scalability, insight and flexibility](#) into your source code and related integration with JIRA, however both solutions allow you to link JIRA to related code changes in Subversion.

Revision	Date	User	Message
#11	Fri Nov 12 17:30:39 EST 2004	Mike	Big, very exciting commit for TEST-3! Files Changed ADD trunk/moved.txt (from trunk/copieddocument.txt #10) DEL trunk/copieddocument.txt ADD trunk/NewFile.java DEL trunk/mydocument.txt
#10	Thu Nov 11 18:12:59 EST 2004	Mike	Fixed TEST-3 Files Changed MODIFY trunk/mydocument.txt

Commits will appear in this tab if the commit log mentions the issue key ('TEST-3' above).

Listeners

Listeners are unique to JIRA, and a very powerful way to extend it.

JIRA has a complete event subsystem that 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.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- [Listener interfaces](#)
- [Example listeners](#)
- [Registering a listener](#)
- [Editing listener properties](#)
- [Removing a listener](#)
- [Custom events](#)
- [See also](#)

Listener interfaces

JIRA has the following concrete Listeners (which extend the base `JiraListener` interface):

<code>com.atlassian.jira.event.JiraListener</code>	The base interface which all other JIRA listener interfaces extend. Covers core listener properties like uniqueness, description, parameters etc. API doc
<code>com.atlassian.jira.event.issue.IssueEventListener</code>	The main listener interface in JIRA, used whenever anything happens to an issue. API doc
<code>com.atlassian.jira.event.user.UserEventListener</code>	This listener is called whenever anything happens to a user within JIRA. API doc

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** — 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** — 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 (e.g. 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.

Registering a listener

i For custom-written listener classes, make sure your listener class is in the classpath where JIRA can see it — the best locations are usually the `<jira-application-dir>/WEB-INF/classes` or `<jira-appli`

cation-dir>/WEB-INF/lib subdirectories of your [JIRA installation directory](#) (as JAR files).

1. Choose



> **System.**

2. Select **Advanced > Listeners** to open the Listeners page.
3. In the 'Add Listener' form at the bottom of the page, complete the following fields:
 - 'Name' — an appropriately descriptive name for the listener.
 - 'Class' — the fully-qualified class name of your listener.

Add Listener

Add a new listener by entering a name and class below. You can then edit it to set properties.

Name

Class

[> Built-in Listeners](#)

i To use one of JIRA's built-in listener classes, first click the '**Built-in Listeners**' link to expand the list of listener classes and then click the name of the specific class in the list. The fully-qualified class name of the built-in listener will be added to the 'Class' field.

4. Click the '**Add**' button and the listener will now be added to the list of listeners above.

Editing listener properties

If your listener accepts parameters or properties, you can edit these by clicking the '**Edit**' link associated with your listener (on the 'Listeners' page in JIRA's Administration area).

When defining your own Listener, there is a method `getAcceptedParams` to overload for defining the parameter names which are passed as an array of String objects. The `init` method is given a Map with the configured values (the JavaDoc is outdated). The `com.atlassian.jira.event.listeners.DebugParamListener` class is a good example of doing this with two parameters.

Removing a listener

To remove a listener, click the '**Delete**' link associated with that listener (on the 'Listeners' page in JIRA's Administration area).

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.

See also

- [Tutorial - Writing JIRA event listeners with the atlassian-event library](#) — this describes how to write listeners using the Atlassian Events library (see [JIRA-specific Atlassian Events](#)), rather than the JIRA Listener Events described above.

Managing webhooks

Webhooks are user-defined HTTP POST callbacks. They provide a lightweight mechanism for letting remote applications receive push notifications from JIRA, without requiring polling. For example, you may want any changes in JIRA bugs to be pushed to a test management system, so that they can be retested.

i Please read the [JIRA Webhooks](#) page (*JIRA developer documentation*)

On this page:

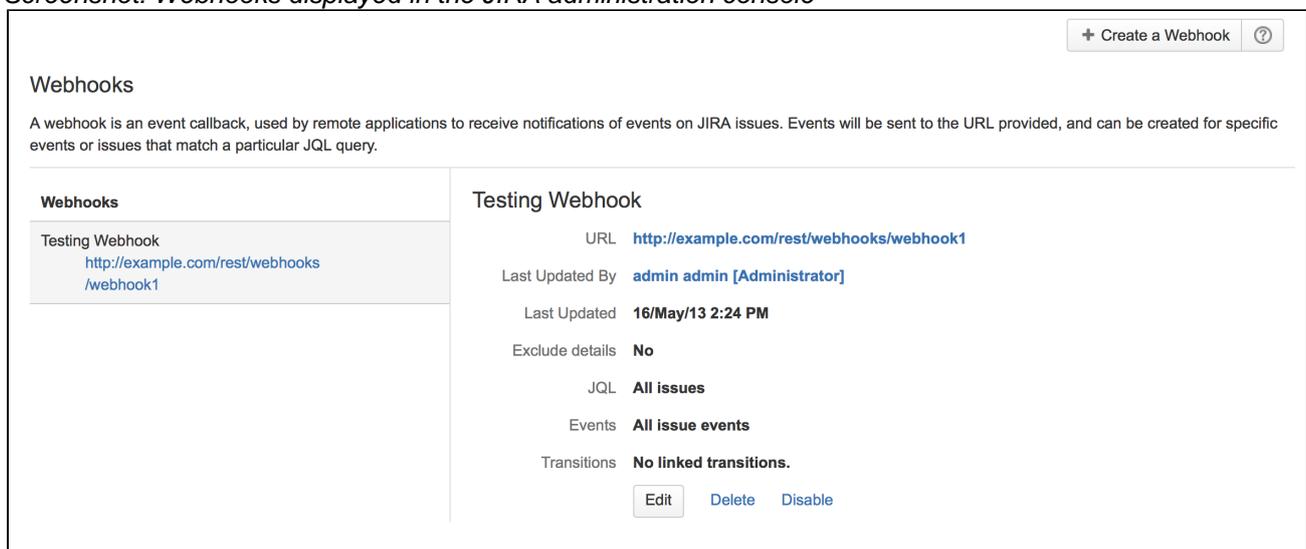
- [Managing webhooks in JIRA](#)

for detailed information on how to configure JIRA webhooks, including a description of the events, how to register a webhook via the REST API, examples, and more. This page only contains instructions on how to use the Webhooks user interface in the JIRA administration console.

Managing webhooks in JIRA

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose 
 - > **System**. Select **Advanced > Webhooks** to open the Webhooks page, which shows a list of all existing webhooks.
3. Here's a few tips on using this page:
 - Click the summary of the webhook in the left 'Webhooks' column to display the details of the webhook. You can edit, delete and disable it via the details panel.
 - Deleting a webhook removes it permanently. If you just want to prevent it from firing, disable the webhook instead.

Screenshot: Webhooks displayed in the JIRA administration console



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.

Writing a new service?

If you are not extending a built-in JIRA service, you should strongly consider writing your new service using the SAL API. Please see our [Scheduling Events via SAL Tutorial](#) for more information.

 **Note:** For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Registering a service

 For custom-written services, make sure your service class is in the classpath where JIRA can see it — the best locations are usually the `<jira-application-dir>/WEB-INF/classes` or `<jira-application-dir>/WEB-INF/lib` subdirectories within of your [JIRA application installation directory](#) (as JAR files).

1. Choose

On this page:

- [Registering a service](#)
- [Editing service properties](#)
- [Removing a service](#)
- [Built-in services](#)
- [Custom services](#)



> **System.**

2. Select **Advanced > Services** to open a page showing all the configured services.
3. In the **Add Service** form at the bottom of the page, complete the following fields:
 - **Name** — a descriptive name for this service.
 - **Class** — the fully-qualified class name of your service. This is likely to have the form `com.atlassian.jira.service.services.type.TypeService`
See [Sample services](#) for provided service class names.
 - **Delay** — the delay (in minutes) between service runs.
For example, to add a debugging service, click the **Built-in Services** link followed by the **Debugging Service** link.
4. After completing the fields on the **Add Service** form, click the **Add Service** button. This opens the **Edit Service** page, where you can configure your new service's options.
 - **Info** Your service's options will vary depending on the type (i.e. class) of service you chose.
5. After completing the remaining options on the **Edit Service** page, click the **Update** button to save your new service's options.

Editing service properties

1. Choose
 - > **System.**
2. Select **Advanced > Services** to open a page showing all the configured services.
3. Click the **Edit** link associated with the service whose properties you wish to edit.

For example, to change the interval at which email is sent from JIRA, edit the **Mail Queue Service** and change the **Delay** from the default value of 1 minute.

Removing a service

1. Choose
 - > **System.**
2. Select **Advanced > Services** to open a page showing all the configured services.
3. Click the **Delete** link associated with the service you wish to remove.

Built-in services

JIRA has some useful services out of the box, which may be used as-is or modified for use in your own environment. The source code for all built-in services is available and should give you a 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

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 in new JIRA installations (once the setup wizard has been completed). Hence, you may find you already have one.

You can find this class within the following directory of an expanded JIRA source archive (which can be downloaded by JIRA customers from <https://my.atlassian.com>):

```
<source-installation-directory>/jira-project/jira-components/jira-core/src/main/java/com/atlassian/jira/service/services/export
Mail handler services
```

JIRA mail handler services are not configurable through JIRA's **Services** page (with the exception of being able to be removed). For more information about configuring a mail handler in JIRA, including the creation of

custom mail handlers, please refer to [Creating issues and comments from email](#).

Custom services

If you are a JIRA developer who wishes to write your own JIRA service, please note that JIRA Service classes must all extend `com.atlassian.jira.service.JiraService`. Most do so by extending `com.atlassian.jira.service.AbstractService` or some more specialized subclass.

Using AppLinks to link to other applications

Application Links (sometimes called "app links") is a bundled app that allows you to set up links, share information, and provide access to certain resources or functionality across multiple Atlassian products. We recommend using OAuth authentication for application links because of the greater security inherent with that protocol. We no longer recommend the Trusted Applications and Basic authentication types.

Linking JIRA to other applications allows you to include information from these systems in JIRA projects and issues. For example, if you link JIRA to Confluence, you can include pointers to wiki pages when creating or editing issues. Another common use case is to link Bitbucket Server with JIRA; this allows you to view branches, commits and pull requests that correspond to your stories in JIRA. In addition to Atlassian applications, you can also link to external applications; for example, you might use a plugin that allows you to share ZenDesk or Salesforce data via an application link.

Create an application link

1. Log in to JIRA as a user with 'JIRA Administrator' permissions.
2. Choose  **> Applications**. Select **Application Links** in the left menu.
3. Enter the URL of the application you want to link to, then click **Create new link**.
 - If you check **The servers have the same set of users...** then this link will be configured using OAuth (with impersonation) authentication.
 - If you are *not* an admin on both servers you won't be able to set up a 2-way (reciprocal) application link. If you want to go ahead and create a 1-way link anyway, clear the **I am an administrator on both instances** checkbox.
4. Use the wizard to finish configuring the link. If the application you are linking to does not have the Application Links plugin, you must supply additional information to set up a link with OAuth authentication.

When you complete the wizard, the Application Links plugin will create the link between your applications using the most secure authentication method that is supported between the two applications. See the [Application Links User Guide](#) for more information.

The new link will appear on the "Configure Application Links" page, where you can:

- Edit the settings of the application link (for example, to [change the authentication type](#) of the link) using the **Edit** icon.
- Specify the default instance if you have multiple links to the same type of application (for example, to multiple Jira servers) using the **Make Primary** link. See [Making a primary link for links to the same application type](#) for more information.

Impersonating and non-impersonating authentication types

OAuth authentication

OAuth authentication redirects a user to log in to the remote application, after which tokens generated on their behalf are used to authorize requests made from the local application. The remote application handling the request uses the access permissions of the account with which the user logged in on that remote application.

Typical scenarios include:

- You are setting up an application link between two applications that do not share the same set of users.
- You want to continue using a link to an application that now allows public sign-on and the link was previously configured with a shared userbase. You can update your application link by changing **OAuth (i**

mpersonation) to **OAuth** when editing the application link.

See [OAuth security for application links](#) for more information.

OAuth with impersonation

Atlassian OAuth with impersonation makes it easy for your users to benefit from the deep integrations between Atlassian applications:

- they're automatically authenticated on the other application and don't get asked to authorize requests.
- they'll only see the information that they have permission to see.

Impersonating authentication makes requests on behalf of the user who is currently logged in.

Note that Atlassian OAuth with impersonation can only be used for application links between Atlassian applications. Furthermore, it should only be used when the two applications share the same userbase, typically managed with an external directory using LDAP.

A typical scenario is:

- You've set up an application link but your users still have to authenticate regularly. This can occur when the application link has been configured to not share the same userbase. If those applications do share the same userbase, you can update your application link by selecting **OAuth (impersonation)** when editing the application link.

See [OAuth security for application links](#) for more information.

Linking to developer tools

When you create a new application link between JIRA and an instance of Bitbucket Server, FishEye, Crucible or Bamboo, 2-legged (2LO) and 3-legged OAuth (3LO) are enabled by default. 2LO is required for information from any of those applications to be included in the summaries in the Development panel; 3LO is used to ensure that a user has authenticated with the other applications before they get to see the information in any of the details dialogs.

An existing application link between JIRA and Bitbucket Server, FishEye, Crucible or Bamboo (that perhaps used Trusted Apps authentication) needs to have 2-legged authentication (2LO) enabled for both outgoing and incoming authentication, so that information from the application can be included in the Development panel summaries.

▼ [Click here to see how to enable 2-legged OAuth...](#)

When updating an older application link to use OAuth, 3-legged authentication is applied by default, but you need to explicitly enable 2LO. Enable 2-legged authentication for the application link from within JIRA as follows:

1. Go to the JIRA admin area and click **Applications**.
2. Click **Edit** for the app link with the other application.
3. For both **Outgoing Authentication** and **Incoming Authentication**:
 - a. Click **OAuth**
 - b. Check **Allow 2-legged OAuth**.
 - c. Click **Update**.

The application link update process will involve logging you into the other application for a short time to configure that end of the link, before returning you to JIRA.

Troubleshooting

Having trouble integrating your Atlassian products with application links?

We've developed a [guide to troubleshooting application links](#), to help you out. Take a look at it if you need a hand getting around any errors or roadblocks with setting up application links.

Administering projects and links across multiple applications

The following pages introduce the various aspects of project administration in JIRA, Confluence, and Bamboo Cloud. You can also learn more about creating links between Atlassian Server and Cloud applications.

Configuring a cross-application link

In order to create links in JIRA to point to other applications, the fields where links are created must use the *Wiki Style Renderer*, e.g. the *Comment* and *Description* fields of your project's JIRA issues.

On this page:

- [Configuring a cross-application link](#)
- [Creating a cross-application project](#)
- [Deleting a cross-application project](#)
- [Linking to server applications from Atlassian Cloud](#)

Procedure

1. Choose



> **Projects.**

2. Select the project that contains the fields you need to configure.
3. On the JIRA project configuration page, go to the **Fields** section.
4. On each field where you will create links, click the Renders link.
5. From the Active Renderer drop-down, select **Wiki Style Renderer**.

If the field configuration applies to multiple projects and you don't want other projects to be affected by the change, copy the referenced **Field Configuration Scheme** and associate your project with the new field configuration before changing the settings.

Creating a cross-application project

If you have multiple Cloud applications, JIRA projects can be associated with objects in the other applications, such as:

- a wiki space
- a build project
- a source repository

The association among these objects makes it possible to automatically link issues, wiki documents, plan, and build result.

Before you begin

Ensure that you at least have access to create projects in JIRA. If you have additional access to other applications in your site (such as Confluence and Bamboo), you can automatically create a corresponding Confluence space or a Bamboo project when your JIRA project is created. For information on how to set application access, see the documentation on [giving users access to JIRA applications](#).

Creating a project

After you complete the setup wizard, a JIRA project is created. If you've chosen to have your site create corresponding objects in other applications, those are created, too. Alternatively, you can manually create objects in other applications and link them together yourself.

After the project is created, you will see the global settings for the project and can continue to configure application-specific settings for the project as needed.

Notes:

- Choosing **JIRA Classic** or **Project Management** creates the *default JIRA project*.
- The *project key* will be used as the prefix of this project's issue keys (e.g. *TEST-100*). Choose one that is descriptive and easy to type.
- The *project lead* is a unique project role. Choose the person who manages the project as the project lead. If there is only one user in your JIRA system, the Project Lead defaults to that person and this field is not available.

Deleting a cross-application project

Deleting a project will delete the project from JIRA only. Any wiki documents, source files, changesets, code reviews, build plans, and build results associated with the project will remain in their respective applications. To delete projects, you must have project admin access for the project in JIRA.

1. Select **Projects > View all projects**. You will see all your projects on the page.
2. Locate the project you want to delete. Click the **Administer Project** button.
3. From the **Actions** button, click **Delete Project**.

The project will be deleted, but corresponding spaces, changesets, etc. in other applications will remain. You must delete them from those application if you want to remove them.

Linking to server applications from Atlassian Cloud

You can set up application links between your cloud and server applications. For more information, see [Link to server products from Cloud](#).

Integrating with collaboration tools**Integrating with Confluence**

Give your team the ability to share, discuss and work with JIRA application issues in Confluence, and create knowledge articles for your service desk customers. Here are some of the ways you can benefit from integrating Confluence with your JIRA applications:

For...	You can...
Bugs	Create a knowledge base article to document a workaround for a bug.
New Features	Create a product requirements document for a new feature.
Self-service	Create knowledge articles that customers can view on the customer portal to find solutions themselves
General JIRA Use Case	Document and collaborate with your team on an issue in Confluence.

And here are just a few of the things Confluence allows you to do:

- Share pages
- Watch pages
- Create knowledge articles from service desk issues
- Collaborative commenting, especially through the use of @mentions
- Form a team network and let them know what you are doing via a status update
- Add images, picture galleries, videos, and more
- Enable various content macros

See [Integrating Jira and Confluence](#) for more information.

Integrating with HipChat

This page also assumes that you are using the latest version of the integration plugin. Some features may not be available with previous versions of the plugin.

You can connect multiple instances of JIRA to the same HipChat instance, however you can't connect

multiple HipChat instances to the same JIRA instance. Integrating JIRA applications and [HipChat](#) gives you and your team the following collaboration power:

- Get notifications in your HipChat rooms when a customer updates a service desk request, or a developer comments on an issue.
- Create a dedicated HipChat room from the issue you're working on and want to discuss with your team.
- Preview JIRA issues and service desk requests directly in HipChat when someone on your team mentions them.

Before you begin

The JIRA Server and HipChat integration shares information between the two applications in the following ways:

- **Push:** JIRA sends notifications to HipChat.
- **Pull:** HipChat retrieves information from JIRA. If your JIRA server is behind a firewall, you will need to make the server addressable from the internet (by assigning an addressable URL). If you are unable to access your JIRA server from behind the firewall, you can still use the integration, you will just be unable to receive pull messages such as JIRA Issue Preview. Alternatively, you can install and configure HipChat Server from behind the same firewall.

Connection status is displayed in the Connect field.

- **Connected:** HipChat and JIRA are connected and working just fine. Carry on.
- **Limited:** HipChat cannot connect to your JIRA server - it may be behind a firewall. You can still receive messages from JIRA in HipChat, but some functionality (such as Issue Preview and @mentions) may not work.
- **Not Connected:** Could not connect to the HipChat server. Integration features will be unavailable until the connection is restored. To diagnose connection issues, contact your JIRA Administrator.
- **Unknown:** HipChat cannot determine the connection status and may be unable to connect to your JIRA server. Some, or all, functionality may not work.

Linking JIRA and HipChat

1. Log in as a JIRA administrator or a Project Administrator.
2. Choose  **> Applications.**
3. Scroll down the page to the **Integrations** section and select **HipChat**.
4. Select **Connect HipChat**.
5. Follow the instructions on the screen to link JIRA to your HipChat site.
6. If your HipChat server runs over HTTPS, the SSL certificate must be imported into JIRA's trust store. See [Connecting to SSL services](#) for information on configuration.

Setting up JIRA notifications in HipChat

1. Sign in to JIRA as an administrator.
2. Choose  **> Applications.**
3. Under **Integrations**, select **HipChat**.
4. Create a link between a project and a HipChat room:
 - Select a project from the project drop-down menu
 - Select a HipChat room
 - Click **Add**
5. Alternatively, click **Edit** to change an existing link.
6. Configure the notification settings you'd like to use

1 STATUS LIMITED

2 SEND A MESSAGE TO THE ROOM WHEN

Project	Room		
Documentation Project	→ Collaboration Family Management Team	Edit	Delete
Documentation Project	→ Cooking	✓ Done	Delete

1. **Status:** Connection between JIRA and HipChat.
2. **Messages:** Select messages to send to the room.

Setting up JIRA issues in the HipChat sidebar

1. Sign in to HipChat. You'll need to be an administrator of the rooms you want to configure.
2. Select **Integrations** from the top menu.
3. From the drop-down, select the room you'd like to configure.
4. Select **Installed** to see the integrations that have been installed for this room.

2 Installed

5. Select the JIRA integration.
6. Select **Enable your glance**. The glance settings will appear.
7. Give your glance a name and set up a basic or JQL filter. The glance name should represent the filter's purpose.

Overview Configure

You are configuring JIRA for HipChat in room: **testroom**
To configure in a different room, select a new room from the dropdown above.

Enable your glance
Show filtered issues from your JIRA projects in the sidebar of your teams HipChat room.
You are using JIRA user **kbui** to configure this glance.

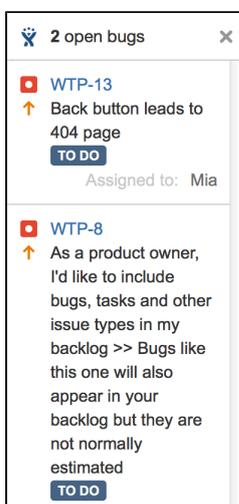
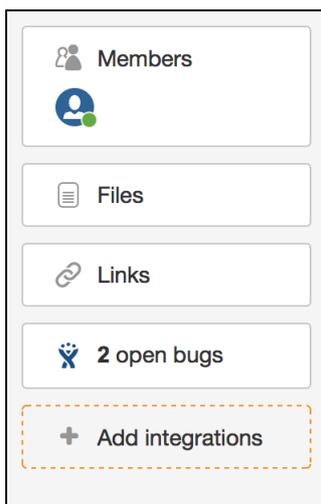
1 open bugs

Website Transformation Project Bug Priority: All Contains text More **Advanced** 2

Status: In Progress, To Do

Save Cancel

1. **Name:** This will display in the HipChat room's sidebar above your issues.
 2. **Issue filter:** The issues you want to display in the sidebar.
8. Click **Save**. Open your room in HipChat and click the glance to see these issues in your sidebar.



Remove OAuth Permissions

You can remove permissions that you have granted to allow JIRA to access HipChat. For instance, if you have given JIRA permission to invite users on HipChat's behalf.

1. Select your avatar to access your profile.
2. Click **Profile**.
3. Select **Tools**.
4. Click **HipChat OAuth Sessions**.
5. Select **Remove Access**.

Integrating with Portfolio for JIRA

Portfolio for JIRA provides a single, accurate place for viewing, planning and managing your work across multiple teams and projects. See [our guide](#) to how JIRA and Portfolio for JIRA work together.

Managing add-ons

About add-ons

An add-on is an installable component that supplements or enhances the functionality of JIRA in some way. For example, the [JIRA Calendar Plugin](#) is an add-on that shows the due dates for issues and versions in calendar format. Other add-ons are available for connecting JIRA to Bamboo, developing for JIRA, and accessing Atlassian support from JIRA.

JIRA comes with many pre-installed add-ons (called system add-ons). You can install more add-ons, either by acquiring an add-on from the [Atlassian Marketplace](#), or by uploading an add-on from your file system. This means that you can install add-ons that you have developed yourself. For information about developing your own add-ons for JIRA, see the [JIRA Developer documentation](#).

To enable various JIRA Gadgets, see [Configuring the default dashboard](#).

On this page:

- [About add-ons](#)
- [About the Universal Plugin Manager](#)

You may notice that the terms 'app', 'add-on' and 'plugin' all appear in the Atlassian documentation and tools. While the terms are often used interchangeably, there is a difference. A plugin is a type of app (formerly known as an add-on) that can be installed into an Atlassian host application. Plugins are what developers create with the Atlassian SDK.

About the Universal Plugin Manager

The Universal Plugin Manager (UPM) is itself an add-on that you use to administer add-ons from the JIRA Administration console. UPM works across Atlassian applications, providing a consistent interface for administering add-ons in JIRA, Confluence, Crucible, Fisheye, Bitbucket Server, or Bamboo.

UPM comes pre-installed in recent versions of all Atlassian applications, so you do not normally need to install it yourself. However, like other add-ons, the UPM software is subject to regular software updates. Before administering add-ons in JIRA, therefore, you should verify your version of the UPM, and update it if needed.

You can update UPM or any add-on from the UPM's own add-on administration pages. In addition to updating UPM, you can perform these tasks from the administration pages:

- Install or remove add-ons
- Configure add-on settings
- Discover and install new add-ons from the [Atlassian Marketplace](#)
- [Enable or disable add-ons](#) and their component modules, including "safe mode"

If the add-on request feature is enabled in your Atlassian application, non-administrative users can also

discover add-ons in the Atlassian Marketplace. Instead of installing the add-ons, however, these users have the option of requesting the add-ons from you, the administrator of the Atlassian application.

For more information on administering the add-on request feature or performing other common add-on administration tasks, see the [Universal Plugin Manager documentation](#).

Upgrading JIRA applications

There are several ways to upgrade JIRA, and the method you choose to use depends on which version of JIRA you use, and the type of environment you use it in. Use this table to determine which steps to follow to complete your JIRA upgrade:

If you need to move JIRA to a new server, or use it in a new environment that has a different operating system, different database type, or different location of attachment or index files, follow the instructions for [Migrating JIRA to another server](#).

Required uptime (SLA)	Hardware/Software Change	Operating system	JIRA package	Current JIRA version	Upgrade process
100% uptime <i>This method is recommended if you're running JIRA Software Data Center 7.3 or above, and require your JIRA instance to be available throughout the upgrade.</i>	Neither operation system, database or home directory will be changed.	MS Windows / Linux	JIRA Software Data Center	JIRA Software Data Center 7.3 and above	Managing zero downtime upgrades
High / Mission Critical <i>This method is recommended for enterprise environments where extended or unplanned downtime might negatively impact the business.</i>	Any	any	any	any	Upgrading JIRA with a fallback method
Low – Medium <i>If you use JIRA in a non-mission critical environment, depending on your specific installation details, you use either the installer or manual method to upgrade.</i>	Neither operation system, database or home directory will be changed.	MS Windows / Linux	Standalone	4.3.0 or later	Upgrading JIRA using the installer
				4.2.x or earlier	Upgrading JIRA manually
		Solaris	any	any	

Upgrading JIRA applications using the installer

In this guide, we'll run you through using the installer to upgrade your JIRA instance to the latest JIRA version on Windows or Linux.

Upgrading to any later version is free if you have current software maintenance. See our [Licensing FAQ](#) to find out more.



Before you begin

Before you upgrade JIRA, there's a few questions you need to answer.

<p>Is installer the right upgrade method for you?</p>	<p>Tell me more... You can choose to upgrade using the installer, or manually using a zip or tar.gz file. In most cases the installer is the easiest way to upgrade your JIRA instance.</p> <p>You will need to upgrade manually if:</p> <ul style="list-style-type: none"> • you are moving to a different operating system or database software as part of this upgrade. • you are upgrading from JIRA earlier than 4.3
<p>Are you eligible to upgrade?</p>	<p>Tell me more... To check if software maintenance is current for your license, go to > Applications > Versions and licenses, and make sure the license support period has not expired.</p> <p>If your support period has expired, follow the prompts to renew your license and reapply it before upgrading.</p>
<p>Have our supported platforms changed?</p>	<p>Tell me more... Check the Supported Platforms page for the version of JIRA you are upgrading to. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • The JIRA installer includes Java (JRE) and Tomcat, so you won't need to upgrade these separately. • If you need to upgrade your database, be sure to read the upgrade notes for the JIRA version you plan to upgrade to (and any in-between) to check for any database configuration changes that you may need to make.
<p>Do you need to make changes to your environment?</p>	<p>Tell me more... Newer JIRA versions sometimes require changes to your environment, such as providing more memory or adjusting your reverse proxy settings.</p> <p>Good to know:</p> <p>We use Upgrade Notes to communicate changes that will impact you, such as:</p> <ul style="list-style-type: none"> • Changes to supported databases, memory requirements or other changes that will impact your environment. • Features that have significantly changed or been removed in this release. • Actions you may need to take in your instance or environment immediately after the upgrade. <p>It's important to read the notes for the version you're upgrading to and those in-between. For example, if you are upgrading from 7.1 to 7.3 you should read the upgrade notes for 7.2 and 7.3.</p>

Check some known issues	<p>▼ Tell me more...</p> <p>Following the upgrade on Linux, you might encounter some problems with certain system text displayed in the application (CAPTCHA and gadgets). Instead of showing regular alphanumeric letters, the text will appear to be garbled and look like symbols. To avoid this problem, you should install several fonts that are required by JIRA. For more info, see JIRA UI shows unreadable text.</p>
-------------------------	---

Plan your upgrade

1. Determine your upgrade path

Use the tables below to determine the most efficient upgrade path from your current version to the latest versions of JIRA.

▼ [Show JIRA Core \(JIRA\)...](#)

Your version	Recommended upgrade path
Earlier than 4.4.5	Upgrade to 4.4.5 , then follow paths below (versions earlier than 4.3.0 will need to be upgraded manually).
4.4.5 to 5.1	Upgrade to any version between 5.2/6.4 , then follow paths below.
5.2 to 6.4	Upgrade to 7.0* , then upgrade to any later version.
7.0 or later	Upgrade to any later version of JIRA Core.

*The 7.0 version introduced significant changes that affect your user management, application access, and JIRA installation setup. Consult the [Migration hub](#) before upgrading.

▼ [Show JIRA Software \(JIRA + Agile\)...](#)

Your version	Recommended upgrade path
Earlier than 4.4.5	Upgrade to 4.4.5 , then follow paths below (versions earlier than 4.3.0 will need to be upgraded manually).
4.4.5 to 5.1	Upgrade to any version between 5.2/6.4 , then follow paths below.
5.2 to 6.4	Upgrade to 7.0* , then upgrade to any later version.
7.0 or later	Upgrade to any later version of JIRA Software.

*The 7.0 version introduced significant changes that affect your user management, application access, and JIRA installation setup. Consult the [Migration hub](#) before upgrading.

▼ [Show JIRA Service Desk...](#)

Your version	Recommended upgrade
Earlier than 2.5.x	Upgrade to 3.0 , then upgrade to the latest version.
3.0	Upgrade directly to the latest version.

2. Complete the pre-upgrade checks

1. Check the [Upgrade Notes](#) for the version you plan to upgrade to (and any in between).
2. Go to



> **System > Support Tools**, and check the results of the health check. Fix any detected problems.

▼ [License expired?](#)

If the software maintenance period included in your license has expired, you can keep using JIRA, but you'll need to renew before you can upgrade.

Go to



> **Applications > Versions and licenses**, and follow the prompts to renew your license.

▼ [Still using the embedded database?](#)

If you are using the **embedded (trial) database**, you should migrate to a different database **before** upgrading.

This database is supplied for evaluation purposes only and is not recommended for production environments.

3. Go to



> **Add-ons**, and scroll down to **JIRA update check** to check the compatibility of your add-ons. Choose the version you plan to upgrade to, then hit **Check**.

▼ [If you have incompatible add-ons...](#)

If your users rely on particular add-ons, you may want to wait until they are compatible before upgrading JIRA. Add-on vendors generally update their add-ons very soon after a major release.

Good to know:

- Compatibility information is often not available immediately after a new release. In many cases, the add-on will still work, so give it a try in a test environment before upgrading the production one

3. Upgrade JIRA in a test environment

1. Create a staging copy of your current production environment.
See [Establishing staging server environments](#) for help creating an environment to test your upgrade in.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported add-ons and customizations before upgrading your production environment.

Upgrade JIRA

4. Back up

1. Back up your database and confirm the backup was created properly.
You can use the database native tools to create the backup. If your database does not support online backups, you'll need to stop JIRA first.
2. Back up your **installation directory** and **home directory**.
The installation wizard will back up your JIRA directories as part of the installation process, but you should also back these directories up manually before starting the upgrade.

▼ [Where is my home directory?](#)

You can find the location of your home directory in the <installation-directory>/atlassian-jira/WEB-INF/classes/jira-application.properties file.

The default paths are:

Windows: C:\Program Files\Atlassian\Application Data\JIRA

Linux: /var/atlassian/application-data/jira

5. Download JIRA

Download the installer for your operating system – JIRA Core, JIRA Software, or JIRA Service Desk.

▼ [Good to know...](#)

- JIRA Software and JIRA Service Desk already contain the JIRA Core installer.
- If you're upgrading both JIRA Core/Software and JIRA Service Desk, upgrade JIRA Core/Software only. You'll later update Service Desk directly in JIRA, without a separate installer.

6. Run the installer

1. Run the installer.

▼ Show me how to do this in Windows...

Run the .exe file. We recommend using a Windows administrator account.

If prompted to allow the upgrade wizard to make changes to your computer, choose 'Yes'. If you do not, the installation wizard will have restricted access to your operating system and any subsequent installation options will be limited.

▼ Show me how to do this in Linux...

Change to the directory where you downloaded JIRA, then execute this command to make the installer executable:

```
$ chmod a+x atlassian-jira-X.X.X-x64.bin
```

Where `x.x.x` is the JIRA version you downloaded.

Next, run the installer – we recommend using `sudo` to run the installer:

```
$ sudo ./atlassian-jira-X.X.X-x64.bin
```

You can also choose to run the installer with root user privileges.

2. Follow the prompts to upgrade JIRA:

- When prompted, choose **Upgrade an existing JIRA installation**.
- Make sure the **Existing JIRA installation directory** suggested by the wizard is correct (especially important if you have multiple JIRA installations on the same machine.)
- Back up these directories** is strongly recommended. This will create a .zip backup of the JIRA home and installation directories.
- The wizard notifies you of customizations in the JIRA installation directory. **Make a note of these** as you'll need to reapply them later. Your current customizations will be overwritten.

3. When JIRA starts for the first time after an upgrade, you'll be presented with an overview of the upgrade. This page will show to the first 10 admins who log in to JIRA post upgrade, and only for 14 days. This overview contains information on critical updates to your JIRA instance, and it also gives you a dynamic overview of your add-ons and application links. You can access this information at any time by selecting



> **Latest upgrade report.**

4. There are some additional steps you may need to take if:

- you use **Crowd** for user management

▼ Click here to expand...

If you are using Crowd for user management, reapply the modifications from the following files from your existing installation directory to the new files. Do not copy the files as they may be different in the new version of JIRA.

- `<Installation-Directory>/atlassian-jira/WEB-INF/classes/crowd.properties`
- `<Installation-Directory>/atlassian-jira/WEB-INF/classes/seraph-config.xml`

After the upgrade

7. Upgrade the database driver

If you're using an Oracle or MySQL database, download the JDBC driver and place it in `<JIRA-installation-directory>/lib`.

- Oracle: JDBC driver 12.1.0.1.
- MySQL: the latest JDBC driver.

8. Re-apply any modifications

During the upgrade, the wizard migrated the following from your existing JIRA installation:

- TCP values in the `server.xml` file.
- Location of your JIRA home directory in the `jira-application.properties` file.
- The following values in the `setenv.sh / setenv.bat` file:
 - `JVM_SUPPORT_RECOMMENDED_ARGS`
 - `JVM_MINIMUM_MEMORY`
 - `JVM_MAXIMUM_MEMORY`
 - `JIRA_MAX_PERM_SIZE`

All other customizations need to be reapplied manually.

▼ Show me how to do this...

Any other configurations or customizations (including any other modifications in the `server.xml` file) are **not migrated** during the upgrade and need to be reapplied manually.

1. Stop your upgraded JIRA instance.
2. Reapply the customizations to the relevant files in the upgraded JIRA installation directory.
3. Restart the upgraded JIRA instance.

We **strongly recommend** you test your customizations in a test instance prior to upgrading your production instance as changes may have been made to JIRA that make your customizations unusable.

▼ Show me the list of important files...

If you're unsure which files you might have modified, take a look at the list of [important files in the JIRA installation directory](#).

9. (Optional) Update JIRA Service Desk

If you're using JIRA Service Desk, you can update it directly in the UI, without downloading a separate installer.

1. Go to  **> Applications > Versions and licenses.**
2. Update JIRA Service Desk. This will automatically update Service Desk to a compatible version.

10. Upgrade your plugins

Once you have confirmed the availability of compatible versions, you should upgrade your plugins after successfully upgrading JIRA. This can be done via



> Add-ons > Manage add-ons.

Troubleshooting

Did something go wrong?

Check the resources on our [Support](#) page, or raise an issue so we can help you resolve your problem.

If you need to retry the upgrade, **you must roll back to your previous setup first**. Do not attempt to run an upgrade again, or start the older version of JIRA again after an upgrade has failed.

Upgrading JIRA applications manually

In this guide, we'll run you through upgrading JIRA manually, if your installation doesn't support using the installer. Upgrading to any later version is free if you have current software maintenance. See our [Licensing FAQ](#) to find out more.

You should use this method to upgrade JIRA if you meet any of the following criteria:



Before you begin

Before you upgrade JIRA, there's a few questions you need to answer.

<p>Is manual the right upgrade method for you?</p>	<p>Tell me more...</p> <p>You can choose to upgrade using the installer, or manually using a zip or tar.gz file. In most cases the installer is the easiest way to upgrade your JIRA instance.</p> <p>You will need to upgrade manually if:</p> <ul style="list-style-type: none"> • you are moving to a different operating system or database software as part of this upgrade. • you use JIRA 4.0.0 or later on Solaris • you are upgrading from JIRA earlier than 4.3 <p>Note that if you're upgrading only JIRA Service Desk, you'll need to use the installer.</p>
<p>Are you eligible to upgrade?</p>	<p>Tell me more...</p> <p>To check if software maintenance is current for your license, go to > Applications > Versions and licenses, and make sure the license support period has not expired.</p> <p>If your support period has expired, follow the prompts to renew your license and reapply it before upgrading.</p>
<p>Have our supported platforms changed?</p>	<p>Tell me more...</p> <p>Check the Supported Platforms page for the version of JIRA you are upgrading to. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you need to upgrade your database, be sure to read the upgrade notes for the JIRA version you plan to upgrade to (and any in-between) to check for any database configuration changes that you may need to make.

Do you need to make changes to your environment?	<p>▼ Tell me more...</p> <p>Newer JIRA versions sometimes require changes to your environment, such as providing more memory or adjusting your reverse proxy settings.</p> <p>Good to know:</p> <p>We use Upgrade Notes to communicate changes that will impact you, such as:</p> <ul style="list-style-type: none"> • Changes to supported databases, memory requirements or other changes that will impact your environment. • Features that have significantly changed or been removed in this release. • Actions you may need to take in your instance or environment immediately after the upgrade. <p>It's important to read the notes for the version you're upgrading to and those in-between. For example, if you are upgrading from 7.1 to 7.3 you should read the upgrade notes for 7.2 and 7.3.</p>
Check some known issues	<p>Tell me more...</p> <p>Following the upgrade on Linux, you might encounter some problems with certain system text displayed in the application (CAPTCHA and gadgets). Instead of showing regular alphanumeric letters, the text will appear to be garbled and look like symbols. To avoid this problem, you should install several fonts that are required by JIRA. For more info, see JIRA UI shows unreadable text.</p>

Plan your upgrade

1. Determine your upgrade path

Use the tables below to determine the most efficient upgrade path from your current version to the latest versions of JIRA.

▼ [Show JIRA Core \(JIRA\)...](#)

Your version	Recommended upgrade path
Earlier than 4.4.5	Upgrade to 4.4.5 , then follow paths below (versions earlier than 4.3.0 will need to be upgraded manually).
4.4.5 to 5.1	Upgrade to any version between 5.2/6.4 , then follow paths below.
5.2 to 6.4	Upgrade to 7.0* , then upgrade to any later version.
7.0 or later	Upgrade to any later version of JIRA Core.

*The 7.0 version introduced significant changes that affect your user management, application access, and JIRA installation setup. Consult the [Migration hub](#) before upgrading.

▼ [Show JIRA Software \(JIRA + Agile\)...](#)

Your version	Recommended upgrade path
Earlier than 4.4.5	Upgrade to 4.4.5 , then follow paths below (versions earlier than 4.3.0 will need to be upgraded manually).
4.4.5 to 5.1	Upgrade to any version between 5.2/6.4 , then follow paths below.
5.2 to 6.4	Upgrade to 7.0* , then upgrade to any later version.
7.0 or later	Upgrade to any later version of JIRA Software.

*The 7.0 version introduced significant changes that affect your user management, application access, and JIRA installation setup. Consult the [Migration hub](#) before upgrading.

2. Complete the pre-upgrade checks

1. Check the [Upgrade Notes](#) for the version you plan to upgrade to (and any in between).
2. Go to



> **System > Support Tools**, and check the results of the health check. Fix any detected problems.

▼ [License expired?](#)

If the software maintenance period included in your license has expired, you can keep using JIRA, but you'll need to renew before you can upgrade.

Go to



> **Applications > Versions and licenses**, and follow the prompts to renew your license.

▼ [Still using the embedded database?](#)

If you are using the **embedded (trial) database**, you should migrate to a different database **before** upgrading.

This database is supplied for evaluation purposes only and is not recommended for production environments.

3. Go to



> **Add-ons**, and scroll down to **JIRA update check** to check the compatibility of your add-ons.

Choose the version you plan to upgrade to, then hit **Check**.

▼ [If you have incompatible add-ons...](#)

If your users rely on particular add-ons, you may want to wait until they are compatible before upgrading JIRA. Add-on vendors generally update their add-ons very soon after a major release.

Good to know:

- Compatibility information is often not available immediately after a new release. In many cases, the add-on will still work, so give it a try in a test environment before upgrading the production one.

3. Upgrade JIRA in a test environment

1. Create a staging copy of your current production environment.
See [Establishing staging server environments](#) for help creating an environment to test your upgrade in.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported add-ons and customizations before upgrading your production environment.

Upgrade JIRA

4. Back up

1. Create an XML backup of your database.
 - ▼ [Show me how to do this...](#)
 - a. Stop users from updating JIRA data.
 - b. Perform an XML backup.
2. Back up your **installation directory** and **home directory**.
 - ▼ [Where is my home directory?](#)
You can find the location of your home directory in the `<installation-directory>/atlassian-jira/WEB-INF/classes/jira-application.properties` file.

The default paths are:

Windows: C:\Program Files\Atlassian\Application Data\JIRA

Linux: /var/atlassian/application-data/jira

3. As soon as the backup is complete, delete the `<home-directory>/dbconfig.xml` file from your existing home directory. This will allow you to connect JIRA to a new database.

5. Download JIRA

Download the zip or tar.gz archive – [JIRA Core](#) or [JIRA Software](#).

▼ [Good to know...](#)

- JIRA Software already contains the JIRA Core installer.
- If you're upgrading both JIRA Core/Software and JIRA Service Desk, upgrade JIRA Core/Software only. You'll later update Service Desk directly in JIRA, without a separate installer.

6. Extract the file and upgrade JIRA

1. Stop JIRA applications.
2. Extract (unzip) the files to a directory (this is your new installation directory, and must be different to your existing installation directory).
3. Edit the `<installation-directory>\atlassian-jira\WEB-INF\classes\jira-application.properties` file to point to your **existing** JIRA home directory. Make sure that you deleted the `dbconfig.xml` file, otherwise JIRA will try to connect to your existing database.
4. Copy your jdbc driver jar file from your existing JIRA installation directory to `atlassian-jira/WEB-INF/lib` in your **new installation directory**.
5. There are some additional steps you may need to take if:
 - you use **Crowd** for user management
 - ▼ [Click here to expand...](#)
 - If you are using Crowd for user management, reapply the modifications from the following files from your existing installation directory to the new files. Do not copy the files as they may be different in the new version of JIRA.
 - `<Installation-Directory>/atlassian-jira/WEB-INF/classes/ouser.xml` (4.3.0 and earlier)
 - `<Installation-Directory>/atlassian-jira/WEB-INF/classes/crowd.properties`
 - `<Installation-Directory>/atlassian-jira/WEB-INF/classes/seraph-config.xml`
 - you are upgrading from version **4.3.0 or earlier**
 - ▼ [Click here to expand...](#)
 - The `jira-application.properties` file, which you might know from advanced configuration, now contains only the location of your JIRA home directory.
 - You should reapply your advanced configuration settings directly in JIRA by going to  **> System > Advanced Settings**.

7. Start JIRA and connect it to the database

1. Start JIRA to launch the setup wizard. Go to `<installation-directory>/bin`, and run the `start-jira.sh / start-jira.bat` file.
2. Access JIRA through your browser, and choose **I'll set it up myself** to get access to more setup options.

3. Select **My own database**, and provide details of a new, empty database.
4. Click **Import your data**, and select the file with your XML backup.

Avoid passing through a proxy when performing an XML restore, especially if your JIRA instance is very large. Using a proxy may cause timeout errors.

5. When JIRA starts for the first time after an upgrade, you'll be presented with an overview of the upgrade. This page will show to the first 10 admins who log in to JIRA post upgrade, and only for 14 days. This overview contains information on critical updates to your JIRA instance, and it also gives you a dynamic overview of your add-ons and application links. You can access this information at any time by selecting



> Latest upgrade report.

6. Take a quick look around your JIRA site to confirm that your projects and issues are present and everything looks normal. You should see the new JIRA version number in the page footer.

After the upgrade

8. Re-apply any modifications

If you have customized JIRA (such as an SSL configuration in the `server.xml` file, or additional memory allocation in the `setenv.sh` / `setenv.bat` file), you'll need to reapply the customizations to the relevant files in the newly upgraded JIRA installation directory.

▼ Show me how to do this...

Any other configurations or customizations (including any other modifications in the `server.xml` file) are **not migrated** during the upgrade and need to be reapplied manually.

1. Stop your upgraded JIRA instance.
2. Reapply the customizations to the relevant files in the upgraded JIRA installation directory.
3. Restart the upgraded JIRA instance.

We **strongly recommend** you test your customizations in a test instance prior to upgrading your production instance as changes may have been made to JIRA that make your customizations unusable.

▼ Show me the list of important files...

If you're unsure which files you might have modified, take a look at the list of [important files in the JIRA installation directory](#).

9. (Optional) Update JIRA Service Desk

If you're using JIRA Service Desk, you can update it directly in the UI, without downloading a separate installer.

1. Go to



> Applications > Versions and licenses.

2. Update JIRA Service Desk. This will automatically update Service Desk to a compatible version.

Troubleshooting

Did something go wrong?

Check the resources on our [Support](#) page, or raise an issue so we can help you resolve your problem.

If you need to retry the upgrade, **you must roll back to your previous setup first**. Do not attempt to run an upgrade again, or start the older version of JIRA again after an upgrade has failed.

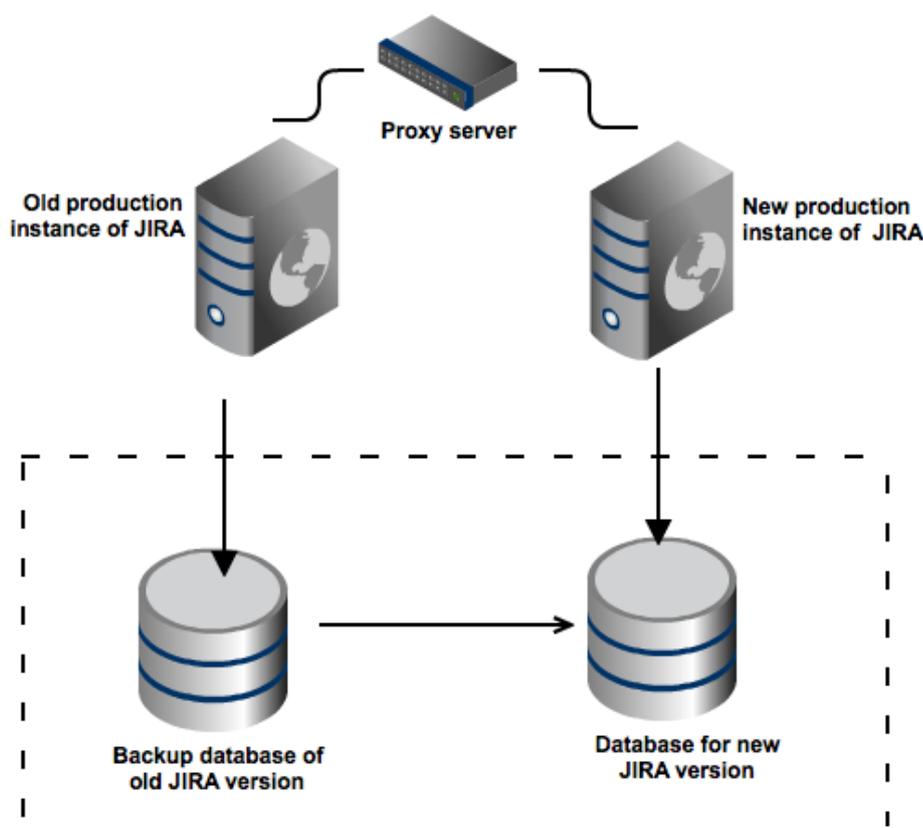
Upgrading JIRA applications with a fallback method

This page describes how to upgrade JIRA in a way that allows you to safely roll back to your previous system if the upgrade process takes longer than expected, or if you encounter issues. This method is especially useful for enterprise environments and for organizations where JIRA is mission-critical for the business, and you can't allow prolonged downtime.

About the upgrade

You'll need to set up a proxy server to have a quick way of redirecting your users either to the existing or to the new instance of JIRA. Then, you'll replicate your existing environment and upgrade the new instance. This is required to preserve your existing environment and have a fallback option, if the upgrade is unsuccessful, or if you simply prefer the old version.

This graphic illustrates the process described in this document. For simplicity, the illustration shows how you can perform an upgrade using two different pieces of hardware. However, you can just as easily install JIRA in different directories on the same server to test and perform an upgrade. In this case, simply ensure that you use separate installation and database directories during the testing.



Upgrading JIRA with a fallback method

1. Set up a proxy server

Before beginning the upgrade process, set up a reverse proxy, such as a load balancer. The proxy server allows you to redirect users to a different JIRA server without having to wait for a DNS change. If, at any point during the upgrade process, you encounter issues you can't resolve, you can restart your existing JIRA instance and reconfigure the proxy server to point to the old server.

If you use monitoring, API calls (such as SOAP, REST, or CLI), or scripts associated with your production server, update them with the new proxy information.

Please see the following documentation for further information on configuring Apache:

- [Integrating JIRA with Apache](#)

- [Integrating JIRA with Apache using SSL](#)

2. Disable your existing JIRA and create backups

Disable your existing JIRA, so that users don't create new data. Next, create backups of your database, and the home and installation directories.

1. Back up your database and confirm the backup was created properly.
You can use the database native tools to create the backup.
2. Back up your **installation directory** and **home directory**.
 - ▼ [Where is my home directory?](#)
You can find the location of your home directory in the <installation-directory>/atlassian-jira/WEB-INF/classes/jira-application.properties file.

The default paths are:

Windows: C:\Program Files\Atlassian\Application Data\JIRA

Linux: /var/atlassian/application-data/jira

3. Set up a new instance of JIRA

The easiest way to set up a new environment is to use the procedure described in [Establishing staging environments](#) to copy the whole JIRA into a new directory. You can also install the new instance, and then restore the database, home directory, and all your customizations.

4. Upgrade your new JIRA

Perform a regular upgrade. For more info, see [Upgrading JIRA using the installer](#).

5. Verify the upgrade and redirect the proxy

Take a look around your new instance and verify that everything is working properly.

- If you're happy with the outcome, redirect the proxy server to the new instance.
- If something is not right, redirect the proxy server to the old instance. Your users can resume work, while you prepare for the new upgrade.

Upgrading JIRA Data Center

This page describes how to upgrade JIRA Data Center. During the upgrade, you'll shut down your JIRA instances, and then copy the new version to each node, upgrading them one by one. If you'd rather have your JIRA working during the upgrade, we recommend that you use [Managing zero downtime upgrades](#) instead.

If you're currently using JIRA Software Server, learn more about upgrading to Data Center [here](#).

Before you begin

Before you upgrade JIRA, there's a few questions you need to answer.

<p>Is this the right upgrade method for you?</p>	<p>Tell me more...</p> <p>This method requires that you shut down JIRA on each of your nodes. It's mostly about upgrading one node, and then copying the upgraded directories to remaining nodes.</p> <p>If you can't allow any downtime in your environment, you should use a different method: Managing zero downtime upgrades.</p>
--	---

Are you eligible to upgrade?	<p>▼ Tell me more...</p> <p>To check if software maintenance is current for your license, go to  > Applications > Versions and licenses, and make sure the license support period has not expired.</p> <p>If your support period has expired, follow the prompts to renew your license and reapply it before upgrading.</p>
Have our supported platforms changed?	<p>Tell me more...</p> <p>Check the Supported Platforms page for the version of JIRA you are upgrading to. This will give you info on supported operating systems, databases and browsers.</p> <p>Good to know:</p> <ul style="list-style-type: none"> • If you need to upgrade your database, be sure to read the Upgrade Notes for the JIRA version you plan to upgrade to (and any in-between) to check for any database configuration changes that you may need to make.
Do you need to make changes to your environment?	<p>Tell me more...</p> <p>Newer JIRA versions sometimes require changes to your environment, such as providing more memory or adjusting your reverse proxy settings.</p> <p>Good to know:</p> <p>We use Upgrade Notes to communicate changes that will impact you, such as:</p> <ul style="list-style-type: none"> • Changes to supported databases, memory requirements or other changes that will impact your environment. • Features that have significantly changed or been removed in this release. • Actions you may need to take in your instance or environment immediately after the upgrade. <p>It's important to read the notes for the version you're upgrading to and those in-between.</p>

Plan your upgrade

1. Complete the pre-upgrade checks

1. Check the [Upgrade Notes](#) for the version you plan to upgrade to (and any in between).

2. Go to



> **System > Support Tools**, and check the results of the health check. Fix any detected problems.

▼ [License expired?](#)

If the software maintenance period included in your license has expired, you can keep using JIRA, but you'll need to renew before you can upgrade.

Go to



> **Applications > Versions and licenses**, and follow the prompts to renew your license.

▼ [Still using the embedded database?](#)

If you are using the **embedded (trial) database**, you should migrate to a different database **before** upgrading.

This database is supplied for evaluation purposes only and is not recommended for production environments.

3. Go to



> **Add-ons**, and scroll down to **JIRA update check** to check the compatibility of your add-ons. Choose the version you plan to upgrade to, then hit **Check**.

▼ [If you have incompatible add-ons...](#)

If your users rely on particular add-ons, you may want to wait until they are compatible before upgrading JIRA. Add-on vendors generally update their add-ons very soon after a major release.

Good to know:

- Compatibility information is often not available immediately after a new release. In many cases, the add-on will still work, so give it a try in a test environment before upgrading the production one.

2. Upgrade JIRA in a test environment

1. Create a staging copy of your current production environment.
See [Establishing staging server environments](#) for help creating an environment to test your upgrade in.
2. Follow the steps below to upgrade your test environment.
3. Test any unsupported add-ons and customizations before upgrading your production environment.

Upgrade JIRA

3. Stop the cluster

You need to stop JIRA on all nodes in the cluster. We also recommend that you configure your load balancer to redirect the traffic away from JIRA until the upgrade is complete on all nodes.

4. Back up

Back up the following components:

- **JIRA database.** You can use the database native tools to create the backup.
- **JIRA installation directory.** It's enough if you back up only one of the whole cluster.
- **JIRA shared directory.** That's a subdirectory of the home directory that is accessible by all nodes in the cluster.
- **JIRA home directory** on each node.

5. Delete the old add-on cache

Delete the old add-on cache on each node in the cluster, so that it doesn't affect the new version. Go to `<home-directory>/plugins/` on each node, and delete the contents of this directory.

For major upgrades, you can delete the whole `plugins` directory. We've found that upgrading without plugins tends to be faster and more reliable.

6. Upgrade the first node

To upgrade the first node:

1. Download [JIRA Software](#) or [JIRA Service Desk Data Center](#). Choose the `tar.gaz` or ZIP archive to speed up the process.
▼ [Good to know...](#)
If you're using both JIRA Software and JIRA Service Desk, upgrade JIRA Software only. You'll later update Service Desk directly in JIRA, without a separate installer.
2. Extract the files to a directory (this will be your new installation directory, and must be different to your existing installation directory.)
3. Go to `<installation-directory>/bin`, and edit the `setenv.bat` / `setenv.sh` file. In `JIRA_HOME`, enter the path to the **existing local home directory** on that node.

You'll find the `JIRA_HOME` parameter at the top of the file. You'll also need to uncomment it.

4. If you're using a MySQL database, download the latest jdbc driver, and place it in `<installation-directory>/lib`.
5. Reapply any customizations, like JVM properties, from the old version to the new one.
6. Copy the new **installation directory**. You will later use it as a template to replicate JIRA to other nodes.
7. Start JIRA. During this step, your database will be upgraded.
8. When JIRA starts for the first time after an upgrade, you'll be presented with an overview of the upgrade. This page will show to the first 10 admins who log in to JIRA post upgrade, and only for 14 days. This overview contains information on critical updates to your JIRA instance, and it also gives you a dynamic overview of your add-ons and application links. You can access this information at any time by selecting  **> Latest upgrade report**.
9. Go to  **> Add-ons > Manage add-ons**, and upgrade your add-ons to the supported versions.
10. Rebuild index in JIRA. For more info, see [Search indexing](#).

7. Copy JIRA to remaining nodes

To replicate JIRA to other nodes:

1. Copy the **installation directory** (the template you created before) to the new node.
2. If the path to the home directory is different on this node, update it in the `setenv.bat / setenv.sh` file.
3. Change indexes on the new node:
 - a. Go to `<home-directory>/caches/indexes`, and delete the contents of this directory.
 - b. Start JIRA on the new node.
 - c. After JIRA is started, verify that the indexes were automatically copied from the first node. If not, you'll need to copy them manually.

Repeat this process for each remaining node.

8. (Optional) Update JIRA Service Desk

If you're using JIRA Service Desk, you can update it directly in the UI, without downloading a separate installer.

1. Go to  **> Applications > Versions and licenses**.
2. Update JIRA Service Desk. This will automatically update Service Desk to a compatible version.

Troubleshooting

Did something go wrong?

Check the resources on our [Support](#) page, or raise an issue so we can help you resolve your problem.

If you need to retry the upgrade, **you must roll back to your previous setup first**. Do not attempt to run an upgrade again, or start the older version of JIRA again after an upgrade has failed.

Managing zero downtime upgrades

Zero downtime upgrades allow you to upgrade your nodes with no downtime for your users i.e. your instance will remain available throughout the upgrade process. We'll explain what you need to do before you begin, and the procedure you need to follow to get your instance upgraded. We recommend you read this page and make sure you understand the process before starting your upgrade.

Note that this process applies to both JIRA Software and JIRA Service Desk Data Center installations. JIRA Software 7.3 or JIRA Service Desk 3.6 are the *minimum* versions you need to be able to use this upgrade

process. If you're running a JIRA installation with both JIRA Software and JIRA Service Desk, don't worry, we got you covered!

As the zero downtime upgrade process can be quite lengthy (depending on how many nodes you have), we've also got a [handy checklist](#) you can use to make sure you've done everything you need to. We still recommend you go through all the steps on this page, the checklist is just a handy tool to help keep track of what you're doing.

Before you begin

Before you start your zero downtime upgrade, there's a few things you'll do:

<p>Ensure you have the installer for your intended version.</p>	<p>▼ Tell me more... Download the appropriate JIRA Software or JIRA Service Desk installer. Make sure you have it available for all your nodes, as you'll need it to upgrade them one at a time.</p>
<p>Ensure you know about your intended version.</p>	<p>▼ Tell me more... Review the release notes and upgrade notes for the version of JIRA platform, JIRA Software and JIRA Service Desk that you're upgrading to. If you plan to skip a few versions during your upgrade, we strongly recommend that you read the release notes and upgrade guides for all versions between your current version and the version to which you are upgrading. Remember, the <i>minimum</i> version you need to be on to use zero downtime upgrades is JIRA Software 7.3, or JIRA Service Desk 3.6.</p>
<p>Ensure you're on a supported platform.</p>	<p>▼ Tell me more... Confirm that your operating system, database, and other applicable platforms and hardware still comply with the requirements for intended JIRA version. The End of Support Announcements page also has important information regarding platform support for future versions of JIRA.</p>

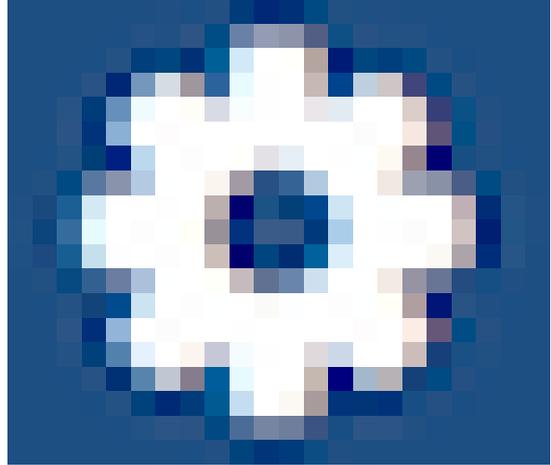
<p>Ensure you check your add-ons.</p>	<p> Tell me more... If you have installed JIRA add-ons (i.e. not included with JIRA), verify that they will be compatible with the version of JIRA Software you are upgrading to. You can find add-on compatibility information on the add-on's home page on the Atlassian Marketplace. </p> <ul style="list-style-type: none"> • If the add-on currently supports your current and your intended version, you don't need to do anything. • If the add-on has an update available that supports your current and your intended version, update the add-on <i>before</i> you start your JIRA upgrade. • If the add-on supports your current version, but requires an update to support your intended version (and the update doesn't support your current version), you should disable the add-on to avoid any issues during the upgrade. Once the upgrade is complete and finalized, you can update and re-enable the add-on. This will mean the add-on is unavailable during the upgrade. • If the add-on supports your current version, but not your intended version (and there is no update available that supports your intended version), it may cause issues during the upgrade and should be disabled if you decide to proceed with your upgrade. You should not enable the add-on until such time as it supports your new version, and can be updated.
<p>Back up your database</p>	<p> Tell me more... Zero downtime upgrades rely on the database being available throughout the upgrade process, so you should perform a database backup using your database's native tools as close to the start of your upgrade procedure as you can. </p>

Ensure your Support Healthcheck, Instance Health, and Support Tools plugins are enabled, and up to date with the latest versions.

▼ [Tell me more...](#)

Zero downtime upgrades checks rely on both of these plugins being enabled to make sure JIRA has the correct information to help you during your zero downtime upgrade. These are enabled by default, but to check you need to:

1. Navigate to



> **Add-ons > Manage add-ons.**

2. Search for the plugins by entering "health" or "support" as applicable, and selecting **System** in the drop-down.
3. Expand the relevant plugin to check if it's enabled or disabled. If disabled, enable it by clicking **Enable**.

If there's an update available for any of the plugins, you'll see an **Update** button. Click it to update to the latest available version.

Prestaging and testing your new version

We strongly recommend performing your upgrade in a test environment that best reflects your production environment first. Don't upgrade your production instance until you are satisfied that your test environment upgrade has been successful and is functioning correctly. This includes checking your add-ons and customizations. If you have any problems with your test environment upgrade which you can't resolve, create an issue at our [support site](#) so that we can assist you.

1. Put JIRA into upgrade mode

Putting JIRA into upgrade mode allows the nodes in your cluster to run on different versions of JIRA Software or JIRA Service Desk until you finalize or cancel your upgrade. To put JIRA into upgrade mode, all your nodes need to be running on the same versions.

1. Navigate to



> **Applications > JIRA upgrades.**

2. Click **Put JIRA into upgrade mode**. This will only be available if your nodes are all on the same version.

When you first put JIRA into upgrade mode, you have the option to cancel the upgrade, which will take JIRA out of upgrade mode. This is available until you start upgrading your nodes. Once you've upgraded your nodes to the same version, you can finalize the upgrade. To cancel the upgrade, you'd need to roll each node back to its original version.

2. Update your JIRA applications

If you run both JIRA Software and JIRA Service Desk in your cluster, then you need to copy the JIRA Service Desk files manually into your JIRA shared home. If you're running JIRA Software or JIRA Service Desk alone, then you can skip this step.

1. Download the required [JIRA Service Desk OBR](#) file. Make sure you download the version that is compatible with the JIRA Software version you're going to install. You can check the compatibility [here](#).
2. Change the extension of the OBR file you downloaded from .obr to .zip.
3. Unzip this file to extract the contents.
4. Copy all the jar files from the directory where you extracted the contents of the zip file and from the child directory "dependencies" and place them in <JIRA shared home>/plugins/installed-plugins (Read more about the shared home directory in section 3 [here](#)). The terminal commands are:
 - a. Linux: `cp *.jar dependencies/*.jar <JIRA shared home>/plugins/installed-plugins`
 - b. Windows: `copy *.jar + dependencies/*.jar <JIRA shared home>/plugins/installed-plugins`

During the upgrade process, upgraded nodes will pick up the new JIRA Service Desk jar files from the shared home, while nodes that haven't been upgraded will continue using the old versions of jars. When your upgrade is complete, all your nodes will be running the new version of JIRA Service Desk.

3. Upgrade your nodes

Once your JIRA instance is in upgrade mode, you can upgrade each node individually. Upgrading a node will involve stopping JIRA, upgrading the installation, and then starting JIRA. Stopping JIRA will remove the node from your cluster, making it unavailable, and any users logged in to that node will lose their current session, before being routed to another node. As the administrator, it's up to you to decide which nodes to upgrade and in which order. You always need to have at least one node online and connected to your cluster to achieve zero downtime. There's some useful information on monitoring a JIRA Data Center node [here](#), and depending on your setup, you may be able to 'drain' your nodes to minimize impact.

You'll be upgrading your nodes by using the JIRA installer, just like in the case of a regular upgrade. The installer performs most of the upgrade tasks for you. However, if you have made customizations to your JIRA installation, you'll need to migrate the customized files manually to the upgraded installation.

You're now ready to upgrade your nodes. Make sure you can access the installer before you start, and select the node you want to upgrade. Then, follow the procedure described below.

Upgrade JIRA by using the installer

When you've upgraded your first node, check that it's available on your cluster, and that users are able to log in and use the node. Once you've done this, repeat the process for each node.

4. Finalize your upgrade

Finalizing an upgrade will allow any required upgrade tasks to run on your instance, and take JIRA out of upgrade mode. Once the required tasks have completed, your installation is upgraded.

1. Navigate to  **> Applications > JIRA upgrades**.
2. Click **Finalize upgrade**. This will only be available if all your nodes are all on the same, new version.

Congratulations! You've upgraded your instance and achieved zero downtime!

Zero downtime upgrade checklist

When you're performing a zero downtime upgrade on a JIRA Software or JIRA Service Desk Data Center instance, it's important that you complete each step and verify it's been successful. You can use this checklist to make sure that you've literally ticked all the boxes.

Before you begin

We strongly recommend performing the upgrade in a test environment first.

Ensure you have the installer for your intended version.

Ensure you know about your intended version by reviewing the release notes.

Ensure you're on a [supported platform](#).

Ensure you check your add-ons.

Back up your database using your database's native tools.

Ensure your Support Healthcheck, Instance Health, and Support Tools plugins are enabled, and up to date with the latest versions.

Ensure autoscaling is disabled if running in AWS or any other private cloud environment that supports autoscaling

Before upgrading a node

Ensure you have initiated ZDU (cluster is in 'Ready to upgrade' mode).

Ensure no long-running tasks are active.

Before running upgrade tasks

Ensure all nodes have rejoined the cluster on the new version.

Ensure all nodes load JIRA as expected.

Run your own smoke tests or test suite to ensure all works as expected.

Back up your database.

Download the checklist so that you have it available for every node.

[Download checklist](#)

Upgrade task troubleshooting

When you upgrade JIRA Software or JIRA Service Desk Data Center using zero downtime upgrades, and depending on what version you're upgrading to, your instance may need to run upgrade tasks on your data. Occasionally, these upgrade tasks will fail to complete, or fail to run correctly. This can be for a variety of reasons. For general troubleshooting, we suggest you review your logs to locate the potential problem.

Rolling back a JIRA application upgrade

You can roll back JIRA to its previous version if you encounter any issues with the upgrade. Any data changed since the last backup will not be present after rolling back.

Before you begin

Make sure you have the following backups from your previous version:

- The JIRA database (created with the database's native tools)
- The JIRA home directory
- The JIRA installation directory

Rolling back the upgrade

To roll back the upgrade, you simply need to restore the database from the backup and copy the installation and home directories to their original locations, like it was in your previous JIRA setup. Once you do it and restart JIRA, your old environment will be restored.

1. Stop the upgrade or the upgraded JIRA instance.
2. Use your database tools to restore the JIRA database from the backup.
3. Restore the JIRA **installation directory** to its original location. In the case of a manual upgrade, the original installation directory should be intact.
4. Restore the JIRA **home directory** to its original location.
5. Start JIRA by running the `start-jira.sh` or `start-jira.bat` file in the `bin` subdirectory of your restored JIRA application installation directory.

▼ [Have you installed JIRA as a service on Windows?](#)

If that's the case, you also need to restart the `Atlassian JIRA` service from the Control Panel. No need to create a new one, since the JIRA service entry is retained even if there's an error during the upgrade to facilitate the rollback.

Well done! You've rolled back JIRA to its previous version.

Establishing staging server environments for JIRA applications

When you upgrade JIRA, we strongly recommend performing the upgrade in a test environment before upgrading your production site. In this guide we'll refer to this test environment as *staging*.

Most JIRA licenses include a free developer license for use in a staging environment. See [How to get a JIRA Developer license](#) to find out how to access your license.

On this page:

- [Architecture strategy](#)
- [Create a staging environment](#)
 - [1. Replicate your environment](#)
 - [2. Replicate your database](#)
 - [3. Replicate JIRA](#)
 - [Additional steps for Data Center](#)
 - [4. \(Optional\) Replicate external user management](#)
 - [5. Modify application links](#)
- [Upgrade your staging environment](#)

Architecture strategy

This page describes how to create a single staging environment for testing your upgrade. However, if Atlassian applications are critical systems for your organization, we recommend a 3-tier architecture that additionally includes a development environment. With this architecture, you have more flexibility when it comes to testing new changes, and can be sure that these changes are safe when you need to roll them out into production. A 3-tier architecture includes the following environments:

- **Production:** your live instance, expecting minimal downtime and well tested changes.
- **Staging:** a pre-production environment, where the systems administration team can establish exact procedures prior to rollout.
- **Development:** a free-for-all environment where users can play with cutting-edge or risky changes.

The procedure described below can be applied both to creating a staging and development environment. Both of them are identical copies of your production, although used in a different way.

Create a staging environment

1. *Replicate your environment*

Your staging environment should closely replicate your real-live environment (production), including any reverse proxies, SSL configuration, or load balancer (for Data Center). You may decide to use a different physical server or a virtualized solution. The main thing is to make sure it is an appropriate replica of your production environment.

For the purposes of these instructions, we assume your staging environment is physically separate from your production environment, and has the same operating system (and Java version if you've installed JIRA manually).

2. Replicate your database

To replicate your database:

1. Back up your production database. Refer to the documentation for your database for more info on the best way to do this.
2. Install your database on the staging server and restore the backup.

The steps for restoring your database backup will differ depending on your chosen database and backup tool. Make sure:

- Your new staging database has a **different** name from your production database.
- Your staging database user account has the **same** username and password as your production database user account.
- Character encoding and other configurations are the same as your production database (for example character encoding should be Unicode UTF-8 (or AL32UTF8 for Oracle databases).

3. Replicate JIRA

To replicate JIRA, make a copy of your JIRA installation and point it to your staging database. These instructions are for JIRA Server (for Data Center there are some [additional steps](#) before you start JIRA).

1. Copy your entire **production installation directory** to your staging server.
2. Copy your entire **production home directory** to your staging server.
3. Edit `<installation-directory>/atlassian-jira/WEB-INF/classes/jira-application.properties` to point to your staging home directory.
4. Edit `<home-directory>/dbconfig.xml` or `<installation-directory>/server.xml` (older versions) to point to your staging database.

This is extremely important! Make sure your staging environment is not pointing to your production database.

5. Start JIRA with the following [System Properties](#) to make sure your staging site does not send or receive notifications and emails. For more info about disabling email, see [Disable email sending/receiving](#).

```
-Datlassian.notifications.disabled=true
-Datlassian.mail.sendsdisabled=true
-Datlassian.mail.fetchdisabled=true
-Datlassian.mail.popdisabled=true
```

▼ What if I need to test email notifications?

If that's the case, you can keep email notifications enabled and [prepare a development server's mail configuration](#).

6. Head to `http://localhost:<port>` and log in to JIRA on your staging server.

7. Go to



> **System > General Configuration**, and change the **base URL** of your staging site (for example `mysite.staging.com`).

8. Go to



> **Applications > Versions and licenses**, and apply your development license. To update the license, click the edit icon next to it.

9. Go to



> **System** > **System info**, and check that JIRA is correctly pointing to your staging database, and staging home directory.

10. Go to



> **System** > **Look and feel**, and change the colors of the staging instance to make it different from the production instance. That's a small change, but it might help you avoid big mistakes.

Additional steps for Data Center

If you have JIRA Data Center, the process is much the same as for JIRA Server described above. You will copy each local home and installation directory to each staging node, and then:

1. Copy the **production shared home directory** to the staging server.
2. Edit `<local-home-directory>/cluster.properties` to point to your staging shared home directory. This change **must** be made on every staging node.

Changes to the `<installation-directory>/jira-atlassian/WEB-INF/classes/jira-application.properties` and `<local-home-directory>/cluster.properties` must be made on **every** staging node.

When it comes time to start JIRA, start one node at a time, as usual.

4. (Optional) Replicate external user management

If you're managing users in Crowd or an external LDAP directory you can:

- replicate Crowd or your external directory in your staging environment and point your JIRA staging site to your staging external directory (recommended).
- provide your staging server with network or local access to the same hosts as your production server.

5. Modify application links

If you have application links between JIRA and other Atlassian applications, you should change the server ID on each staging application. See [How to change the server ID of Confluence](#) and [Changing Server ID for Test Installations](#) for JIRA.

If you don't change the server ID and update your application links there is a chance that when you create a new application link in production it will point to your staging server instead.

Upgrade your staging environment

Once you have created your staging environment, you can upgrade it in the same way you would your production environment.

Make a note of how long the upgrade takes, as this information will help you plan your production system outage and communicate with your users.

You can also use your staging environment to test any customizations or essential add-ons in your site.

Migrating JIRA applications to another server

This document describes how to migrate/upgrade to JIRA applications on different server hardware, or in a different server environment that entails one or more of the following:

- a new operating system that will run JIRA applications,
- new locations for storing your index and/or attachments, or

- a new database or database system that will store JIRA application data.

- Migrating JIRA to another server
 - 1. Back up
 - 2. Download JIRA
 - 3. Extract the file and upgrade JIRA
 - 4. Start JIRA and connect it to the database
- After the migration
 - 5. Re-apply any modifications
- Troubleshooting

Migrating JIRA to another server

To migrate JIRA to a new server or location, you'll need to install a new JIRA instance. Once you've completed the installation, you'll migrate your existing data between the databases, and then move your home directory and all existing customizations.

1. Back up

1. Create an XML backup of your database.

▼ [Show me how to do this...](#)

You can create an XML backup in the JIRA UI. To keep the database consistent, you should stop your users from making any changes.

a. [Stop users from updating JIRA data.](#)

b. [Perform an XML backup.](#)

2. Back up your **installation directory** and **home directory**.

▼ [Where is my home directory?](#)

You can find the location of your home directory in the `<installation-directory>/atlassian-jira/WEB-INF/classes/jira-application.properties` file.

The default paths are:

Windows: `C:\Program Files\Atlassian\Application Data\JIRA`

Linux: `/var/atlassian/application-data/jira`

3. As soon as the backup is complete, delete the `<home-directory>/dbconfig.xml` file from the copy of your home directory that you'll be using for the new installation. This will allow you to connect JIRA to a new database.

2. Download JIRA

Download the zip or tar.gz archive – [JIRA Core](#) or [JIRA Software](#).

3. Extract the file and upgrade JIRA

1. Extract (unzip) the files to a new directory.
2. Edit the `<installation-directory>\atlassian-jira\WEB-INF\classes\jira-application.properties` file to point to your **existing** JIRA home directory. Make sure that you deleted the `dbconfig.xml` file, otherwise JIRA will try to connect to your existing database.
3. Copy your jdbc driver jar file from your existing JIRA installation directory to `jira-atlassian/WEB-INF/lib` in your **new installation directory**.
4. There are some additional steps you may need to take if:

- you use **Crowd** for user management
 - ▼ [Click here to expand...](#)

If you are using Crowd for user management, reapply the modifications from the following files from your existing installation directory to the new files. Do not copy the files as they may be different in the new version of JIRA.

 - <Installation-Directory>/atlassian-jira/WEB-INF/classes/ouser.xml (4.3.0 and earlier)
 - <Installation-Directory>/atlassian-jira/WEB-INF/classes/crowd.properties
 - <Installation-Directory>/atlassian-jira/WEB-INF/classes/seraph-config.xml

4. Start JIRA and connect it to the database

1. Start JIRA to launch the setup wizard. Go to <installation-directory>/bin, and run the start-jira.sh / start-jira.bat file.
2. Access JIRA through your browser, and choose **I'll set it up myself** to get access to more setup options.
3. Select **My own database**, and provide details of a new, empty database.
4. Click **Import your data**, and select the file with your XML backup.

Avoid passing through a proxy when performing an XML restore, especially if your JIRA instance is very large. Using a proxy may cause timeout errors.

5. Take a quick look around your JIRA site to confirm that your projects and issues are present and everything looks normal. You should see the new JIRA version number in the page footer.

After the migration

5. Re-apply any modifications

If you have customized JIRA (such as an SSL configuration in the `server.xml` file, or additional memory allocation in the `setenv.sh` / `setenv.bat` file), you'll need to reapply the customizations to the relevant files in the newly upgraded JIRA installation directory.

▼ [Show me how to do this...](#)

Any other configurations or customizations (including any other modifications in the `server.xml` file) are **not migrated** during the upgrade and need to be reapplied manually.

1. Stop your upgraded JIRA instance.
2. Reapply the customizations to the relevant files in the upgraded JIRA installation directory.
3. Restart the upgraded JIRA instance.

We **strongly recommend** you test your customizations in a test instance prior to upgrading your production instance as changes may have been made to JIRA that make your customizations unusable.

▼ [Show me the list of important files...](#)

If you're unsure which files you might have modified, take a look at the list of [important files in the JIRA installation directory](#).

Troubleshooting

Did something go wrong?

Check the resources on our [Support](#) page, or raise an issue so we can help you resolve your problem.

Your existing JIRA installation should be intact. You can restart it and resume your work.

Migrating from JIRA Cloud to Server applications

This page is for people who are currently using JIRA Cloud, and wish to move to JIRA Server (a JIRA installation hosted on your own servers).

JIRA Cloud is typically ahead of JIRA Server, which means that some features may not be available after you've moved to JIRA Server.

If you want to move a project, not your entire site, see [Restoring a project from backup](#).

On this page:

- [Limitations](#)
- [Migrating to JIRA Server](#)
 - 1. Back up
 - 2. Download the installer
 - 3. Install JIRA Server
 - 4. Import your data from JIRA Cloud into JIRA Server
- [After the migration](#)
 - 5. Change the admin password
 - 6. Check which plugins you have in Cloud
 - 7. Install your plugins in JIRA Server
 - 8. Reset the passwords for your users

Limitations

JIRA Cloud applications are regularly updated with the latest features and improvements, which means they are a later version than the latest downloadable version of JIRA applications. If you want to migrate from JIRA Cloud applications to JIRA Server applications, be aware of the following limitations.

▼ [Click here to expand...](#)

Feature loss

JIRA Cloud typically contains features that are not yet released in the latest version of JIRA Server.

Password reset

Your JIRA Cloud users will need to reset their passwords before they can log in to the new JIRA Server instance.

JIRA application licenses

Your Atlassian Cloud license can't be used in an instance installed from JIRA Server applications. You'll need to generate a new JIRA Server application license from <https://my.atlassian.com>.

You can reuse your licenses for plugins in your instance installed from JIRA Server applications. The licenses for Atlassian plugins and Gliffy for JIRA applications can be viewed on <https://my.atlassian.com>. For all other third-party plugins, contact the third-party vendor for a license.

Migrating other Cloud applications

The instructions on this page only apply to JIRA applications. If you are migrating other Cloud applications (e.g. Confluence Cloud to an instance installed from Confluence Server), see this page: [Backing up and exporting data](#).

Note, if you are migrating JIRA Cloud applications *and* other applications (e.g. Confluence Cloud) to an instance hosted on your own servers, you'll lose integration features that are native to Cloud. These can be re-enabled by configuring application links between your applications. See [Using AppLinks to link to other applications](#) for instructions. Contact support if you need assistance.

Migrating to JIRA Server

1. Back up

Back up your JIRA Cloud application data.

1. Log in to JIRA Cloud as an administrator.
2. Create an XML backup. For more info, see [Exporting issues](#). Note that the export process will strip your cloud application and plugin licenses out of the XML, so the licenses will remain available in your

JIRA Cloud site but not in your Server instance.

2. Download the installer

Download the installer for your operating system – [JIRA Core](#), [JIRA Software](#), or [JIRA Service Desk](#).

3. Install JIRA Server

Install your JIRA applications. For detailed instructions, refer to [Installing JIRA applications](#).

Note that during the installation, you will be asked if you have existing data. This is where you can import your XML backup, as described in the next step.

4. Import your data from JIRA Cloud into JIRA Server

In step 2 of the setup wizard (Application Properties), you'll be asked whether you have existing data. Click **Import your existing data**, and follow instructions to import your XML backup.

When importing, you might see a warning that you're importing data from an earlier JIRA version. You can ignore it and continue with the import.

If your backup is 2 GB or more, import the attachments separately

For large backups, we recommend that the attachments are imported separately. To do this:

1. Unzip the backup file.
2. Compress the `activeobjects.xml` and the `entities.xml` files only.
3. Import the compressed file in the setup wizard, as described above.
4. Copy the contents of the `attachments` directory into the `<home-directory>/data/attachments` directory for JIRA Server.

After the migration

5. Change the admin password

1. Log in to your new JIRA application, using the following credentials:
 - Username: `sysadmin`
 - Password: `sysadmin`
2. Change the password immediately after logging in.

6. Check which plugins you have in Cloud

Any plugins that you are currently using with JIRA Cloud application will need to be installed in your JIRA application installation. For example, Gliffy, Tempo, etc.

Choose



> **Add-ons**. The 'Find add-ons' screen shows add-ons available via the [Atlassian Marketplace](#). Choose **Manage add-ons** to view the plugins currently installed on your JIRA applications. Choose **Manage add-ons** and note the plugins listed under the **User-installed Plugins** section. You will need to note the plugin names and versions.

7. Install your plugins in JIRA Server

For each plugin that you noted in the previous step, install it in your JIRA applications. Make sure to install the latest version of the plugin. Atlassian does not provide support for data that is downgraded as a result of installing an older version of a plugin.

See [Managing Add-ons](#) for instructions on how to install a plugin. You will need to manually add the plugin license keys.

8. Reset the passwords for your users

After you finish migrating from JIRA Cloud to JIRA Server, you must reset the passwords for your users. To do this, you can either:

- notify all users that they need to click **Forgot password** to reset their password. [Learn how to send an email to all JIRA users.](#)
- log in as **sysadmin** and reset everyone's password.

Well done! You've migrated your JIRA Cloud site into a JIRA Server instance.

Getting started with JIRA Data Center on AWS

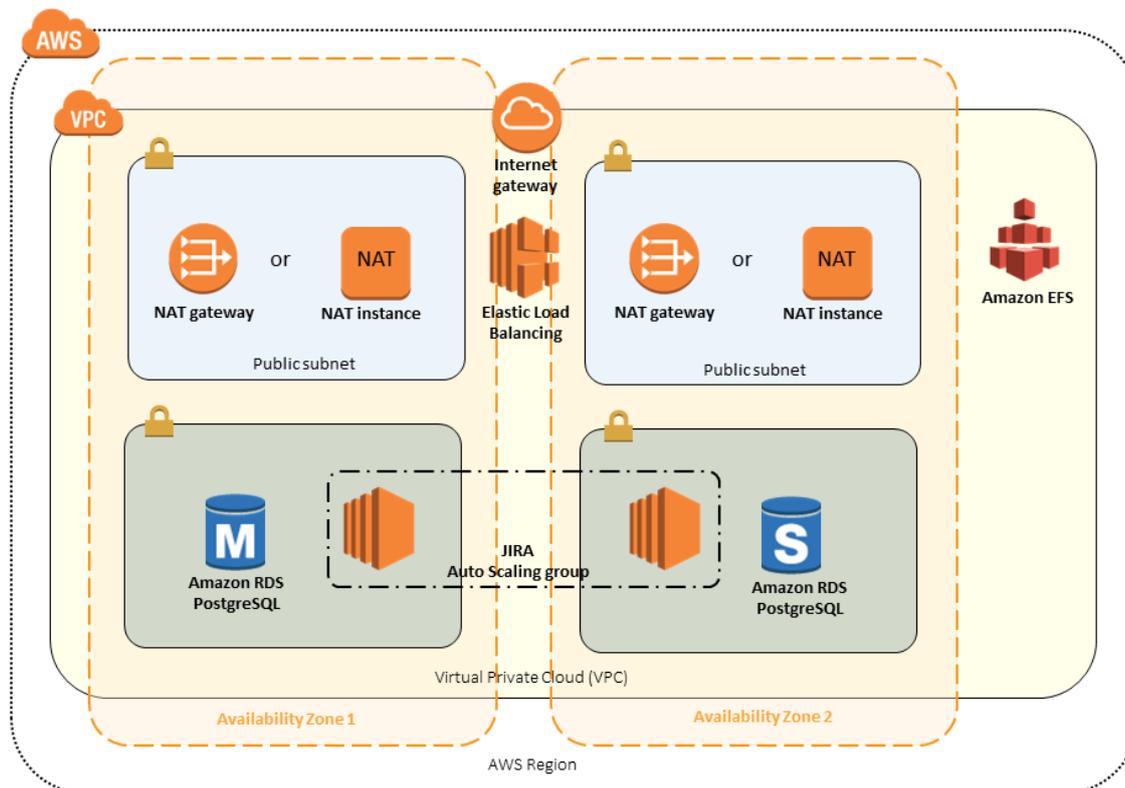
JIRA Data Center is an excellent fit for the Amazon Web Services (AWS) environment. Not only does AWS allow you to scale your deployment elastically by resizing and quickly launching additional nodes, it also provides a number of managed services that work out of the box with JIRA Data Center instances and handle all their configuration and maintenance automatically.

Interested in learning more about what Jira Data Center provides? [Click here for an overview.](#)

Deploying JIRA Data Center using the AWS Quick Start

The simplest way to deploy your entire Data Center cluster in AWS is by using the Quick Start. The Quick Start launches, configures, and runs the AWS compute, network, storage, and other services required to deploy a specific workload on AWS, using AWS best practices for security and availability.

Here's an overview of the architecture for the JIRA Data Center Quick Start:



The deployment consists of the following components:

- **Instances/nodes.** One or more Amazon Elastic Cloud (EC2) instances as cluster nodes, running JIRA.
- **Load balancer.** An Amazon Elastic Load Balancer (ELB), which works both as load balancer and SSL-terminating reverse proxy.
- **NFS server.** An EC2 instance as dedicated NFS server, with an attached Elastic Block Store (EBS) volume for the shared home directory.
- **Database.** An Amazon Relational Database (RDS) instance as the shared database.
- An Amazon Elasticsearch Service domain for code and repository search.

How does it work?

To deploy JIRA Data Center, the AWS Quick Start uses a CloudFormation template that was created by the JIRA team. Think of it as a script that controls what is deployed, and in what way. It's the template that defines how big your database is, and how many JIRA instances are installed in your environment, also specifying such settings, as the DNS name, startup parameters (Catalina), or passwords. Thanks to that, all components required to run JIRA can be created, and later updated or removed, as a single "stack".

Since we wanted to create a quick implementation, the Quick Start doesn't allow the same level of customization as in the case of manual installation. It should be treated as a reference implementation that shows how JIRA Data Center can be used with AWS, and be either used as-is, or serve as an example for [creating your own template](#).

What's in the Quick Start?

Several components, such as database and JIRA version, are fixed and can't be changed. The table below shows details of the environment deployed with the Quick Start:

JIRA product	JIRA Software or JIRA Service Desk.		
JIRA version	Always the latest version, no version picker. After deploying the stack, you can, however, upgrade your JIRA instance to any version you want.		
Operating system	Amazon Linux.		
Database	Amazon RDS for PostgreSQL.		
Cluster nodes	One or more EC2 instances running JIRA.		
Load balancer & reverse proxy	Amazon Elastic Load Balancer (ELB), with Listeners configured as follows:		
	Load balancer protocol	Load balancer port	Instance protocol
	HTTP	80	HTTP
	TCP	7999	TCP
NFS server	An EC2 instance as dedicated NFS server, with an attached Elastic Block Store (EBS) volume for the shared home directory.		
EFS server	An EC2 instance as dedicated EFS server for the cluster, with an EBS volume storing the JIRA shared home directory. This contains all of JIRA's attachments and other data. You need to ensure you set up your JIRA AWS deployment in a region that offers EFS support).		

Configurable parameters

When deploying JIRA Data Center using Quick Start, you'll be asked to configure some parameters, such as database size, custom DNS, or the number of JIRA instances in your environment. For a full list and recommendations, see [Configuring JIRA Data Center on AWS](#).

Getting started

You'll need two things to get started – the Quick Start itself, and the docs that will guide you through it.

- [Quick Start](#)
- [JIRA AWS documentation](#)

Configuring JIRA Data Center on AWS

While deploying JIRA Data Center on AWS using the Quick Start, you'll be asked to configure several parameters related to JIRA setup.

Configurable parameters

Here's a list of parameters that you can configure.

JIRA product	JIRA Core, JIRA Software, or JIRA Service Desk.
Min. and max. number of cluster nodes	The number of JIRA instances in your environment.
JIRA instance type	The size of your JIRA environment, including vCPU, memory, and storage. See: JIRA instance sizes .
Database instance class	The size of your database, including vCPU, memory, and throughput. See: Database sizes .
JIRA and database passwords	Password for the JIRA and database users.
Database storage	Database allocated storage size, in gigabytes.
HA Database across Availability Zones	Database high availability, enabled or disabled.
Existing DNS name (optional)	Custom DNS name, e.g. <code>jira.atlassian.com</code> . See: Setting custom DNS name .
Catalina Options	JIRA startup parameters.
SSL certificate ARN (optional)	Amazon Resource Name (ARN) of your SSL certificate. It depends on a certificate that you want to use: <ul style="list-style-type: none"> • If you want to use your own certificate that you generated outside of Amazon (IAM certificate), you need to first import it to AWS Certificate Manager. After a successful import, you'll receive the ARN. • If you want to create a certificate with AWS Certificate Manager (ACM certificate), you will receive the ARN after it's successfully created.

JIRA instance sizes

You can choose the following [Amazon EC2 instances](#) for your JIRA Data Center instance.

Model	vCPU	Mem (GiB)	Instance Store (GB)	EBS optimizations available	Dedicated EBS Throughput (Mbps)
c3.xlarge (default)	4	7.5	2 x 40 SSD	Yes	-
c3.2xlarge	8	15	2 x 80 SSD	Yes	-

c3.4xlarge	16	30	2 x 160 SSD	Yes	-
c3.8xlarge	32	60	2 x 320 SSD	-	-
c4.large	2	3.75	-	Yes	500
c4.xlarge	4	7.5	-	Yes	750
c4.2xlarge	8	15	-	Yes	1,000
c4.4xlarge	16	30	-	Yes	2,000
c4.8xlarge	36	60	-	Yes	4,000
i2.xlarge	4	30.5	1 x 800 SSD	Yes	-
i2.2xlarge	8	61	2 x 800 SSD	Yes	-
i2.4xlarge	16	122	4 x 800 SSD	Yes	-
i2.8xlarge	32	244	8 x 800 SSD	-	-
m3.large	2	7.5	1 x 32 SSD	-	-
m3.xlarge	4	15	2 x 40 SSD	Yes	-
m3.2xlarge	8	30	2 x 80 SSD	Yes	-
m4.large	2	8	-	Yes	450
m4.xlarge	4	16	-	Yes	750
m4.2xlarge	8	32	-	Yes	1,000
m4.4xlarge	16	64	-	Yes	2,000
m4.10xlarge	40	160	-	Yes	4,000
r3.large	2	15.25	1 x 32 SSD	-	-
r3.xlarge	4	30.5	1 x 80 SSD	Yes	-
r3.2xlarge	8	61	1 x 160 SSD	Yes	-
r3.4xlarge	16	122	1 x 320 SSD	Yes	-
r3.8xlarge	32	244	2 x 320 SSD	-	-

Recommended size of the JIRA Data Center instance:

Active Users	EC2 instance type	Recommended number of nodes
0 – 250	c3.xlarge	1-2*
250 – 500	c3.xlarge	1-2*
500 – 1000	c3.2xlarge	2
1000 – massive scale	c3.4xlarge+	3+

* For high-availability, we recommend deploying 2 or more cluster nodes as a minimum.

Database sizes

You can choose the following [DB instance classes](#):

Model	vCPU	Memory (GB)	EBS Throughput (Mbps)
db.m4.large (default)	2	8	450
db.m4.xlarge	4	16	750
db.m4.2xlarge	8	32	1,000
db.m4.4xlarge	16	64	2,000
db.m4.10xlarge	40	160	4,000
db.r3.large	2	15.25	-
db.r3.xlarge	4	30.5	-
db.r3.2xlarge	8	61	-
db.r3.4xlarge	16	122	-
db.r3.8xlarge	32	244	-
db.t2.medium	2	4	-
db.t2.large	2	8	-

Administering JIRA Data Center on AWS

While working with your JIRA Data Center on AWS, you can expand your environment by adding additional nodes, upgrade the existing JIRA instances, or connect to them over SSH.

Setting custom DNS name

When deploying JIRA Data Center on AWS, you get a default domain name that points to the Amazon's load balancer. You'll be using it to access JIRA. This domain name depends on the load balancer's name and the AWS region, but in general it looks like this: `my-loadbalancer-1234567890.us-west-2.elb.amazonaws.com`. You can change it to something more familiar, e.g. `jira.atlassian.com`, by entering your own domain name in the **Existing DNS (optional)** parameter in the Quick Start. You'll need a domain name to do this, if you don't have one already, you can register it [here](#).

To set the custom DNS name:

1. When deploying JIRA with the Quick Start, enter your domain name (FQDN) in the **Existing DNS (optional)** parameter. It'll be saved in the `proxyName` parameter in Apache Tomcat, which is a web server used by JIRA. All nodes will be using this domain name.
2. After the deployment, when you know the address of the Amazon's load balancer, associate it with your domain name. To do this, you'll need to use your DNS service to create a CNAME record where you enter the source and target URLs, creating an alias. See [Associating your custom domain name with your load balancer name](#).

If you have already deployed JIRA, you can still change the parameters that are used by your stack, be it the instance type or the domain name. See [Changing resource properties](#).

Scaling up and down

To change the number of nodes in the cluster:

1. Set the desired range, so the minimum and maximum number of nodes that can be started in your stack.
 - a. In the AWS console, go to **Services > CloudFormation**, select the stack, and click **Update Stack**.

- b. Change the **Minimum number of cluster nodes** and **Maximum number of cluster nodes** parameters as desired. After this step, the number of nodes (controlled by the Auto Scaling group) will be set to the minimum number.
2. Change the size of the Auto Scaling group to increase the number of available nodes.
 - a. In the AWS console, under **Auto Scaling**, select **Auto Scaling Groups**.
 - b. Select your Auto Scaling group.
 - c. On the **Details** tab, choose **Edit**.
 - d. For **Desired**, increase the desired capacity by one. For example, if the current value is 1, type 2.

The desired capacity must be less than or equal to the maximum size of the group. If your new value for Desired is greater than Max, update Max.

 - e. Click **Save**.

Connecting to your nodes over SSH

Sooner or later you will need to SSH to your JIRA cluster node(s) and file server to perform configuration or maintenance tasks. Note that you must keep your SSH private key file (the PEM file you downloaded from Amazon and specified as the **Key Name** parameter) in a safe place. This is the key to all the nodes in your instance, and if you lose it you may find yourself "locked out". See [Administering JIRA Data Center on AWS](#) for more information.

Note: `JiraDataCenter.template` deploys all EC2 instances in the Subnets specified by the **Internal subnets** parameter. If you have specified **Internal subnets** that are completely unreachable from outside, then you may need to launch an EC2 instance with SSH running and accessible in one of the the **External subnets**, and use this as a "jump box" to SSH to any instances in your **Internal subnets**. That is, you SSH first to your "jump box", and from there to any instance deployed in the **Internal subnets**.

When connecting to your instance over SSH, use `ec2-user` as the user name, for example:

```
ssh -i keyfile.pem ec2-user@ec2-xx-xxx-xxx-xxx.compute-1.amazonaws.com
```

The `ec2-user` has `sudo` access. SSH access as `root` is not allowed.

Backing up

We recommend you use the AWS native backup facility, which utilizes snapshots to back up your JIRA Data Center.

Migrating your existing instance into AWS

To migrate an existing instance into AWS:

1. Migrate its database to PostgreSQL (if not already).
2. Take a backup of your existing home directory and database.
3. Copy the backup file to your file server EC2 instance.
4. Unpack the backup file under `/media/at1/jira/shared` of your file server.
5. Restore the PostgreSQL database dump contained in the backup file to your RDS instance with `pg_restore`.

Upgrading JIRA Data Center on AWS

Upgrading JIRA Data Center on AWS is similar to a regular upgrade on a local environment. You will switch the installer to a different version, and then deploy new EC2 instances in the version you've chosen, also achieving zero downtime.

Although this guide is written to help you upgrade your JIRA stack, you can use it to install any version you want.

Before you begin

To upgrade your JIRA Data Center, you'll need the following:

- private key (.pem) file created while deploying the Quick Start
- SSH client

Upgrading JIRA instances

Mount the EFS storage

The current installer, used by the Quick Start to deploy your instances, is in the EFS storage. You can't access it directly, because the Quick Start deploys all instances into private subnets. To work around it, you'll create a new instance in a public subnet, and use it to mount the EFS storage.

1. Create a new instance in a public subnet. Let's refer to it as *jumpbox*.
 - ▼ [Show me how to do this...](#)
 - a. In the AWS console, go to **Services > EC2** (if you can't find it, type *EC2* in the search bar).
 - b. In the menu on the left, select **Instances**, and then click **Launch Instance**.
 - c. Select the operating system, e.g. Amazon Linux, which is a default OS used by the Quick Start.
 - d. Select the size. We'd recommend `t2.nano`, since you don't need much capacity on this instance.
 - e. In the **Configure Instance Details** page, make sure you get these settings right:
 - Network: Choose your stack's VPC.
 - Subnet: Choose any public subnet. Your stack should have two private, and two public subnets.
 - Auto-assign Public IP: Enable.
 - f. Add storage, and click **Next**.
 - g. In the **Configure Security Group** page, select an existing security group, and choose the security group of your stack from the list.
 - h. Click **Review and Launch**.
 - i. Select the key pair that you created while deploying the stack, and launch the instance.

The new instance will be started in your stack, which might take several minutes. You can later use it to connect to all instances in the stack, but we'll just need it to mount the EFS.

2. Connect to the new instance over SSH.
 - ▼ [Show me how to do this...](#)
 - a. In the AWS console, go to **Services > EC2 > Instances**, and select the jumpbox that you just launched.
 - b. From the Description tab, copy the Public DNS. You'll need it to connect to this instance.
 - c. Get your private key (.pem) file ready.
 - d. Open the SSH client, and connect to the jumpbox with this command:

```
ssh -i privatekey.pem ec2-user@public_dns
```

For example:

```
ssh -i atlassian.pem
ec2-user@ec2-35-164-58-207.us-west-2.compute.amazonaws.com
```

3. Mount the EFS storage on this instance, so you can access the JIRA installer.
 - ▼ [Show me how to do this...](#)
 - a. In the AWS console, go to **Services > EFS**, and select the EFS used by your stack.
 - b. Click **Amazon EC2 mount instructions**. It will show you the commands needed to mount the file server. You're interested in "Mounting your file system".
 - c. Connect to the jumpbox over SSH.
 - d. Run the `mount` command, as described in "Mounting your file system".

- e. After mounting the EFS storage, you can browse the files it contains. These should include the **JIRA installer** and the **version file**.

With access to the JIRA files, you can update the installer and the related version file.

Upgrade your JIRA instances

Having access to the JIRA files, you can update the installer and the related version file. Once you complete these steps, you can terminate your instances one by one, and start the new ones, in the new version.

1. Update the installer and the version file.
 - a. [Download the JIRA installer](#).
 - b. Access your EFS storage. If you missed that part, see [Mount the EFS storage](#).
 - c. Switch the current installer to the new one. Make sure the file name keeps the same pattern.
 - d. Edit the version file in the same directory, and update the version.
2. Put JIRA into upgrade mode, so that it accepts your instances being in different versions during the upgrade.
 - a. Log in to your JIRA instance on AWS.
 - b. Go to

> Applications > JIRA upgrades, and click **Put JIRA into upgrade mode**.

For more information about the upgrade mode, see [Managing zero downtime upgrades](#).

3. Terminate your instances in AWS one at a time.
 - a. In the AWS console, go to **Services > EC2 > Instances**.
 - b. Right-click an instance, and select **Instance State > Terminate**. A new instance will be automatically started for each terminated instance. To achieve zero downtime, wait until the new instance is started before you terminate the next one, it might take several minutes.
4. After all instances are upgraded, finalize the upgrade.
 - a. In JIRA, go to

> Applications > JIRA upgrades, and click **Finalize upgrade** (in some cases, it might be **Run upgrade tasks** instead.)

Well done! You upgraded your JIRA instance to a new version. You can see the current version by going to

> Applications > Versions and licenses.

Getting started with JIRA Data Center on Azure

JIRA Software Data Center is an excellent fit for the Microsoft Azure environment. Azure provides a number of managed services that work out of the box with JIRA Software Data Center instances, and handle all their configuration and maintenance automatically. You can also choose the number and size of your nodes to suit your organization's needs, and if you need to scale up or down, Azure allows you to do this.

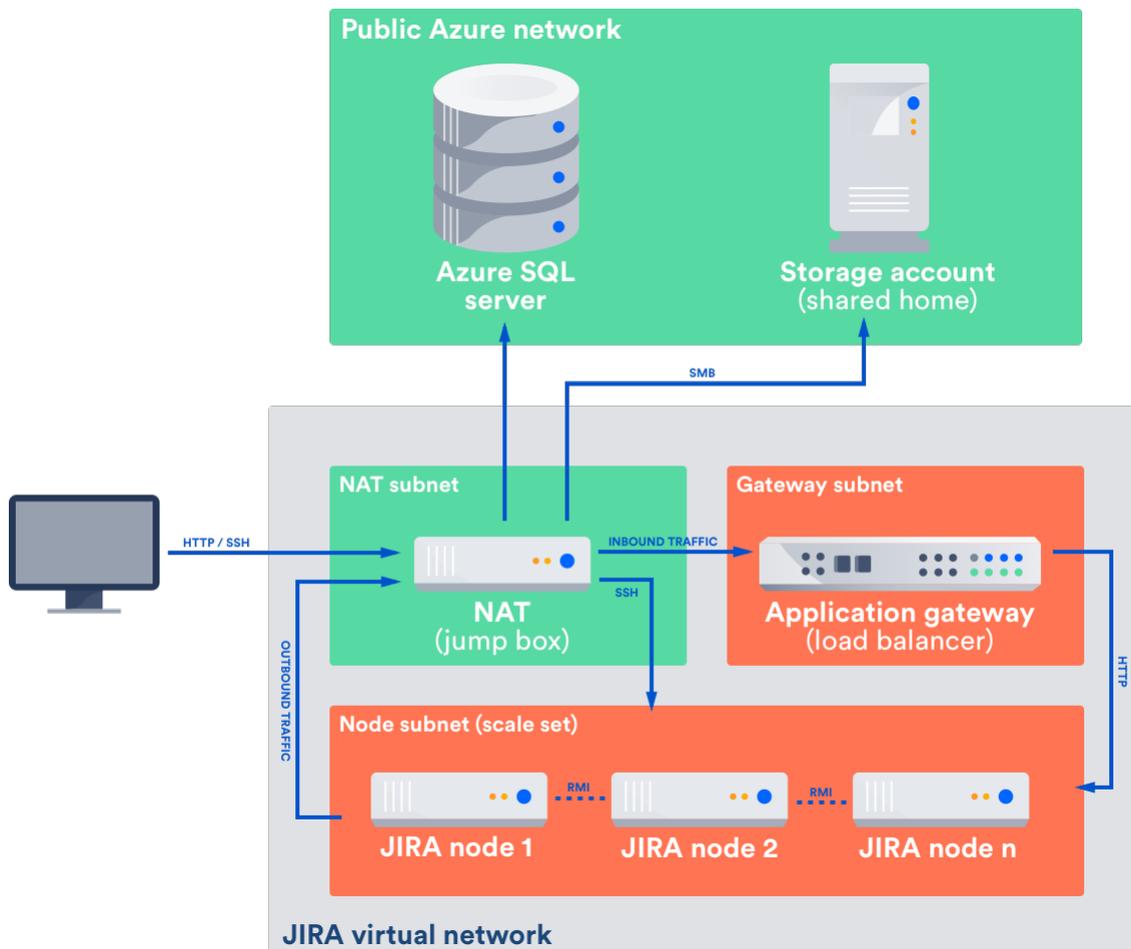
We've set up a reference template that lets you deploy JIRA Software Data Center in Microsoft Azure, and you can then configure it depending on your organization's Azure best practices. We strongly recommend you set up user management, central logging storage, a backup strategy, and monitoring, just as you would for a JIRA Software Data Center installation running on your own hardware.

On this page:

- [How it works](#)
- [Deploying JIRA Software Data Center to Azure](#)
- [Configurable parameters](#)
- [JIRA node sizes](#)

How it works

Here's an architectural overview of what you'll get when deploying JIRA Software Data Center using the template:



The deployment consists of the following components:

- One or more Azure standard VM instances as cluster nodes, running JIRA Software, in a scale set
- Azure SQL server and database
- a storage account for the shared home directory, which contains attachments and other files accessible to all JIRA nodes
- a NAT box (or jumpbox)
- an application gateway

Azure SQL instances can't be created in a virtual private network/subnet. To ensure security, the SQL Server firewall has been configured to only allow traffic from the private network that the cluster nodes reside in. The SQL Server firewall rules require a public IP address upfront during deployment, and as the application gateway's public IP address isn't known at this stage of the deployment, we use the public IP address of NAT box (jumpbox). This network topology means that all traffic from the cluster to the SQL Server is routed through the NAT box, and all public traffic to the cluster is also routed through the NAT box, including all SSH traffic and L4 traffic to the application gateway.

The application gateway also acts as a load balancer for your scale set of JIRA Software nodes.

We use a storage account for JIRA Software's shared home directory. As with the Azure SQL Server, this service exists outside the virtual network. It's mounted on each JIRA Software node, and it's treated as any other file would be.

Deploying JIRA Software Data Center to Azure

Once you have all the required details, you'll need to:

1. Sign in to your Microsoft Azure account, and access the [Microsoft Azure marketplace](#).
2. Search for the JIRA Software Data Center product, and select it.
3. Complete the wizard (you'll need the details [listed below](#)), and confirm you're happy with the details you've entered on the **Summary** page.
4. Purchase your subscription.

JIRA Software will start up, this could take a few minutes. Once it's up and running, you can access JIRA Software via the public IP address provided on your [Azure portal](#), using the details you provided for the JIRA Software admin account during the deployment. From here, you can configure JIRA Software to suit your requirements.

Configurable parameters

Here's a list of parameters that you can configure. Parameters marked * are optional

CNAME*	The Canonical Name record (CNAME) for your organization. If you don't provide one, Azure will generate a random sub domain for your instance.
Subscription	Your Microsoft Azure subscription type.
Resource group	If you have an existing resource group, you can use it, or create a new one.
Location	This is the region where Azure will house your deployment.
JIRA admin credentials	You need to provide a name and password for the initial JIRA admin on your instance.
Number and size of nodes	This provides the initial number of nodes and their size, and can be reconfigured at a later date. For recommendations, see JIRA node sizes .
Database credentials	You need to provide a name and password for the database.
Jumpbox credentials	You need to provide a name and SSH public key for the jumpbox.
JIRA node credentials	You need to provide a name and password for your nodes.

JIRA node sizes

The following sizes are recommended by Atlassian.

Model	vCPU	Mem (GB)	Max cached and temp storage throughput: IOPS/MBps (cache size in GiB)	Max uncached disk throughput: IOPS/MBps
DS2_V2	2	7	8,000 / 64 (86)	6,400 / 64
DS3_V2	4	14	16,000 / 128 (172)	12,800 / 128
DS4_V2	8	28	32,000 / 256 (344)	25,600 / 256
Standard_F4s	4	8	16,000 / 128 (48)	12,800 / 192
Standard_F8s	8	16	32,000 / 256 (96)	25,600 / 384

Administering JIRA Software Data Center on Azure

Once you've deployed JIRA Software Data Center to Azure using the reference deployment template, administering the application is similar to managing an application on your own hardware, with the exception that you'll need to go via the jumpbox to access your nodes and shared home directory. To access your jumpbox and nodes you'll need:

- the SSH credentials you provided during setup,
- the JIRA node credentials you provided during setup
- the public DNS name or IP address of your jumpbox (you can obtain this through the Azure portal via **Me**

- **nu > Resource groups > <your resource group> > jiranat**), and
- the node IP addresses, listed against the `jiracluster (instance n)` row in **Connected devices**. (You can obtain this through the Azure portal via **Menu > Resource groups > <your resource group> > jiranat**).

Connecting to your Azure jumpbox and nodes over SSH

Access the jumpbox via a terminal or command line using:

```
$ ssh JUMPBOX_USERNAME@DNS_NAME_OR_IP_ADDRESS
```

Once you've accessed the jumpbox, we can jump to any of the nodes in the cluster, using:

```
$ ssh NODE_USERNAME@NODE_IP_ADDRESS
```

You'll then be asked for your node password - after providing this, you should be connected to the node.

Accessing your `dbconfig.xml` or `server.xml` files

For your Azure deployment, you may need to make changes to your `dbconfig.xml` or `server.xml` files, just as you would for a deployment on your own hardware:

- your shared `server.xml` lives in `/media/atl/jira/shared/server.xml`
- your shared `dbconfig.xml` lives in `/media/atl/jira/shared/dbconfig.xml`

In Azure, these files are copied from the *shared home* to the *local home* when a **new** node joins the cluster.

So, when making changes in `server.xml` or `dbconfig.xml` on existing nodes, it's important to also update the files in the *shared home*, otherwise a new node joining the cluster will be setup with outdated configuration.

These files are only accessible from the existing nodes. The *shared home* is mounted (think of it as a network harddisk) on each node under `/media/atl/jira/shared`.

So from an existing node (when you're logged in through SSH), you can go to `/media/atl/jira/shared`

Backing up

We recommend you use the Azure native backup facility, which utilizes snapshots to back up your JIRA Software Data Center.

Migrating your existing instance into Azure

We recommend you use the standard options available to [migrate data between JIRA instances](#).

Layout and design

The layout and design of your JIRA applications can be customized to an extent, so that they more closely reflect your organization's look and feel, and requirements in relation to user preferences (such as the default language, user preferences and announcement banner).

Search the topics in 'Layout and design':

The following pages provide you with more information on setting up your JIRA applications so you can get the most out of them!

- [Configuring the look and feel of your JIRA applications](#)
- [Configuring an announcement banner](#)

- Configuring the default dashboard
- Choosing a default language
- Configuring the default issue navigator
- Creating links in the application navigator
- Configuring the user default settings

You may also wish to extend JIRA's functionality by installing and/or enabling new plugins. Read the [Managing add-ons](#) page for further information.

Configuring the look and feel of your JIRA applications

As a JIRA administrator, you can customize the look and feel of your JIRA applications to match your company's environment. This page will walk you through the following:

- How to change your logo
- How to show your site title
- How to change the favicon
- How to change JIRA application colors
- How to change the gadget colors
- How to change date and time formats

How to change your logo

The logo appears in the top left corner of every JIRA application page. The height of the logo image must be constrained to 30 pixels. We also recommend you use an image with a width of 57 pixels.

1. Choose



> **System.**

2. In the User interface section, click **Look and feel**.
3. In the Logo section, upload the image for the logo you want to use in your JIRA application. You can also upload from a URL beginning with 'http://' or 'https://'.

How to show your site title

If enabled, your site name will appear next to your logo.

1. Choose



> **System.**

2. In the User interface section, click **Look and feel**.
3. Select the **Show Site Title** checkbox to make your instance name appear next to your logo in the header.

How to change the favicon

The favicon appears typically to the left of your browser's URL field, and on browser tabs displaying a page on your JIRA site. To upload a favicon, make sure it's in PNG format, with dimensions of 32x32 pixels, 71x71 DPI, and with 8-bit color depth.

1. Choose



> **System.**

2. In the User interface section, click **Look and feel**.
3. In the Favicon section, upload the image for the favicon in your JIRA application or upload from a URL beginning with 'http://' or 'https://'.

How to change JIRA application colors

You can use color options to control the appearance of the entire JIRA user interface. The colors you choose for each option can be anything that is valid for both a font tag, and a stylesheet's 'color' attribute.

1. Choose



> **System.**

2. In the User interface section, click **Look and feel**.
3. In the Colors section, modify the color schemes for the different elements of your JIRA instance as needed using the pop-up color chooser, or by specifying your own (eg. '#FFFFFF', 'red').
4. To return to the original color scheme, just clear any values that you've set, or click **Revert** in the row where you made the change.

How to change the gadget colors

You can use any of the color options to change the color of a gadget's frame on the JIRA dashboard.

1. Choose



> **System.**

2. In the User interface section, click **Look and feel**.
3. In the Gadget Colors section, select the color option for the gadget's frame on your JIRA dashboard, and then modify the color as needed using the pop-up color chooser, or by specifying your own (eg. '#FFFFFF', 'red').
4. To return to the original color scheme, just clear any values that you've set, or click **Revert** in the row where you made the change.

Color 1 is the default frame color for newly-added gadgets. The colors you specify for each of the options can be anything that is valid for both a font tag, and a stylesheet's 'color' attribute.

How to change date and time formats

You customize the way times and dates are presented to users throughout the JIRA user interface. When specifying dates and times, they should be based on the [Java SimpleDateFormat](#).

1. Choose



> **System.**

2. In the User interface section, click **Look and feel**.
3. In the Date/Time Formats section, click the value of the element you want to configure, then update the value as necessary.

Issue date/time fields show a relative instead of absolute date/time format. For example, "Yesterday" would appear instead of "20 May 2013 12:00 PM". You can still see the absolute date/time by hovering over the field. The date/time format reverts to absolute after a week.

Here are some further examples of US date/time configurations:

Preferred Date/Time	Value of the <code>jira.date.time.picker.java.format</code> property	Value of the <code>jira.date.time.picker.java.rpt.format</code> property
2010-10-15 08:50	yyyy-MM-dd HH:mm	%Y-%m-%d %H:%M (see ISO 8601 format)
15/Oct/10 8:50 AM	dd/MMM/yy h:mm a	%d/%b/%y %l:%M %p

10/15/10 08:50 AM	MM/dd/yy hh:mm a	%m/%d/%y %I:%M %p
----------------------	------------------	-------------------

Configuring an 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.

If you are using JIRA Server, the banner can be configured to contain HTML text. If you are using JIRA Cloud, you can only use wiki markup in the banner.

⚠ Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- [Configuring an announcement banner](#)
- [Banner visibility mode](#)

Configuring an announcement banner

1. Choose



> **System.**

2. Select **User Interface > Announcement banner** in the System panel below.
3. Enter the required text in the **Announcement** field.
4. Select the required **Visibility Level** for the banner.
5. Click the **Set Banner** button.

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

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**

Configuring the default dashboard

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

JIRA allows Administrators to configure the default dashboard. The gadgets on the default dashboard can be re-ordered, switched between the left and right columns, additional gadgets can be added, and some gadgets can be configured. The layout of the dashboard (e.g. number of columns) can also be configured.

All changes made to the default dashboard will also change the dashboards of all users currently using the default. However, gadgets that users do not have permissions to see will not be displayed to them. For example, the 'Administration' gadget, although it may exist in the default dashboard configuration, will not be visible to non-admin users. [Gadgets](#) are the information boxes on the dashboard. JIRA comes pre-configured with a set of standard dashboard gadgets. It is also possible to develop custom gadgets and plug them into JIRA using its flexible [plugin system](#).

⚠ Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Adding and configuring gadgets on the default dashboard

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. Choose



> System.

2. Select **User Interface > System dashboard** to open the Configure System Dashboard page.
3. On the 'Configure System Dashboard' page, you can do the following:
 - Move a gadget by drag-and-drop.
 - Re-configure existing gadgets.
 - Choose a different layout.

i By default, there is a limit of 20 gadgets per dashboard page. If you wish to raise this limit, edit the `jira-config.properties` file, set `jira.dashboard.max.gadgets` to your preferred value and then restart JIRA.

See also

- [Using dashboard gadgets](#)
- [Adding a gadget to the directory](#)
- [Subscribing to another application's gadgets](#)

Using dashboard gadgets

On this page:

- [About gadgets](#)
- [Pre-installed gadgets](#)
- [Extension gadgets](#)
- [Creating new gadgets](#)

About gadgets

Gadgets display summaries of JIRA project and issue data on the [dashboard](#). You can customize gadgets to display project and issue details relevant to particular users.

Adding Atlassian gadgets to external websites

You can also add Atlassian gadgets to compatible external websites, like iGoogle. For instructions on how to do this, refer to [Adding an Atlassian Gadget to iGoogle and Other Web Sites](#).

Pre-installed gadgets

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

Gadget	Description
Activity Stream Gadget	Displays a summary of your recent activity.
Administration Gadget	Displays checklist of common administration tasks and links to administrative functions and documentation.
Assigned To Me Gadget	Displays all open issues in all projects assigned to the user who views the dashboard.
Average Age Gadget	Displays a bar chart of the average number of days that issues have been unresolved.
Bamboo Charts Gadget *	Displays charts and plan statistics from a Bamboo server.
Bamboo Plan Summary Chart Gadget *	Displays a graphical summary of a build plan.
Bamboo Plans Gadget *	Displays a list of all plans on a Bamboo server, and each plan's current status.

Bugzilla ID Search Gadget	Allows the user to search all JIRA issues for references to Bugzilla IDs.
Calendar Gadget *	Shows issues and versions in a calendar format based on their due date. Calendars can be based on an issue filter or on a project.
Clover Coverage Gadget *	Displays the Clover coverage of plans from a particular Bamboo server.
Created vs Resolved Gadget	Displays a difference chart of the issues created vs resolved over a given period.
Crucible Charts Gadget *	Displays various charts showing statistical summaries of code reviews.
Favorite Filters Gadget	<p>Displays a list of the favorite filters for the user viewing the dashboard.</p> <div style="border: 1px solid red; padding: 5px; margin-top: 10px;"> <p>If your JIRA instance has many users and favorite filters, this gadget might cause JIRA to be slower than usual.</p> </div>
Filter Results Gadget	Displays the results of an issue filter.
FishEye Charts Gadget *	Displays two charts showing showing statistics about a sourcecode repository.
FishEye Recent Changesets Gadget *	Displays a number of recent changesets from a FishEye repository.
In Progress Gadget	Displays all issues that are in progress and assigned to the user viewing the dashboard.
Introduction Gadget	Displays a configurable introduction message on the dashboard.
Issue Statistics Gadget	Displays the collection of issues returned from a filter, broken down by a field.
Pie Chart Gadget	Displays issues from a project or issue filter, grouped by a statistic type, in pie-chart format. Issues can be grouped by any statistic type (e.g. Status, Priority, Assignee, etc).
Projects Gadget	Display information and filters related to a project(s).
Quick Links Gadget	Displays useful links to issues associated with the current user.
Recently Created Issues Gadget	Displays a bar chart of the rate at which issues are created, as well as how many of those issues are resolved.
Resolution Time Gadget	Displays a bar chart of the average resolution time (in days) of resolved issues.
Road Map Gadget	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.

Text Gadget *	Displays configurable HTML text on the dashboard.
Time Since Issues Gadget	Displays a bar chart of the number of issues that something has happened to within a given time period. The 'something has happened' is based on a date field that you choose, such as 'Created', 'Updated', 'Due', 'Resolved' or a custom field.
Two Dimensional Filter Statistics Gadget	Displays tabular data based on a filter.
Voted Gadget	Shows issues for which you have voted.
Watched Gadget	Shows issues you are watching.

i For more ideas, see the [big list of all Atlassian gadgets](#).

*This gadget is only available if you have installed/configured the relevant plugin.

Extension gadgets

Other gadgets are available as plugins on the [Atlassian Marketplace](#). To use these plugins, install and enable them.

Creating new gadgets

You can create new gadgets by writing an XML descriptor file, packaged as an [Atlassian plugin](#). See [Developing Gadgets](#) for more information.

Related topics

[The big list of Atlassian gadgets](#)

Adding a gadget to the directory

The JIRA gadget directory displays all the gadgets that are available for JIRA users to add to their dashboard.

You need to have administrator privileges to add a gadget to the directory. If you have permission to add gadgets to and remove gadgets from the directory itself, you will see the '**Add Gadget to Directory**' and '**Remove**' buttons on the 'Add Gadget' screen, as shown below.

On this page:

- [Adding a Gadget that is Not a Plugin](#)
- [Adding a Gadget that must be Installed as a Plugin](#)

Security implications

Add only gadgets from sources that you trust. Gadgets can allow unwanted or malicious code onto your web page and into your application. A gadget specification is just a URL. The functionality it provides can change at any time.

There are two types of gadgets: those that must be installed as plugins, and those that can be added as simple gadget URLs.

Adding a Gadget that is Not a Plugin

If the gadget is hosted on another server and can be added to the directory as a simple URL, then you can simply add it via your dashboard's 'Add Gadget' option.

To add a gadget to your directory,

1. First you need to find the URL for the gadget's XML specification file. Gadget authors and publishers make their gadget URLs available in different ways. Below are the instructions for an Atlassian gadget and a Google gadget.

- Follow the steps below if you need to find the URL for a gadget that is published by an Atlassian application, such as JIRA or Confluence: A gadget's URL points to the gadget's XML specification file. Gadget URLs are shown on the '**Gadget Directory**' screen that is displayed when you click '**Add Gadget**'. In general, a gadget's URL looks something like this:

```
http://example.com/my-gadget-location/my-gadget.xml
```

If the gadget is supplied by a plugin, the URL will have this format:

```
http://my-app.my-server.com:port/rest/gadgets/1.0/g/my-plugin.  
key:my-gadget/my-path/my-gadget.xml
```

For example:

```
http://mycompany.com/jira/rest/gadgets/1.0/g/com.atlassian.str  
eams.streams-jira-plugin:activitystream-gadget/gadgets/activit  
ystream-gadget.xml
```

To find a gadget's URL in JIRA:

- Go to your dashboard by clicking the '**Dashboards**' link at the top left of the screen.
- Click '**Add Gadget**' to see the list of gadgets in the directory.
- Find the gadget you want, using one or more of the following tools:
 - Use the scroll bar on the right to move up and down the list of gadgets.
 - Select a category in the left-hand panel to display only gadgets in that category.
 - Start typing a key word for your gadget in the '**Search**' textbox. The list of gadgets will change as you type, showing only gadgets that match your search term.
- Right-click the '**Gadget URL**' link for that gadget and copy the gadget's URL into your clipboard.

To find a gadget's URL in Confluence:

- Open the '**Browse**' menu and click '**Confluence Gadgets**' to see the list of available Confluence gadgets.
 - Find the gadget you want.
 - Right-click the '**Gadget URL**' link for that gadget and copy the gadget's URL into your clipboard.
- Follow the steps below if you need to find the URL for a Google gadget:
 - a. Go to the [Google gadget directory](#). (You can also get there by clicking 'Add Stuff' from your iGoogle home page.)
 - b. Search for the gadget you want.
 - c. Click the link on the gadget to open its home page.
 - d. Find the '**View source**' link near the bottom right of the page. Right-click the link and copy its location to your clipboard. This is the gadget's URL.
2. Now you can add the gadget to your directory. Go to the dashboard by clicking the '**Dashboard**' link or the '**Home**' link at the top left of the screen.
 3. The dashboard will appear. Click '**Add Gadget**'.
 4. The '**Add Gadget**' screen appears, showing the list of gadgets in your directory. Click '**Add Gadget to Directory**'.
-  You will only see this button if you have administrator permissions for your dashboard.
5. The '**Add Gadget to Directory**' screen appears. Type or paste the gadget URL into the text box.
 6. Click '**Add Gadget**'.
 7. The gadget appears in your gadget directory. (It will be highlighted for a short time, so that you can see it easily.)

Adding a Gadget that must be Installed as a Plugin

If the gadget must be installed as a plugin, you cannot add it via the gadget directory user interface. Instead, you will need to follow the instructions for adding a plugin, as described in [Managing add-ons](#). Once you have installed your plugin, the gadget will automatically appear in the directory.

Related topics

[The big list of Atlassian gadgets](#)

Subscribing to another application's gadgets

Security Implications

Add only gadgets from sources that you trust. Gadgets can allow unwanted or malicious code onto your web page and into your application. A gadget specification is just a URL. The functionality it provides can change at any time.

If you have administrator privileges, you can configure your application to subscribe to gadgets from other Atlassian applications. This feature allows administrators to make all the gadgets from one application available in another application, without having to enable each gadget individually via the gadget URL.

To make use of this feature, you will need two or more applications that support the feature.

The gadgets included are those provided by the other application or via plugins installed into that application. They do *not* include external gadgets that the other application has added to its directory.

To subscribe to gadgets from another application,

1. Go to the dashboard by clicking the **'Dashboard'** link or the **'Home'** link at the top left of the screen.
2. The dashboard appears. Click **'Add Gadget'**.
3. The **'Add Gadget'** screen appears, showing the list of gadgets in your directory. See the [gadget directory screenshot](#) below. Click **'Gadget Subscriptions'**.
 You will only see this button if you have administrator permissions for your dashboard, and if your application supports gadget subscriptions.
4. The **'Gadget Subscriptions'** screen appears, showing the applications to which your application already subscribes. Click **'Add Subscription'**.
5. The **'Add Subscription'** screen appears. See the [screenshot](#) below. Enter the base URL of the application you want to subscribe to. For example, `http://example.com/jira` or `http://example.com/confluence`.
6. Click **'Finished'** to add the subscription.

[Screenshot: Gadget directory with 'Gadget Subscriptions' button](#)

Gadget Directory

Search

All (67)

Bamboo (8)

Charts (16)

Clover (2)

JIRA (53)

Other (4)

Wallboard (23)



Activity Stream

By Atlassian

Lists recent activity in a single project, or in all projects.

<https://sgriffin.jira-dev.com/rest/gadgets/1.0/g/com.atlassian.streams.streams-jira-...>



Assigned to Me

By Atlassian

Displays all unresolved issues assigned to me

<https://sgriffin.jira-dev.com/rest/gadgets/1.0/g/com.atlassian.jira.gadgets:assigned-...>



Average Age Chart

By Atlassian

Displays the average number of days issues have been unresolved.

<https://sgriffin.jira-dev.com/rest/gadgets/1.0/g/com.atlassian.jira.gadgets:average-...>



Average Number of Times in Status

By Atlassian

Screenshot: Adding a gadget subscription

Add Subscription

Type or paste the base URL of the Atlassian application to subscribe to:

Please note that subscribing to this URL will add it to your instance's whitelist. For more information on how to edit this whitelist please see the [documentation](#).

When you subscribe to gadgets from another application, people will be able to use all those gadgets on their dashboards or pages in your application. The gadgets are those provided by the other application or via plugins installed into that application. They do not include external gadgets that the other application has added to its directory.

Only subscribe to applications that you trust! Gadgets can allow unwanted or malicious code onto your web page.

You can subscribe to any Atlassian application that publishes its gadgets for other applications to find.

An application's base URL looks something like this: <http://example.com/jira>

Related topics

[The big list of Atlassian gadgets](#)

Choosing a default language

Overview

Most user-visible pages in JIRA are now internationalized. When JIRA is first installed, you can select from the default languages:

- Chinese,
- Czech,
- Danish,
- Dutch,
- English (UK or US),
- Estonian,
- Finnish,
- French,
- German,
- Hungarian,
- Icelandic,
- Italian,
- Japanese,
- Korean (South Korean),
- Norwegian,
- Polish,
- Portuguese (Brazilian),
- Romanian,
- Russian,
- Slovak,
- Spanish (Spain), and
- Swedish.

! **Note:** For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Changing the default language

1. Choose



> **System.**

2. Select **General Configuration** to open the Administration page.
3. Click the '**Edit Settings**' button, then select the appropriate language in the drop-down box next to 'Default language'.

Any additional languages you have installed will appear in the list. See [Translating JIRA](#).

Per-user language selection

Individual users can manage their user profile, which will override the default language (see above).

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.

Known issues

- When you create a project using a system language other than English, JIRA will create duplicates of default issue types, statuses, resolutions, and priorities. These duplicates won't be translated into

On this page:

- [Overview](#)
- [Changing the default language](#)
- [Per-user language selection](#)
- [Overriding the default translations of issue types, resolutions, statuses, and priorities](#)
- [Known issues](#)
- [Related topics](#)

other languages chosen by your users. To work around this, either create new projects with the language set to English (and then change back to your preferred language) or provide custom translations for these duplicates (see [Translating JIRA constants](#)).

Related topics

- [Translating JIRA](#)

Translating JIRA

We're creating JIRA in English, but we know it's being used by teams around the globe. Here you can learn more about the translations we provide, and find some ways to translate JIRA on your own.

What translations of JIRA are currently available?

Official languages

JIRA ships with a number of translations in the most commonly-requested languages, which are updated and corrected with each release. You can easily update these via the Universal Plugin Manager - see [Managing add-ons](#).

As a JIRA administrator, you can choose the default language from the list of installed languages (see [Choosing a default language](#)). If you're a JIRA user, you can also choose your preferred language in the user profile.

Custom languages

If your preferred language is not on the list of official languages, there are a few things you can do:

- Go to [Packages Atlassian](#), and check if your language is listed there. You can download and add it to your JIRA instance, or even translate some strings on your own. [This kb article](#) will help you achieve that.
- Use the [InProduct Translations](#) plugin to translate JIRA on the fly, in any way you want.

Making changes to translations

We're constantly working on improving and updating our translations. If you had a problem or just think something could be translated better, give us a shout by creating a suggestion on our public JIRA instance. We'll get our best people to review and correct it.

▼ [Tell me how to do it...](#)

To report a problem with translations in JIRA:

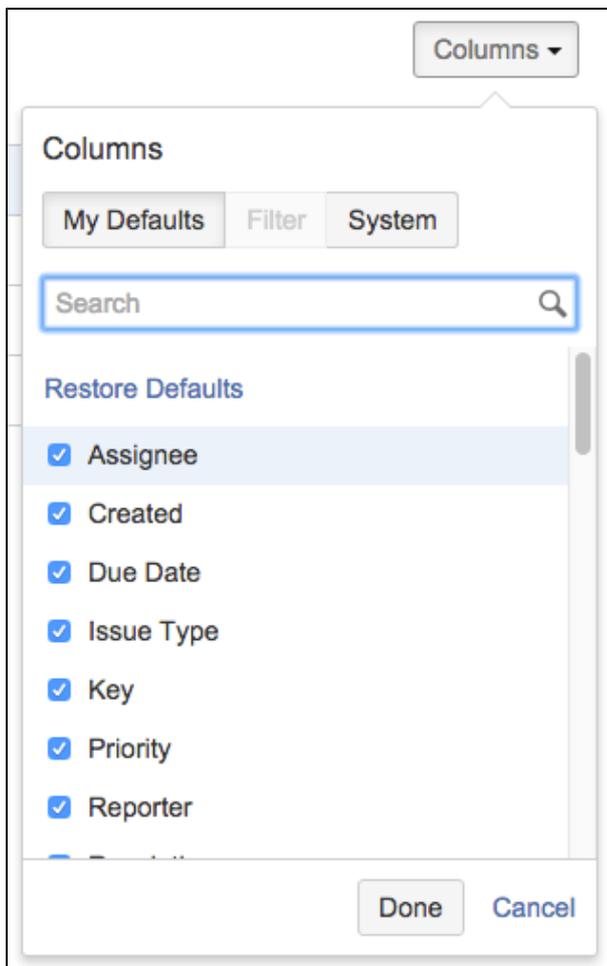
1. Go to [Atlassian Translations](#).
2. Click **Create** to create a suggestion. If possible, include the following details:
 - Product: JIRA (Server)
 - Language
 - Mistranslated text. You can provide your own translation here if you want.
 - Brief explanation
 - (Optional) Screenshot
 - (Optional) Version

What about translations of the documentation?

Currently, we only offer the Japanese translation of the JIRA documentation. See [JIRA Core](#), [JIRA Software](#), or [JIRA Service Desk](#).

Configuring the default issue navigator

JIRA applications let you change the columns of the table of search results for any search results displayed using the List view. Click **Columns** at top right of the issue table to open the column configuration dialog, shown below.



Column Configuration Dialog

This displays the list of the columns used in the current table of results. Choose the columns you want with checkboxes and click **Done** to finish. Notice that the Filter option is greyed out, this is because the the issue table results are not coming from a filter. See [Changing the column configuration for your own filters](#) for an example of using this dialog to set the displayed columns for your own filters.

Sorting and rearranging columns

- To sort issues, just click on a column header.
- To rearrange the column layout, press and hold the mouse button to enter "column drag mode."

My defaults, filter, and system

If the currently selected button is **My Defaults**, this indicates that the columns you are seeing are from your user account preferences. **Filter** is an available option whenever the issue search results come from a saved filter. If you are a JIRA Admin, you will also see the **System** tab, where you can change the columns for all users who have not set their own defaults.

JIRA administrators can configure the columns that appear in the Issue Navigator for all users that do not have personal column filters defined. When administrators are configuring default 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. JIRA administrators can also select which views are available in the JIRA system, as views are configurable via [plugins](#).

On this page:

- My defaults, filter, and system
- Changing the column configuration for your own filters
- Troubleshooting

Changing the column configuration for your own filters

If you are searching using a saved filter *and if the filter is owned by you*, use the **Filter** button to customize the columns displayed when users see results from that filter. When sharing a filter with other users, it's sometimes helpful to choose the relevant columns for those results. For example, if your filter searches for issues that are open bugs, you may decide to remove the columns for status and issue type for that filter since they will all be the same. Filters don't always have columns configured, but when they do, those columns will be shown unless the user chooses to use their defaults using the **My Defaults** button.

For any JIRA filters that you own, you can change the displayed columns as follows.

1. Click on the name of a JIRA filter you own.
2. Click the **Columns** button at top right of the currently displayed columns. This opens the column configuration dialog.
3. Select or deselect checked items in the list.
4. Click **Done** when you are finished.

Troubleshooting

If you cannot find a column, please make sure that you haven't run in to any of the following restrictions:

- You can only see columns for issue fields that have not been [hidden](#) and that you have [permissions](#) to see.
- It is possible to add any of the existing [custom fields](#) to the column list, as long as the fields are visible and you have the right permissions.
- Some project-specific custom fields, even if selected, do not appear in the Issue Navigator for all issues. Project-specific custom fields will be shown only if the filter has been restricted to that project only.
- Issue type custom fields aren't displayed by default in the Issue Navigator. However, if you include the issue type configured for the custom field in the query, you can select the custom field in the Column Configuration dialog to add the custom field to the Issue Navigator.

Creating links in the application navigator

You can add custom links in the application navigator, to make it easier for users to navigate to frequently used information.

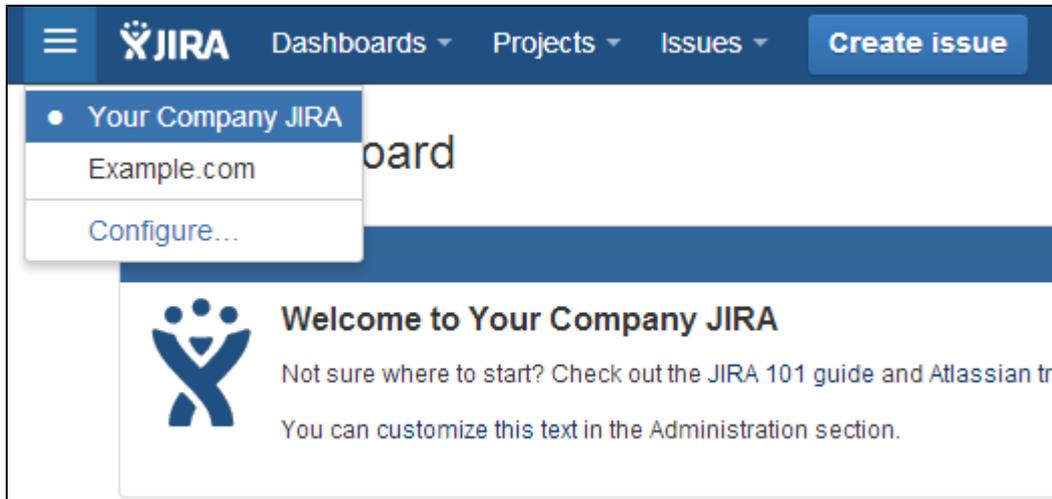
What is the application navigator?

The application navigator is the



control in the top left of the JIRA header that displays a menu of links to other applications. It is only displayed to users if there is more than one link. You can customize the links that appear in the application navigator, as well as making certain links only visible for specific users.

Screenshot: Application navigator



Adding links to the application navigator

If applications are linked to your JIRA instance via [application links](#), those applications will automatically appear in the application navigator. If you don't have any applications linked, the application navigator icon (



) will appear only for administrators. After links have been set up, the application navigator icon will automatically be visible to all users.

1. Choose



> **Applications.**

2. Select **Application Navigator.**
3. Create links by entering a name and the URL on the page.

After you've created a link, it will appear in the application navigator for all your applications after a few minutes (up to 10). Or, if you want links to appear immediately, you can navigate to the application navigator administration page in each application and refresh the page.

If you want to make a link appear in the application navigator for only specific users, use the **Groups** box to specify which groups can see the link. To hide the link from all users, select the **Hide** checkbox (for example, if you want to temporarily hide the link without deleting it entirely).

i When you make a link visible for a specific group, the link visibility is only set up in the application where you are configuring the link. For example, if you change the visibility in the JIRA administration screen and you also want it to be visible to the same users in Confluence, you must make the same changes in the Confluence administration settings.

To modify links that were created and are managed in other applications (for example, in a different JIRA application), edit the link in that application. You cannot delete links to linked applications, you must delete the application link instead.

Configuring the user default settings

Administrators can change the default user settings which are applied to user accounts on creation. These settings can be changed by the user on an individual basis through their profile.

Note: An administrator can force the user to use a specific Email format by clicking the **Apply** link. The user will then be unable to edit this setting.

Changing the user default settings

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Select



> **System > Default user preferences** to open the User Default Settings page.

3. Click the **Edit default values** button. The User Default Settings window displays.
4. Make the changes you wish to apply. A summary of the available changes is listed below.

Setting	Option
Email format	Outgoing email notifications from JIRA can be sent as HTML or text format.
Issues per page	This will set the number of issues displayed on each Issue Navigator page. Enter a value between 1 and 1000.
Default access	Choose the default access setting for when you create new filters and dashboards, which can be either shared with all other users (Public) or restricted to your viewing only (Private).
Notify users of their own changes	Choose between making JIRA send you email notifications about issue updates made by either both you and other people (Notify me) or other people only (i.e. Do not notify me).
Autowatch own issues	Choose between allowing JIRA to automatically make you a watcher of any issues that you create or comment on.

5. Click the **Update** button. Your changes have been applied.

Note: The first time you access the User Default Settings window, the Email format is set to text. This will be applied if you click **Update**. Ensure you have selected the correct Email format you wish to apply.

User management

You can use JIRA to manage its own users, or you can connect JIRA to an external user management system. You can also use JIRA as a user management system for other Atlassian products, so that your users have the same login details for all their Atlassian products.

Managing users

This section covers all aspects of using JIRA for user management.

Learn everything from how to create and view a user, to deactivation and monitoring user activity.

Managing groups

Learn which groups exist by default when you install a JIRA application as well as how to create, edit, and delete groups, add users to groups,. This section also covers assigning permissions to JIRA functions.

Advanced user management

Learn about the advanced user management features in JIRA, such as allowing other Atlassian products to connect to JIRA for user management, enabling public signup, and user management limitations and recommendations.

User directories

Learn more about your JIRA user directory and how to connect to external directories for external user management.

Managing users

As a JIRA administrator, you can manage users directly in JIRA, or enable public signup so users can create their own accounts. You can refer to these pages for information on managing users across multiple projects and applications.

For all of the following procedures, you must be logged in as a user with the JIRA Administrators or JIRA System Administrators global permission.

Documentation	What you can do
Create, edit, or remove a user	Learn about different ways to create users in JIRA, edit user information and properties, and deactivate or delete users who no longer need to use the system.
Assign users to groups, project roles, and applications	Give users access to different functions in JIRA like project roles and applications. Users are created without any access so this is a critical step to allow your team to get started.
Monitor a user's activity	Keep an eye on user activity to keep your system running well. This information can help Administrators analyze JIRA performance and also know which users have been inactive for a while.
Prevent automatic login	Administrators can control which user login information is stored or turn off this feature completely. Read more to learn about the details and benefits.
Manage password policy	Make sure your JIRA system is secure by implementing a password policy and CAPTCHA.

Create, edit, or remove a user

Users in JIRA applications can be managed manually or via External User Management. This page helps you manage these users manually, and references external user management systems where required. In order for a user to log in and access a JIRA application, they must have application access. Application access is obtained by being a member of a group assigned to an application. Membership to these groups can be changed at any time on a per user basis.

Before you begin

You must have the JIRA Administrator or JIRA System Administrator global permission to be able to manage users in JIRA applications.

Create users

There are several ways to create a user in JIRA. Read on to learn which method is right for your team.

If you're adding users to a JIRA Service Desk project, check out [Setting up service desk users](#).

Create a user in JIRA

Create a user directly in JIRA if you have a small team. Consider external user management (LDAP or Active Directory) if you have a lot of hands on deck. Maintaining permissions for individual user ID's can be messy if you have too many users, so there are other options for your large staff.

▼ To create a user:

1. Select



> **User Management.**

2. In the User browser, click **Create User**.
3. Enter the **Username**, **Password**, **Full Name** and **Email** address.
4. Optionally, select the **Send Notification Email** checkbox to send the user an email containing:
 - their login name.
 - a link from which to set their password (this link is valid for 24 hours).
5. If you have more than one JIRA application installed, you will need to select the JIRA application you would like to give the user access to.
6. Click the **Create** button.

After the user is created, you'll be brought to a screen where you can view the user information and perform

additional functions such as edit details, edit groups or properties, and delete the user.

Invite users to JIRA

You can invite users to JIRA through email. When the users accept the invite and they are created in JIRA, they will be given access to the [applications set as default](#).

Note, JIRA's SMTP mail server must be configured to send notifications before you can invite users via email.

▼ [To invite a user](#)

1. Open the **User** browser and click the **Invite Users**.
2. Enter the email addresses of the users that you want to invite. Add multiple users by separating the email addresses with a comma.

Note: You cannot invite users by sending an invitation to a distribution list.
3. Click the **Send** button to send the invitations.
 - Each invitation can only be used to create a user under the email address that it was sent to, and can only be used once.
 - Each invitation will expire seven days after the day it was sent.
 - Your user license count will not be affected until users accept the invitation and the users are created.

Use a mail handler, connect to an internal directory, or enable public signup

There are a few other ways to create a user in JIRA. These methods are more specialized and can fill a specific need of your team.

▼ [Others](#)

Automatically create a user

You can use a mail handler to allow JIRA applications to create issues or comments via emails received. The handler can also be configured to create new users based on the sender's email address. This method can be used from time to time if you want JIRA to create new user accounts from any received email messages whose **From:** field contains an address that does not match one associated with an existing JIRA user account. This allows the creator of the email message to be notified of subsequent updates to the issue. See '[Creating issues and comments from email](#)' for full documentation.

Connect to an internal directory with LDAP authentication

You can connect your JIRA application to an LDAP directory for delegated authentication. This means that JIRA will have an internal directory that uses LDAP for authentication only. Choose this option if you want to set up a user and group configuration within your application that suits your needs, while checking your users' passwords against the corporate LDAP directory. This option also helps to avoid the performance issues that may result from downloading large numbers of groups from LDAP. See '[Connecting to an internal directory with LDAP authentication](#)' for more information on configuration.

Allow users to sign up publicly

For some organizations using JIRA Service Desk, it's appropriate to allow users to create their own accounts without needing an Administrator. This is a good way to empower users without using up all of your JIRA Service Desk licenses, but can raise some security concerns. See '[Enabling public signup and CAPTCHA](#)' for more information about enabling public sign up and CAPTCHA.

Select default applications for new users

If you have more than one JIRA application, it's possible to select which applications new users will automatically be assigned to. If you manually create a user, the applications you select as defaults will be preselected. However it's possible to change this while creating the user. If you allow users to sign up via email, via public signup, or through a mail handler, they will be given access to the applications you select:

▼ [To set default applications](#)

1. **Choose**


> Applications.
2. Select **Set defaults for new users**.

3. Select the application/s to set as default and click the **Set defaults** button.

You've now set the default applications to be used for new user creation. These users will be assigned to the default groups of the applications you have selected.

Edit a user

Modifying user information, such as name, email, address, and password, is easy with the JIRA internal directory. If you are using an external authentication method such as LDAP or Active Directory, you'll have to make changes in that system rather than in JIRA.

Edit a username, full name, or email address

These three attributes can be modified together, in a few simple clicks, if you're using the JIRA internal directory to manage users. When updating a username, it's important to note that:

- JIRA cannot update external usernames - for example, users that are coming from an LDAP server or Crowd instance. However, JIRA can update JIRA users stored in an "Internal Directory with LDAP Authentication."
- If you are using your JIRA instance as a JIRA User Server for other applications, (e.g., Confluence), you will not be able to use this feature. If you aren't sure about this, check under **User Management > JIRA User Server** to confirm that no external applications have been configured to use JIRA as a Crowd Server.

▼ To update user information:

1. Select  **> User Management.**
2. Find the user in the user list using the filter form at the top of the page.
3. Click **Edit** in the Operations column.
4. Make the changes to username, full name, or email address and click **Update** to finish.

Change a password

Administrators can change user passwords directly in JIRA when using the internal directory. A password cannot be changed if users are managed from an LDAP server or Crowd instance.

▼ To update a password:

1. Select  **> User Management.**
2. Find the user in the user list using the filter form at the top of the page.
3. Click on the **username**.
4. Choose **Actions > Set Password**.
5. Enter and confirm the new password, and click the Update button to finish.

Add a property to a user

A Property is an extra piece of information about a user that you can store in JIRA. A Property consists of a Key of your choice, like 'Phone number' or 'Location', plus a corresponding Value (eg. '987 654 3210', 'Level Three'). User Properties do not have an effect in the project apart from adding additional information about the user. Plugins, however, can frequently use this data.

▼ To add a property:

1. Select  **> User Management.**
2. Find the user in the user list using the filter form at the top of the page.
3. Click on the **username**.
4. Choose **Actions > Edit Properties**. The **Edit User Properties** screen will be displayed.
5. Enter the new **Key** and its **Value**, then click the **Add** button to finish.

Remove a user

Have a user that no longer needs access to JIRA? Read about the different ways to remove access. Rather than deleting a user, we recommend that you deactivate their account. Deactivating a user's account will prevent the account from being used, but it will preserve that user's history of activity.

Deactivate a user

JIRA administrators can deactivate a JIRA user, which disables that user's access to JIRA. This avoids the need for a JIRA administrator to delete the user's account from the system.

This feature is useful when a JIRA user leaves the organization or changes departments because their history of JIRA activity is preserved in the system. If a user with a deactivated JIRA account needs access again at some point in the future, their JIRA user account can be easily reactivated.

▼ To deactivate a user:

1. Select  > **User Management** and find the user in the user list.
2. Click **Edit** in the Operations column.
3. Clear the **Active** checkbox.
4. Select **Update** to confirm the change.
5. The user will now appear in the user list with a strikethrough their username and full name, and the text '(inactive)'.

Note

- To deactivate a [project](#) or [component lead](#), assign other users as the relevant project or component leads first. These users cannot be deleted without first replacing their roles. An error message will appear asking you to assign another user first.
- If your JIRA instance is configured to use an external Atlassian Crowd user directory, the user will be deactivated in JIRA if they are deactivated in Crowd.
- JIRA does not deactivate users who are configured and deactivated/disabled in an external Microsoft Active Directory or LDAP-based user directory, with the exception of JIRA users configured with 'delegated LDAP authentication'.

When you deactivate a user, that user:

- Will no longer be able to log in to JIRA.
- Will not count towards your JIRA user license limit.
- Can't be assigned issues or added as a watcher to issues whenever issues are created or edited.

However:

- A user who was assigned, was watching, or had reported any issues in JIRA before their account is deactivated, will still appear as the respective assignee, watcher, or reporter of those issues. This situation remains until another user is specified as the assignee or reporter, the deactivated user is removed as a watcher from them, or the account is reactivated.
- A user who voted on any issues in JIRA before their account is deactivated will continue to appear as a voter on these issues.
- Will continue to appear on the JIRA user interface with '(Inactive)' displayed after their name.
- Can still be used to filter issues in a JIRA search query.
- Will not receive any [email notifications](#) from JIRA, even if they continue to remain the assignee, reporter, or watcher of issues.

Delete a user

We recommend you think carefully before deleting a JIRA user. Consider deactivating instead and see the section above for more information.

Before you delete, note that:

- You cannot delete a user from within JIRA if you are using External User Management (However, you can deactivate the user. See instructions above).
- You cannot delete a user from JIRA if they have:
 - reported or been assigned to any issues.
 - commented on any issues.
- The filters and dashboards of a user will be deleted when the user is deleted, even if the filters or

dashboards are shared with other users.

- All issues that have been reported by or assigned to the user you are attempting to delete, are respectively hyperlinked to a list of the individual issues in the Issue Navigator.

▼ To delete a user:

1. Select



> **User Management** and find the user in the user list.

2. Click the **Delete** link in the **Operations** column.

The confirmation screen that follows will summarize 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.

3. Take any actions required to disassociate the user with JIRA. The error message will give you exact instructions but these may include:
 - Reassigning any issues currently assigned to the user.
 - Bulk-editing the issues created by the user and changing the 'Reporter' to someone else. You will also need to allow editing of closed issues if some of the issues the user created are closed and you do not wish to reopen them.
 - Changing the owner of shared dashboards owned by the user. See [Managing shared dashboards](#).
 - Changing the project lead for any projects where the user is a lead.
4. If there are no issues assigned to, or reported by the user, and the user has not commented on any issues, the confirmation screen will display a **Delete** button. Click to delete the user.

Assign users to groups, project roles, and applications

When a user is created in JIRA, they'll be automatically added to the default user group for your installation (jira-users, jira-software-users, jira-servicedesk-agents). As an administrator, you can choose to create a more granular security model by creating multiple user groups that grant different levels of access within the JIRA instance (see more about user groups in the [managing groups](#) section).

This section will give you instructions on how to assign permissions by adding users to groups and assigning a user to a project role.

Before you begin

You must have the JIRA Administrator or JIRA System Administrator global permission to be able to manage users in JIRA applications.

Add a user to a group

The best way to give a user access to specific JIRA functions is to add a user to a predefined user group.

▼ To add a user to a group:

1. Select



> **User Management** to view the user list.

2. Find the user in the user list using the filter form at the top of the page.
3. Click **Groups** in the Operations column.
4. Use the search box to find the group that you want to add the user to. You can add more than one group at a time in that search field if you need to add the user to multiple groups.
5. Click **Join selected groups** and the user will be added.

Assign a project role

One way to give users access to a project role is to grant access at the user level. If you have fewer than 50 JIRA users, you can manage user permissions by manually assigning users to a project role. If you have more than 50 users, we recommend adding users to a group that can then be assigned to a project role, as explained above.

▼ To assign access to project role on the user level:

1. Select



> **User Management** to view the user list.

2. Find the user in the user list using the filter form at the top of the page.
3. Click **Project Roles** in the Operations column.
4. Select **Edit project roles** to add / remove a user from a project role. On this screen, you can also see the Groups that provide access to each project role.
5. Check the box for the project role that you want to give access to (**Administrators, Developers, or Users**) and click **Save** to finish.

Assign a user to an application

You may need to give an existing user access to an application, or remove access to an application. Application access can be assigned and removed in the User browser. When you assign a user to an application in this manner, they will be added to that applications [default group](#). When you remove application access, the user is removed from all the groups that could grant them access to that application.

▼ To assign access to an application:

1. Open the **User** browser.
2. Select the user you wish to assign or remove access from. The user is displayed and the applications they are assigned to are display as checked under **Applications and groups**.
3. Check the box for the application/s you want to assign to a user. Uncheck the box to remove access. Note that the changes are made in real time, as you add or remove access, the group memberships change.

Monitor a user's activity

As a JIRA administrator, you can view individual user activity or user sessions on a global level.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

View individual user login activity

Administrators may want to check individual user login activity to:

- See if certain users are still active in JIRA.
- View the age of a disabled user to clean up any ID's that have passed a certain timeframe.

▼ To see login activity:

1. Choose



> **User Management**.

2. Click on the **username** in the list.
3. You will be presented with a window including;
 - Login count
 - Last login
 - Previous login
 - Last failed login
 - Current failed login count
 - Total failed login count
4. Use this information wisely.

View global user sessions

JIRA provides a list of users who are currently accessing JIRA. Use this information to:

- Know who to contact before planned downtime.
- Regularly monitor the average number of active users to evaluating your licensing quota.
- View user count if the system is under duress.

- And more!
- ▼ To view current user sessions for the JIRA instance:
 1. Choose  >**System**.
 2. In the left panel, select **Security > User Sessions** to open the Current User Sessions in JIRA page.

Note

It's possible to have session ID's for computers that are not logged in. For example, when someone accesses JIRA without logging in, a unique session is created without a username. This is shown as "Not Available" in the User column.

Manage password security

Create a more secure JIRA environment by enabling a password policy, setting custom password settings, or enabling password similarity checks.

Enabling a password policy

The JIRA password policy is disabled by default. This policy is only useful when JIRA users are able to change their own passwords. If JIRA is connected to an external user management system (LDAP, Active Directory, Crowd), this policy should not be used since passwords are maintained externally from JIRA.

- ▼ To enable a password policy:

1. **Select**  > **Security**.
2. Click **Password Policy** in the left panel. Select one of the following options:
 - a. **Disabled** – The equivalent of having no password policy (this is the default).
 - b. **Basic** – Requires passwords to be at least 8 characters long and use at least 2 character types. Rejects passwords that are very similar to the previous password or the user's public information.
 - c. **Secure** – Requires passwords to be at least 10 characters long and use at least 3 character types including at least 1 special character. Rejects passwords that are even slightly similar to the previous password or the user's public information.
 - d. **Custom** – Lets you use your own settings (see below for more information).
3. Click the **Update** button to finish.

Setting custom password policies

There are many optional fields that can be set when you choose a custom password policy.

- ▼ [Custom settings](#)

Set 'Custom' password settings

Update the necessary fields to meet your company's password standards:

1. **Password Length** – Set a minimum and maximum length for your passwords. The defaults are 8 and 255.
2. **Character Variety** – Use these fields to set requirements around types of characters – uppercase letters, lowercase letters, special characters, and so on.
3. **Similarity Checks** – See the section below for details on this feature.

Similarity checks for 'Custom' password settings

This is a system check to make sure that your users aren't creating a new password that is too similar to the current password, the user's name, or email address. It can be set to **Ignored**, **Lenient**, or **Strict**.

What's the difference between Lenient and Strict?

- **Lenient** checks for obvious similarities, like reversing the `username` or moving the front letter to the end.
- **Strict** checks for more subtle variations, like mixing up the letters or adding just one new character. It also performs a character frequency analysis.

Enabling CAPTCHA

If your JIRA application server is accessible from outside your organization'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 recognize 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. See '[Enabling public signup and CAPTCHA](#)' for more information about enabling this option.

Password FAQ

~ FAQ

Question: What is Character Variety and why should I use it?

Answer: Character variety refers to the different types of characters you can create on a keyboard: lowercase letters, uppercase letters, numbers, and special characters. Requiring different character types makes passwords harder to guess, but it might also make them harder to remember. Use your best judgment when setting these fields, keeping in mind your company's requirements as well as your user base.

Question: Does this policy affect existing passwords?

Answer: The policy is only enforced as passwords are changed; there is no way to detect whether or not existing passwords satisfy the policy or to force the users to update their passwords if the policy has been changed. As a workaround, you can use [this Crowd REST resource](#) to forcibly change the users' passwords to something they won't know, thereby requiring them to reset it to get back in, and the password reset enforces the policy rules.

Prevent automatic login

Overview

When a user logs in to JIRA, they have the option of making JIRA remember their login information by selecting the 'Remember my login' checkbox before they click the 'Log In' button. When they do that, a 'Remember my login' token is stored by the JIRA server and a cookie containing this token is set in the user's browser.

A user who revisits JIRA from the same computer and browser, will automatically be logged in if JIRA detects that one of the user's 'Remember my login' tokens has a matching token contained in one of the browser's cookies. If the user logs out of JIRA, the 'Remember my login' token is cleared from the JIRA server.

To maximize and maintain the security of your JIRA instance, JIRA provides features for:

- Disabling 'remember my login' functionality for the JIRA instance.
- Clearing 'Remember my login' tokens for individual user accounts.
- Clearing all 'Remember my login' tokens stored by your JIRA instance.

Manage automatic logins:

- To maximize security by requiring a user to enter all of their credentials each login.
- If users have been accessing your JIRA application in a public environment.
- If users aren't in the habit of formally logging out of JIRA.

Before you begin

You must be logged in as a user with the **JIRA Administrators** [global permission](#) to complete any of the following procedures.

Clear a 'remember my login' token for a specific user

JIRA administrators can clear all 'Remember my login' tokens associated with a user's account through the JIRA administration console.

▼ To clear a login token for a user:

1. Choose



> **User Management.**

2. Find the user in the list and click the **Username** or **Email Address** of the user whose 'Remember my login' tokens you wish to remove. Details about that user and their login information is displayed.
3. Click the '**Remember My Login**' link to display that user's **Remember My Login** page.
4. Click the '**Clear All**' button to remove all 'Remember my login' tokens associated with this user account from the JIRA server.

Clear all 'remember my login' tokens for the entire JIRA instance

JIRA administrators can also clear all 'Remember my login' tokens from their JIRA instance with a few simple clicks.

▼ To clear a login token for the instance:

1. Choose



> **System.**

2. In the left panel, select **Security > Remember My Login** to open the Remember My Login for All Users page.
3. Click the '**Clear All**' button to remove all 'Remember my login' tokens from the JIRA server.

Disable 'remember my login on this computer' option for your JIRA instance

If you never want JIRA to remember login tokens, you can choose to disable 'remember my login' tokens for the entire JIRA instance.

▼ To disable this feature:

Option 1 (recommended)

The checkbox for this option can be disabled by setting the `jira.option.allowcookies` property to `false` in your `jira-config.properties` file. You will need to restart JIRA in order for this change to take effect.

Option 2

Edit the `./atlassian-jira/includes/loginform.jsp` file.

SAML SSO for JIRA Data Center applications

Security Assertion Markup Language (SAML) is an XML-based data format that allows a service to exchange authorization data with an identity provider (IdP). The most common use case is allowing a user to sign in to multiple software applications using the same authentication details, usually a username and password. This is referred to as single sign-on (SSO).

We provide the functionality for JIRA Software Data Center and JIRA Service Desk Data Center applications to connect to your IdP so that you can provide your users with an SSO experience. This *only* handles authentication. Application access and any required authorizations, such as ensuring that users belong to the appropriate groups/roles and have the necessary permissions, should be configured in the user directory and/or the application itself.

Setting up single sign-on

You'll need to configure your application and your IdP to provide single sign-on for your users.

Supported Identity Providers

SAML single sign-on should work with any identity provider implementing the SAML 2.0 Web Browser SSO Profile, using the HTTP POST binding.

We currently perform tests with the following identity providers:

- [Microsoft Active Directory \(using ADFS 3.0\)](#)
- [Microsoft Azure Active Directory](#)
- [OneLogin](#)
- [Okta](#)
- [PingIdentity](#)

Setting up SSL/TLS

To make sure that SAML authentication is secure and private, you need to set up SSL/TLS in the application. You can find the relevant steps in our [Running JIRA applications over SSL or HTTPS](#) page. Once set up, you need to make sure that the application's [configured base URL](#) is using the HTTPS protocol.

Setting up SSL/TLS using a reverse proxy

If you want to use a reverse proxy, please refer to these product specific documents, describing the exact configuration steps:

- [Proxying Atlassian server applications with Apache HTTP Server \(mod_proxy_http\)](#)
- [Integrating JIRA with Apache using SSL](#)
- [Securing your Atlassian applications with Apache using SSL](#)

When using a reverse proxy that terminates SSL/TLS, you need to make sure that the request URL the application server sees matches the fully-qualified domain name for the reverse proxy. This is usually achieved by configuring the `<Connector>` directive with the appropriate `proxyName`, `proxyPort`, `secure` and `scheme` settings. Please check the documentation above for specific examples.

Setting up your Identity Provider

If you want your application to provide SSO, you'll need to add it to your IdP. The exact process varies depending on the IdP, but you'll usually need to:

- Define an 'application' in your IdP
- Provide some data about the application, including data you can access on your application's [Authentication screen](#)
- Make sure the NameID attribute of the users in your IdP is set to the username in your Atlassian application
- Give the appropriate users permission to use the application

At the end of the setup process your IdP will provide you with a set of data that you'll need to configure your Atlassian application.

Configuring SAML Authentication in your Atlassian application

1. Navigate to the SAML Authentication screen by selecting **> System > SAML Authentication**.
2. Select **SAML single sign-on**.

Configure the following settings:

Setting	Notes
Single sign-on issuer	<p>This value is provided by your IdP, as part of setting up SAML. It's sometimes also called 'Entity ID'</p> <p>The issuer is the IdP your application will be accepting authentication requests from</p>

Identity provider single sign-on URL	<p>This value is provided by your IdP, as part of setting up SAML.</p> <p>It defines the URL your users will be redirected when logging in.</p>
X.509 Certificate	<p>This value is provided by your IdP, as part of setting up SAML. This is sometimes referred to as a 'Signing certificate'. The key usually starts with '-----BEGIN CERTIFICATE-----'.</p> <p>This contains the public key we'll use to verify that all received SAML authentication requests have been issued by your IdP.</p>
Login mode	<p>This defines how your users can use single-sign on. The options are:</p> <ul style="list-style-type: none"> • Use SAML as secondary authentication – the default way to log in will be the standard application login form. You can log in using SAML if you go to your IdP and select your application, or by using the this URL to log in: <i>BASE-URL/plugins/servlet/external-login</i>. We recommended this method so you can test that everything is configured correctly, and that users can log in using SSO. • Use SAML as primary authentication – in this mode, all browser-based users will be redirected from the application's login screen to the IdP to log in. It's still possible to authenticate by: <ul style="list-style-type: none"> • Basic Auth • Form-based auth via dedicated REST endpoint • Existing Remember Me tokens <p>You should only enable this mode once you've verified that SAML authentication is working as expected.</p>
Remember user logins	<p>When checked, successful user logins will be remembered in the user's browser. When browsing to their application, users will be logged in automatically without having to authenticate again using SAML.</p>
Include customer logins (JIRA Service Desk only)	<p>When checked, all login requests from your Service Desk customers using the customer portal will be redirected to the configured IdP. If not selected, customers must log in through the customer portal.</p>

3. The following information is provided on the Authentication screen, and will be required to configure your IdP:

Setting name	Notes
Assertion Consumer Service URL	This is the URL the IdP will return SAML authentication requests to.
Audience URL (Entity ID)	This is the URL the IdP will prepare SAML authentication requests for.

4. Click **Save configuration**.

Once you've configured both your application and your IdP, you're ready to start using SSO.

Best practices

- SAML authentication requests are only valid for a limited time. You should make sure the clocks on the server running your application/s and the IdP are synchronised.
- If users and groups in your application are configured using [User Directories](#), you'll usually want to use the same LDAP directory to be the source of users for both your IdP and Atlassian application. Users need to exist in the user directory *before* they can log in using SSO.

Troubleshooting

- If you make a mistake configuring the SAML authentication, or are unable to log in using your IdP, you can restore login form authentication by using issuing a DELETE request (using a username and password for an administrator configured in your user directory):

```
curl -u admin_user:admin_password -X DELETE
http://base-url/product/rest/authconfig/1.0/saml
```

- If an authentication error occurs, the user will only see basic details about what went wrong. For security reasons, the details about the underlying problem are not shown. You'll need to check the application logs to see the cause of the problem.
- In some cases you might also experience errors shown by your IdP. For those you will need to use the support and tools provided by your IdP, rather than Atlassian support.
- When using **SAML as primary authentication** and you have CAPTCHA enabled in the application, users that use HTTP basic authentication (for example in REST resource calls) may get locked out if they enter an incorrect password too many times. In these cases, an administrator will need to reset the user's CAPTCHA in the user list screen.

Enabling public signup and CAPTCHA

About public signup and CAPTCHA

For some organizations it is appropriate to enable public signup, which allows users to create their own accounts. If these users create accounts that gives them access to JIRA Core or JIRA Software, these accounts will consume a license for these applications. If public signup is switched on for JIRA Service Desk, and customers create their own accounts, these accounts do not consume a license.

If public signup is not enabled, then only a JIRA administrator can [create new user accounts](#).

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.

On this page:

- [About public signup and CAPTCHA](#)
- [Enabling public signup for JIRA Core and JIRA Software](#)
- [Enabling public signup for JIRA Service Desk](#)
- [Enabling CAPTCHA for JIRA application login screens](#)

If your JIRA application server is accessible from outside your organization'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 recognize 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.

Enabling public signup for JIRA Core and JIRA Software

1. Log in as a user with the '**JIRA Administrators**' [global permission](#).
2. Choose 
 - > **System**. Select **General Configuration** to open the Administration page.
3. Click '**Edit Configuration**' at the end of the page.
4. In the 'Mode' drop-down, select '**Public**'.
5. Click the '**Update**' button at the bottom of the screen.
6. Log out of JIRA, then click the '**Log In**' link at the top right of the screen and verify that the '**Sign Up**' link is displayed at the bottom of the login screen.

Enabling public signup for JIRA Service Desk

With public signup enabled, agents can invite new customers to a service desk project, and new customers can create accounts on the Customer Portal and through email. Enabling public signup for your service desk project also enables a honeypot technique which helps prevent spambots from creating accounts through the customer portal.

You must first enable public signup at the system level:

1. Log in as a user with the '**JIRA Administrators**' [global permission](#).
2. Choose 
 - > **Applications**. Scroll down to the **JIRA Service Desk** section and choose **Configuration**.
3. In the **Public signup** section, enable the setting.

You or a service desk project administrator can then open a service desk at the project level:

1. Go to **Project administration** > **Customer permissions**.
2. Select **Anyone can sign up for a customer account on my Customer Portal**.

New customers will be added to the **Service Desk Customers** project role. Note that customer accounts created via public signup don't count towards a service desk license.

In situations where users are unable to change their passwords, check that a Delegated Authentication Directory is not the highest in the order of User Directories. As a workaround, you can change the order of User Directories, or alternatively use a connection to a LDAP directory instead.

Enabling CAPTCHA for JIRA application login screens

CAPTCHA can be enabled so that anyone attempting to sign up to your JIRA instance through the JIRA login screen will be presented with a random sequence of letters, that they must type to confirm they're a real person. This is to try prevent spamming, and malicious attacks.

1. Log in as a user with the '**JIRA Administrators**' [global permission](#).
2. Choose 
 - > **System**. Select **General Configuration** to open the Administration page.
3. Click '**Edit Configuration**' at the end of the page.
4. Locate 'CAPTCHA on signup' and select '**On**'.
5. Click the '**Update**' button at the bottom of the screen.
6. Log out of JIRA, click the '**Log In**' link at the top right of the screen, then click the '**Sign Up**' link and verify that a random sequence of letters is displayed at the bottom of the '**Sign Up**' screen — e.g.

"winzers" in the following screenshot:

Sign up

i To sign up for JIRA simply enter your details below.

Username *

Password *

Confirm Password *

Full Name *

Email *

Please enter the word as shown below



Managing groups

JIRA groups

A JIRA group is a convenient way to manage a collection of users. You can use groups throughout JIRA to:

- Allow application access.
- Grant global permissions or project specific access.
- Receive email notifications.
- Access issue filters and dashboards.
- Reference workflow conditions.
- Integrate with project roles.

JIRA default groups

Two groups are automatically created when you install JIRA for the first time: the jira-administrators group and one user group associated with the application.

Group	Application	Description	Use
jira-administrators	All	<p>Contains people who are JIRA system administrators. By default, this group:</p> <ul style="list-style-type: none"> • is a member of the Administrators project role. • has the JIRA Administrators and the JIRA System Administrators global permissions. 	<ul style="list-style-type: none"> • Membership should be limited to a few JIRA Administrators or super users. • Provides unlimited access. • Recommended to never delete or alter permissions for this group because it would limit accessibility to the JIRA instance.

jira-core-users	JIRA Core	By default, these groups have the Browse Users, Create Shared Filter, Bulk Change and Manage Group Filter Subscriptions global permissions.	<ul style="list-style-type: none"> • Optional user groups. • May be useful if your JIRA instance has very few users that require generic, standard access. • Can be deleted if your JIRA instance requires granular, specific access for individual groups of users.
jira-software-users	JIRA Software		
jira-servicedesk-users	JIRA Service Desk		

Note

If you're using External User Management, you won't be able to create, delete, or edit groups or group membership from within JIRA, and automatic group membership will not apply. However, you'll still be able to assign groups to [project roles](#).

View, create, or delete a group**Before you begin**

You must be logged in as a user with the **JIRA Administrators** or JIRA System Administrators global permission to perform the following procedures.

View the group browser

The Group Browser in JIRA allows you to view, create, and edit groups, while also allowing you to modify members, and view group permissions and settings.

▼ [To view the group browser:](#)

1. Choose



>**User Management**.

2. Select **Groups** to open the Group Browser.
3. Click the group name to see the permissions, email notification schemes, security levels, and saved filters.

Create a group

Create new groups in JIRA to customize security permissions based on roles. Users may be added to many groups depending on the level of access that they need to do their job.

▼ [To create a group:](#)

1. From the group browser, type the new group name in the **Add Group** form.
2. Click **Add Group** and you're done.

Note

New groups are created without access to JIRA functions so you'll have to assign permissions to the group before members can inherit functionality.

Delete a group**Before you delete**

- Check whether the group is being used by any [permission schemes](#), [email notification schemes](#), [issue security levels](#), or saved filters.

- Consider the impact this may have on users in that group. For example, if a user receives access to a specific feature only from this group, then the user will no longer have that permission and it may impact their work.

▼ To delete a group:

1. Choose



> **User Management.**

2. Select **Groups** in the left panel to open the Group Browser.
3. Click **Delete** in the Operations column .
 - a. You will be redirected to a confirmation screen that explains that users will be removed from the group through its deletion. The users themselves will not be deleted from JIRA during this operation.
 - b. Note that you cannot delete a group that is currently the only default group for an application. The **Delete** link will be greyed out.
4. Consider carefully and click **Delete** to finish (or **Cancel** if you've decided to reconsider).

Modify group membership

Editing group membership

To edit a group's membership, click the **Edit Members** link in the row for that group in the **Group Browser**. This takes you to a form that allows you to add or remove users from the group.

Note

- When a user is created and assigned to an application, they are automatically added to that application's Default group.
- 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 log in [permissions](#) (i.e.application access) without first reducing the number of users with log in [permissions](#).
- If the group has the 'JIRA System Administrators' [global permission](#), you cannot edit its membership unless you have the 'JIRA System Administrators' global permission.

Automatic group membership

To automatically add newly-created users to a particular group, you can assign the group as an [application's default group](#), and then assign the application as the [default application for user creation](#). Or specify the group name in the 'Default Group Memberships' option when Connecting to an LDAP directory. See [Adding users to groups automatically](#) for instructions.

Assign group access to a project role

You can grant access to project roles and applications through groups. Simply add a user to a group with predefined security settings to give them the access they need. Creating a clear security model, with specialized user groups, that grant specific access to applications is the easiest way for longterm admin support.

The best way to give users access to a project role is to grant access to a group. This way you can assign a group access and then simply add a user to the group and save yourself some time managing individual user permissions.

▼ To assign access to a project role on the group level:

1. Select



> **Projects.**

2. Click the title of the project that you want to assign permissions to.
3. Click **Users and Roles**.
4. Click the **Add users to a role** link.
5. Type the group (or user) names you want to add to a role.
6. Select the role from the drop-down.
7. Click **Add** to finish.

See '[Assign users to groups, project roles, and applications](#)' for more information.

Manage group access to applications

Users need to belong to a group to access JIRA applications. Administrators can assign access to groups through the Application access page.

Each application that's licensed in JIRA is listed on this page, and each application will display its total allowed users, and users remaining. Each application should have at least one group associated with it, and one default group should be selected. When a user is created in JIRA, and assigned to an application, they will be automatically added to the default group for that application. We strongly recommend you only have one default group assigned to each application. Use the default group to allow application access, and [create and manage other groups](#) to control [project specific permissions and access](#).

For example, when JIRA Software is installed, it automatically creates the `jira-software-user` group in JIRA, and assigns it as the default group for the JIRA Software application. When you create a user and select JIRA Software as the application they should have access to, they will be automatically added to the `jira-software-user` group. Once JIRA Software is installed, you can add further groups to the application on the Application access page, and all members of those groups will have access to JIRA Software. If a user is a member of more than one of those groups, they will only consume one user on your JIRA Software license. For more information on creating users, see [Create, edit, or remove a user](#).

You need to have the **JIRA Administrator global permission** to access and edit the Application access page.

Before you get started, know that:

- Each application that's licensed in your JIRA instance is listed on this page.
- Each application will display its total allowed users and users remaining.
- Each application should have at least one group associated with it, and one default group should be selected.

Assign a group to an application

All members of the groups assigned to an application will be able to log in to that application. You may assign multiple groups to any application. The users of these groups will count towards the user tier of your license. If a user is a member of multiple groups assigned to an application, they will only count as one licensed user.

▼ To assign a group to an application:

1. Choose  **> Applications**.
2. Select **Application access** in the left-hand menu. The Application access page displays all your applications and their associated groups, including their default groups.
3. Locate the application you want to assign a specific group to, and click the **Select group...** drop-down.
4. Select the group you wish to add. The group will be added to the application.

Assigning a group to multiple applications

You may assign the same group to several applications. One reason to do this to is ensure members of the group always have full application access. For example, you may have users who requires full access to all your applications. You could create a separate group for them, and add this group to

each application. Care should be taken when assigning a group to multiple applications, as the group members will consume a license for each application. The only exception is JIRA Core. A user with access to any other application automatically has access to JIRA Core, so they will not consume a license for JIRA Core if they belong to a group associated with another application.

Assign a default group to an application

When an application is installed, it automatically creates and assigns a default group to itself. You can manually change the default group, and we strongly recommend that you only have one default group assigned to an application.

▼ To assign a default group:

1. Choose



> Applications.

2. Select **Application access** in the left-hand menu. The Application access page displays all your applications and their associated groups, including their default groups.
3. Check the box in the **Default groups** column for the group you want to assign as a default group. Note you **must** have at least one default group at any time. If you want to change the default group, you must first assign a second default group before you can un-check the box for the current group.

You can also set which application you'd like new users to be added to in JIRA. This is covered in more detail on '[Assign users to groups, project roles, and applications](#)'.

Assigning multiple default groups to an application

We strongly recommend you only have one default group for each application. New users created and assigned application access are added to that application's default group. If this group is also assigned to another application, the new user will also gain access to the additional application, potentially creating a security hole in your system.

For example, if you assign a group called Group A to JIRA Software and make it the default group, all new users added to JIRA Software will be added as members of Group A. If you then add Group A to JIRA Service Desk, all users in Group A will now have full access to both JIRA Software and JIRA Service Desk. This also means that when you create a new user and add them to JIRA Software, they will also gain access to JIRA Service Desk and consume a license for both applications.

Advanced user management

- [Allowing connections to JIRA for user management](#)
- [Diagrams of possible configurations for user management](#)
- [Managing nested groups](#)
- [User management limitations and recommendations](#)

Allowing connections to JIRA for user management

You can allow other applications to connect to your JIRA Server for management of users and groups, and for authentication (verification of a user's login).

Examples of such applications: Atlassian Confluence, FishEye/Crucible, Bamboo, or another JIRA Server.

⚠ Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- Allowing an application to connect to JIRA for user management
- Diagrams of some possible configurations

Allowing an application to connect to JIRA for user management

Subject to certain limitations, you can connect a number of Atlassian applications to a single JIRA application for centralized user management.

When to use this option: You can connect to a server running **JIRA 4.3** or later, **JIRA Software 7.0** or later, **JIRA Core 7.0** or later, or **JIRA Service Desk 3.0** or later. Choose this option as an alternative to Atlassian Crowd, for simple configurations with a limited number of users.

To configure an application to connect to JIRA as a user server:

1. Add the application:
 - a. Choose  **> User Management.**
 - b. Select **JIRA User Server.**
 - c. **Add** an application.
 - d. Enter the **application name** and **password** that the application will use when accessing your JIRA server application.
 - e. Enter the **IP address** or addresses of the application. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
 - f. **Save** the new application.
2. Set up the JIRA user directory in the application:

(For example, see [Connecting Confluence to JIRA applications for user management](#) or [Connecting JIRA applications to another server](#).)

 - a. Log in to the application that is going to connect to your JIRA server for user management.
 - b. Go to the application's '**User Directories**' administration area.
 - c. Add a new directory of type '**Atlassian JIRA**'.
 - d. Define the directory order (see [Managing multiple directories](#)).
3. Create any groups in your JIRA server that are required by the other application. For example, see [Connecting Confluence to JIRA for User Management](#).

Diagrams of some possible configurations

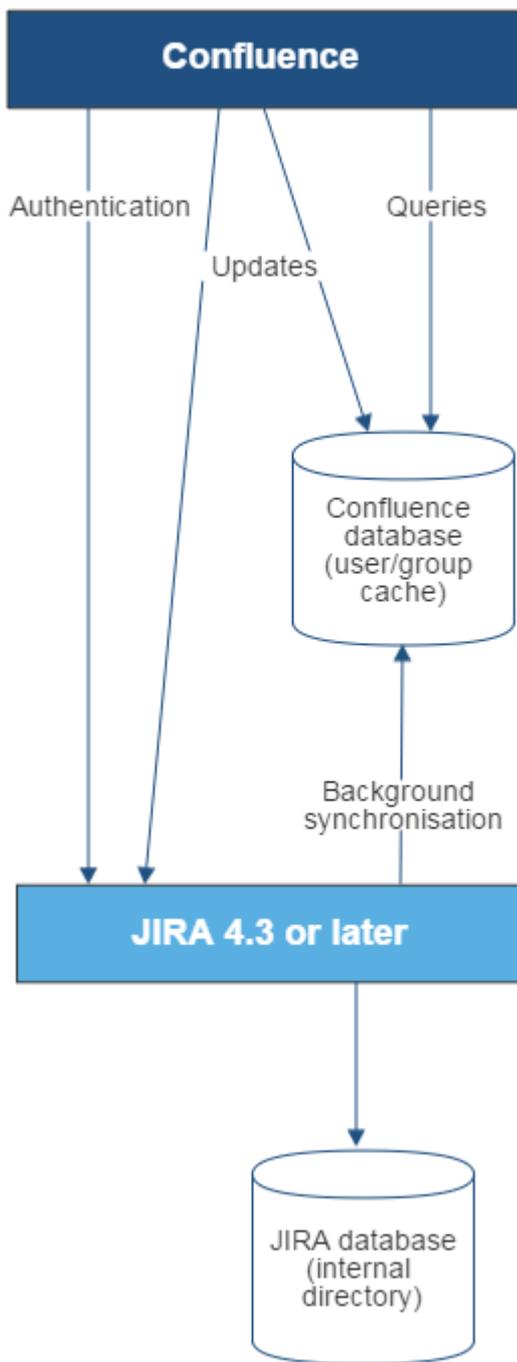


Diagram above: Confluence connecting to JIRA for user management.

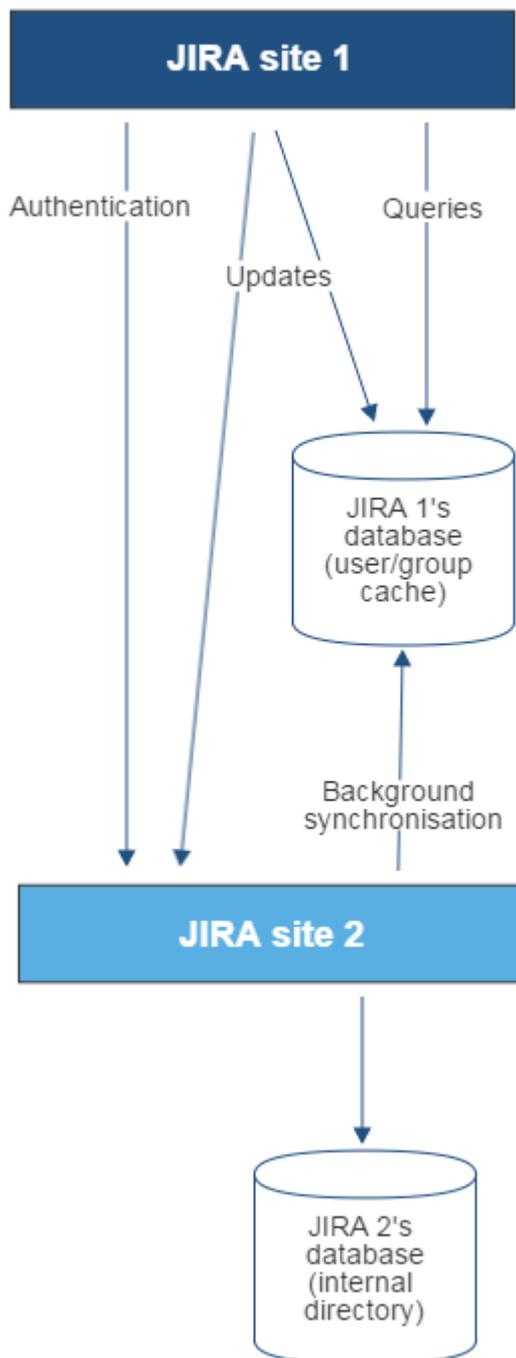


Diagram above: One JIRA site connecting to another for user management. JIRA site 2 does the user management, storing the user data in its internal directory.

Related topics

Configuring user directories

- Configuring the internal directory
- Connecting to an LDAP directory
- Connecting to an internal directory with LDAP authentication
- Connecting to Crowd or another JIRA application for user management
- Managing multiple directories
- Migrating users between user directories
- Synchronizing data from external directories

Diagrams of possible configurations for user management

The aim of these diagrams is to help people understand each directory type at a glance. We have kept the

diagrams simple and conceptual, with just enough information to be correct.

Some things that we do **not** attempt to show:

- In most cases, we do not attempt to show that you can have multiple directory types mapped to JIRA at the same time. We illustrate that fact in just the first two LDAP diagrams.
- We have not included a diagram for Confluence's legacy connection to JIRA database.
- We do not attempt to show all of the possible configurations and layered connections that are available now that you can use JIRA as a directory manager.

On this page:

- JIRA internal directory
- JIRA with read/write connection to LDAP
- JIRA with read-only connection to LDAP, with local groups
- JIRA internal directory with LDAP authentication
- JIRA with LDAP authentication, copy users on first login
- One JIRA instance connecting to another
- Confluence and JIRA connecting to Crowd
- A number of applications connecting to JIRA

JIRA internal directory

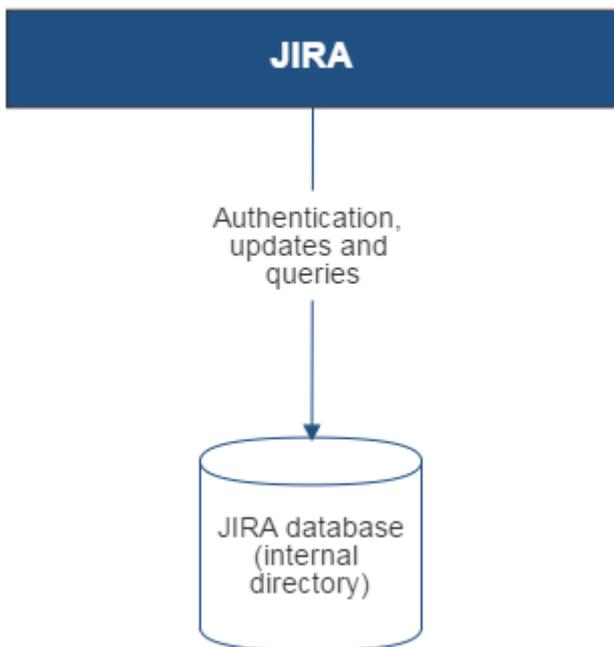


Diagram above: JIRA using its internal directory for user management.

JIRA with read/write connection to LDAP

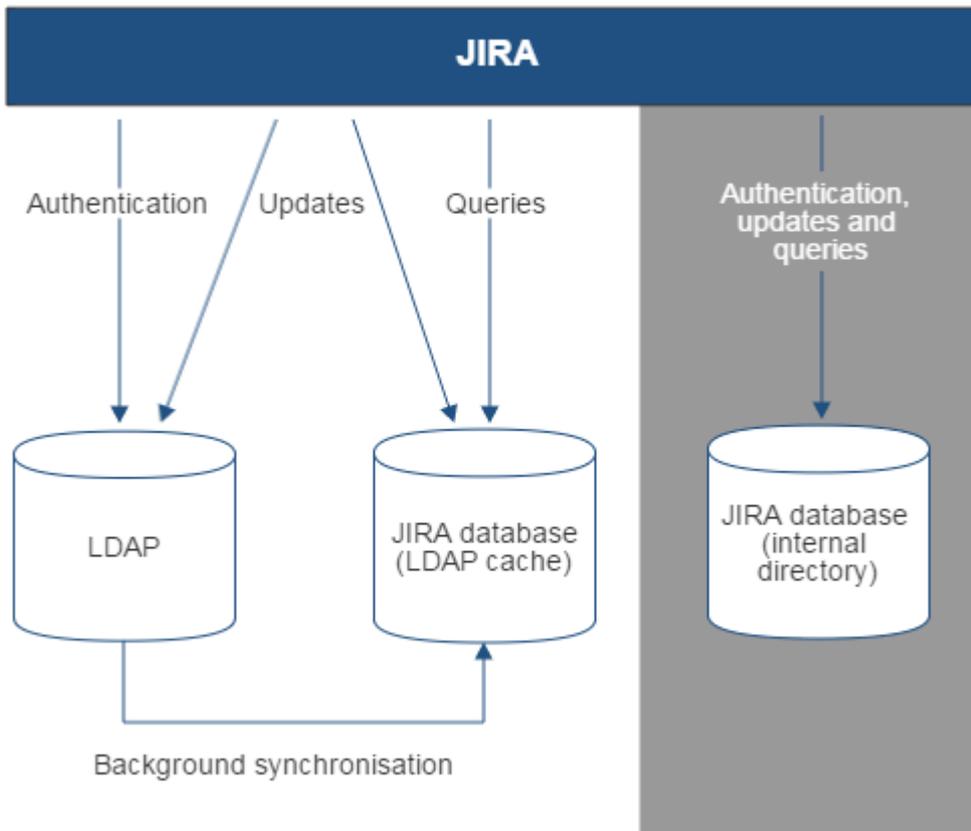


Diagram above: JIRA connecting to an LDAP directory.

JIRA with read-only connection to LDAP, with local groups

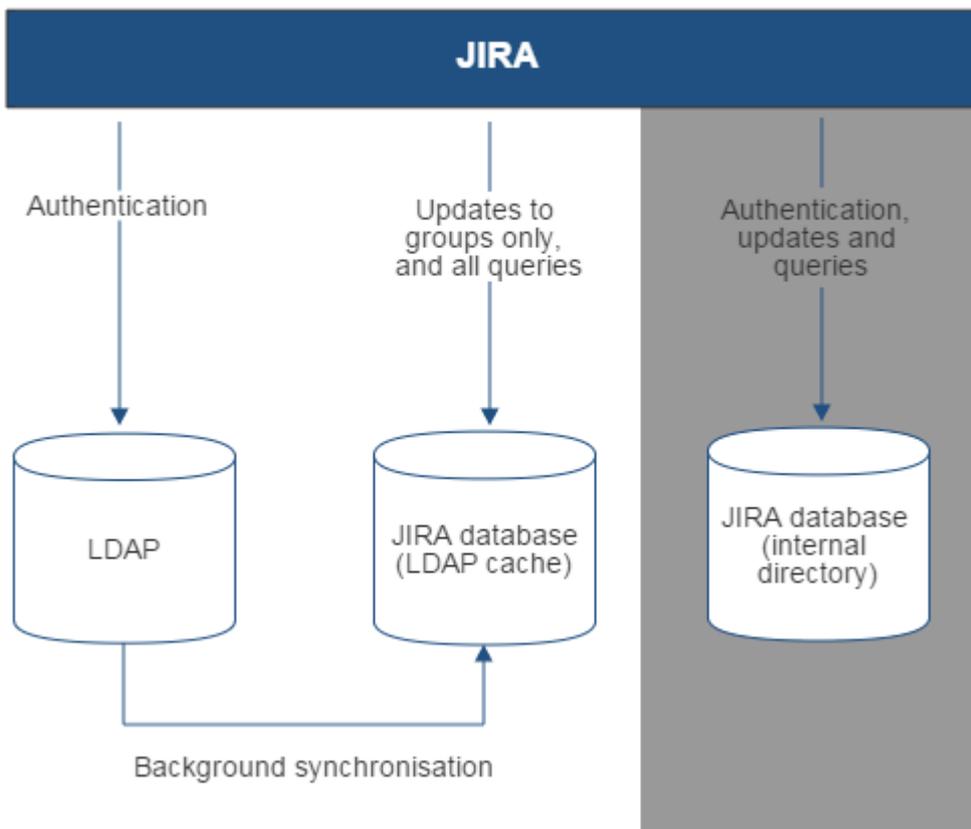


Diagram above: JIRA connecting to an LDAP directory with permissions set to read only and local groups.

JIRA internal directory with LDAP authentication

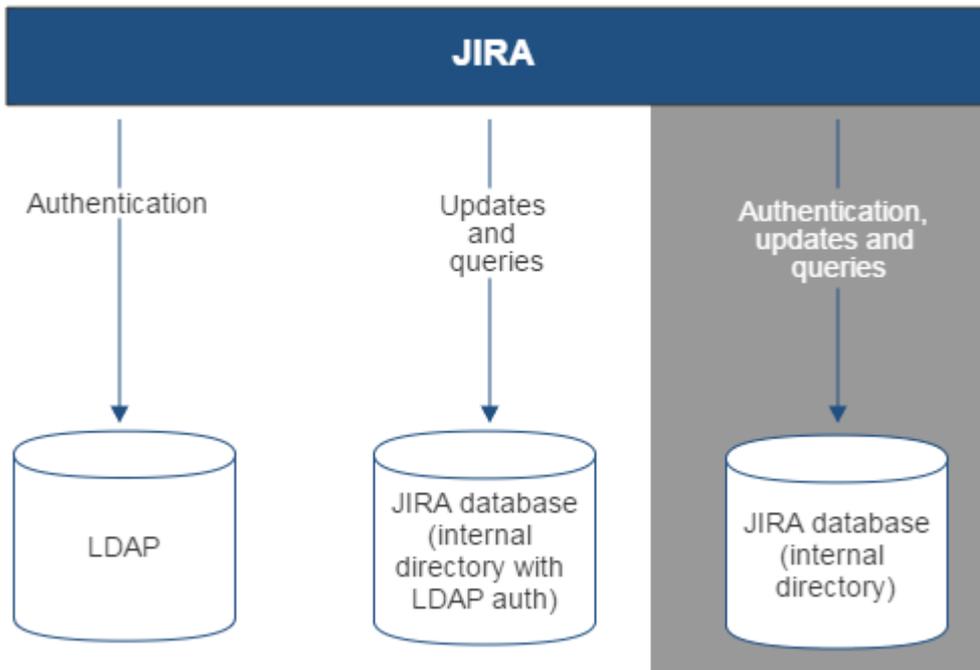


Diagram above: JIRA connecting to an LDAP directory for authentication only.

JIRA with LDAP authentication, copy users on first login

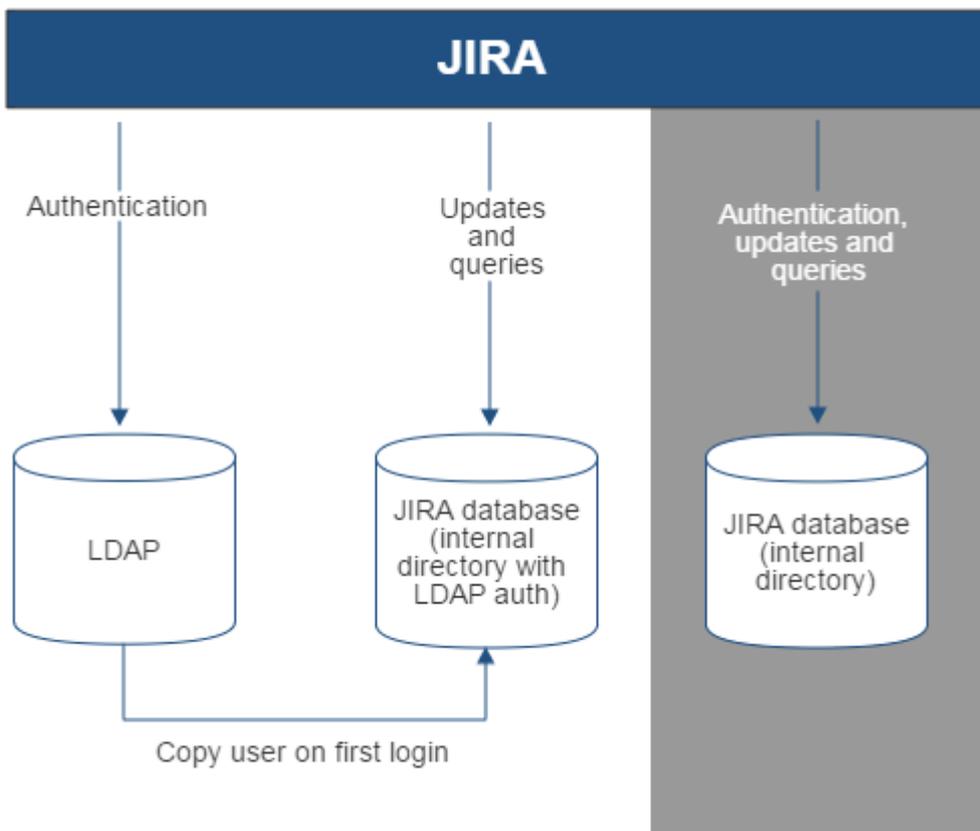


Diagram above: JIRA connecting to an LDAP directory for authentication only, with each user copied to the internal directory when they first log in to JIRA.

One JIRA instance connecting to another

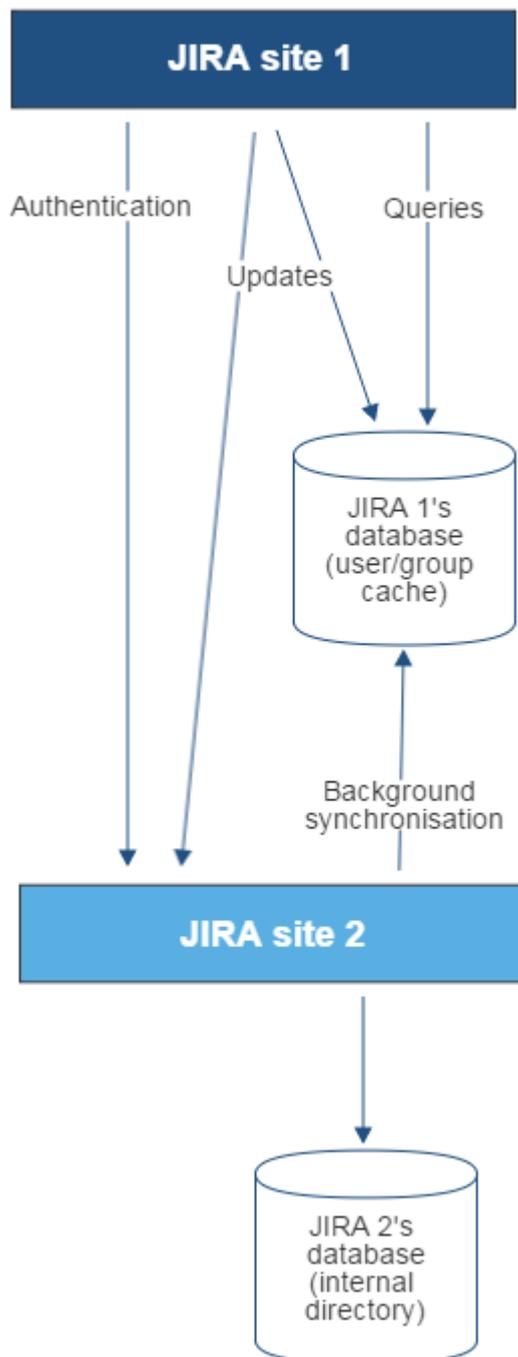


Diagram above: One JIRA site connecting to another for user management. JIRA site 2 does the user management, storing the user data in its internal directory.

Confluence and JIRA connecting to Crowd

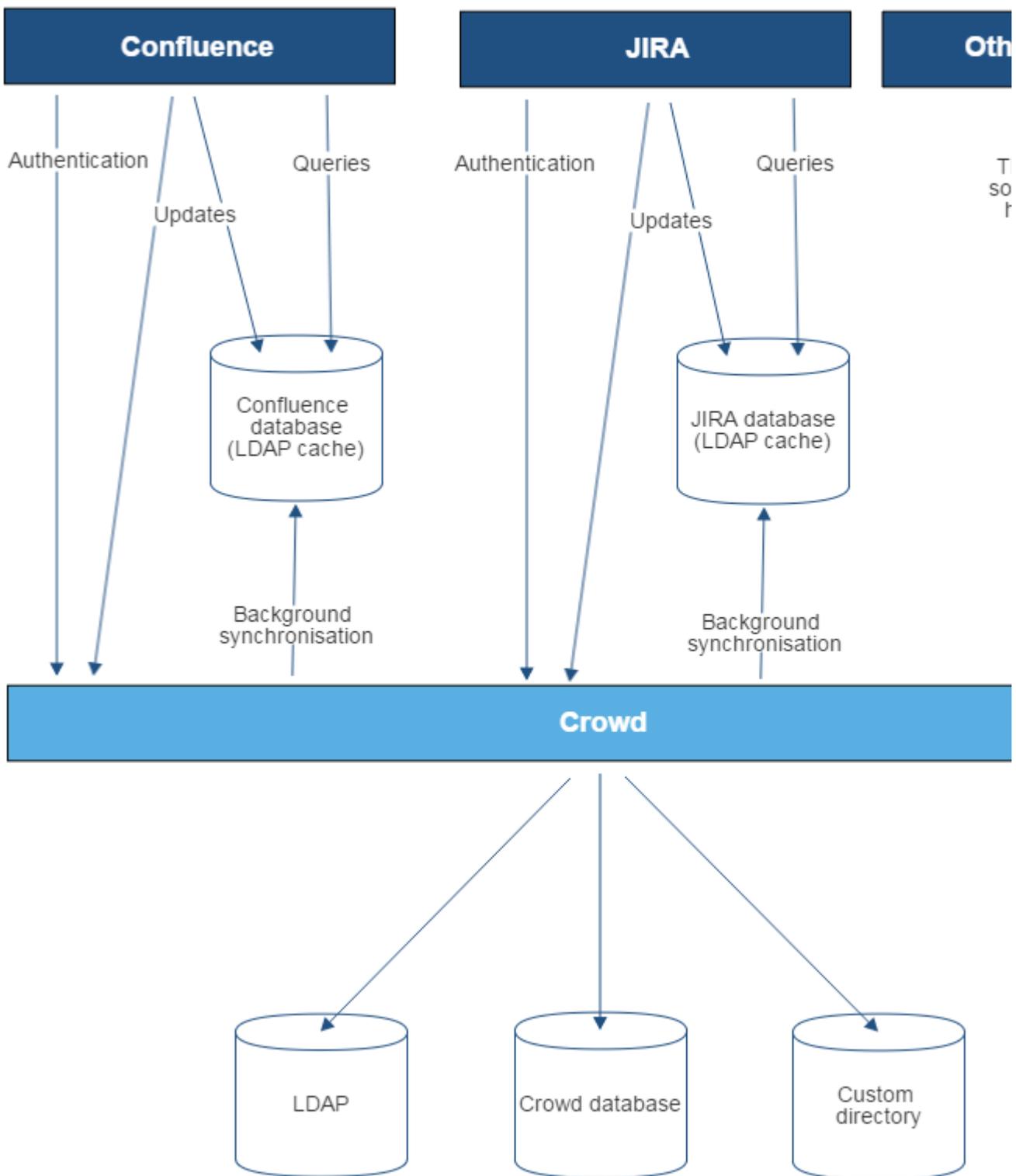


Diagram above: Confluence, JIRA and other applications connecting to Crowd for user management.

A number of applications connecting to JIRA

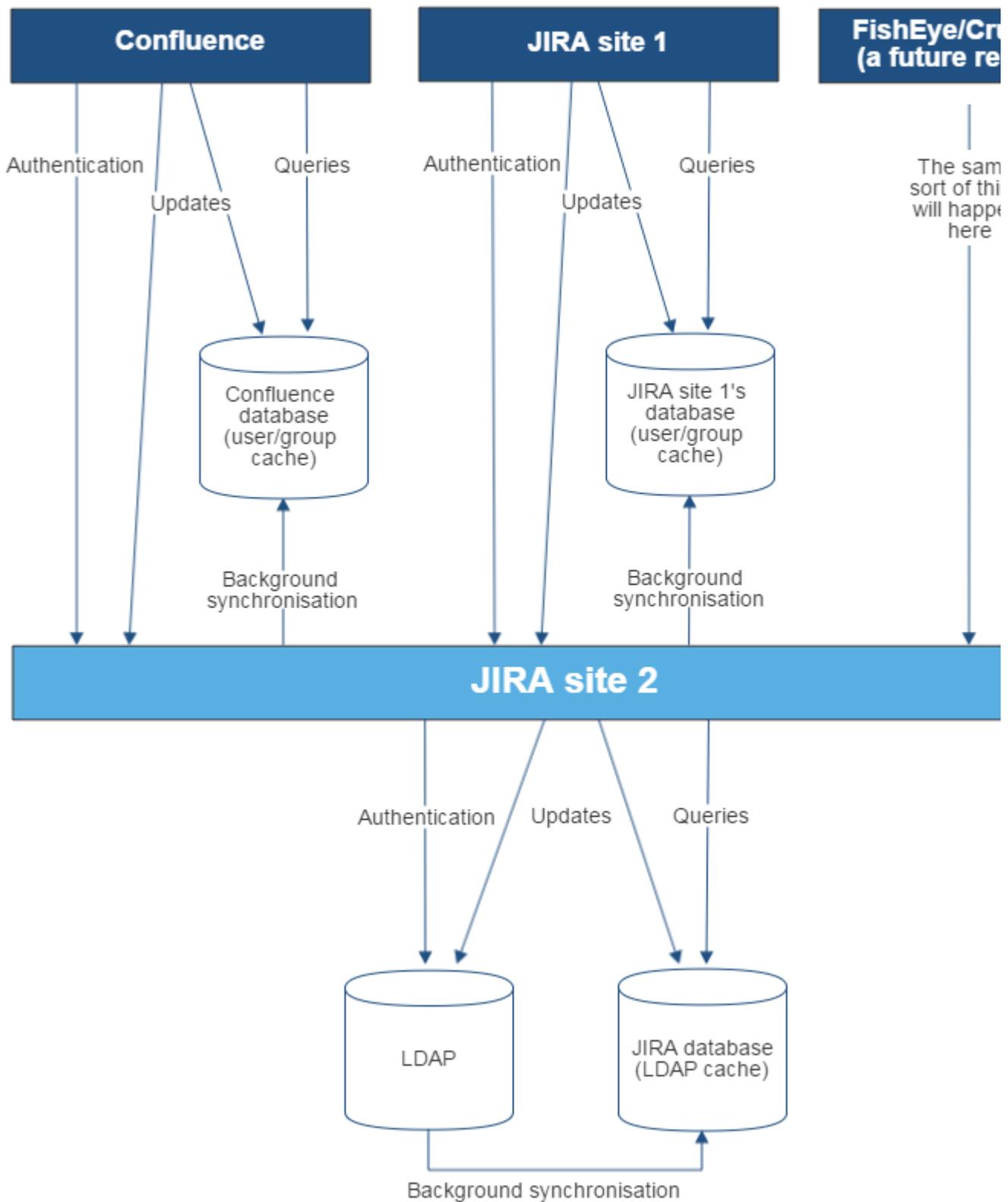


Diagram above: A number of applications connecting to JIRA (site 2) for user management, with JIRA in turn connecting to an LDAP server.

Related topics

Configuring user directories

- Configuring the internal directory
- Connecting to an LDAP directory
- Connecting to an internal directory with LDAP authentication
- Connecting to Crowd or another JIRA application for user management

- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)

Managing nested groups

Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called *nested groups*. Nested groups simplify permissions by allowing sub-groups to inherit permissions from a parent group.

This page describes how JIRA handles nested groups that exist in one or more of your directory servers. Note that if you're using nested groups, you can't use an [LDAP directory for delegated authentication](#).

Enabling Nested Groups

You can enable or disable support for nested groups on each directory individually. Select **User Directories** from the JIRA administration menu, **edit** the directory and select **Enable Nested Groups**. See [Configuring user directories](#).

Notes:

- Make sure that your directory server supports nested groups before you enable nested groups for a specific directory type in JIRA.
- You can nest internal groups in internal groups, or external groups in external groups. You can't nest an internal group in an external group or vice versa.
- Please read the rest of this page to understand what effect nested groups will have on authentication (login) and permissions in JIRA, and what happens when you update users and groups in JIRA.

Effect of Nested Groups

This section explains how nested groups affect logging in, permissions, and viewing and updating users and groups.

Login

When a user logs in, they can access the application if they belong to an authorized group or any of its sub-groups.

Permissions

The user can access a function if they belong to a group that has the necessary permissions, or if they belong to any of its sub-groups.

Viewing lists of group members

If you ask to view the members of a group, you will see all users who are members of the group and all users belonging its sub-groups, consolidated into one list. We call this a *flattened* list.

You can't view or edit the nested groups themselves, or see that one group is a member of another group.

On this page:

- [Enabling Nested Groups](#)
- [Effect of Nested Groups](#)
 - [Login](#)
 - [Permissions](#)
 - [Viewing lists of group members](#)
 - [Adding and updating group membership](#)
- [Examples](#)
 - [Example 1: User is member of sub-group](#)
 - [Example 2: Sub-groups as members of the jira-developers group](#)
- [Notes](#)

Adding and updating group membership

If you add a user to a group, the user is added to the named group and not to any other groups.

If you try to remove a user from a **flattened** list, the following will happen:

- If the user is a member of the top group in the hierarchy of groups in the flattened list, the user is removed from the top group.
- Otherwise, you see an error message stating that the user is not a direct member of the group.

Examples

Example 1: User is member of sub-group

Imagine the following two groups exist in your directory server:

- staff
- marketing

Memberships:

- The **marketing** group is a member of the **staff** group.
- User **jsmith** is a member of **marketing**.

You will see that **jsmith** is a member of both **marketing** and **staff**. You will not see that the two groups are nested. If you assign permissions to the **staff** group, then **jsmith** will get those permissions.

Example 2: Sub-groups as members of the jira-developers group

In an LDAP directory server, we have the groups **engineering-group** and **techwriters-group**. We want to grant both groups developer-level access to the JIRA. We will have a group called **jira-developers** that has developer-level access.

- Add a group called **jira-developers**.
- Add the **engineering-group** as a sub-group of **jira-developers**.
- Add the **techwriters-group** as a sub-group of **jira-developers**.

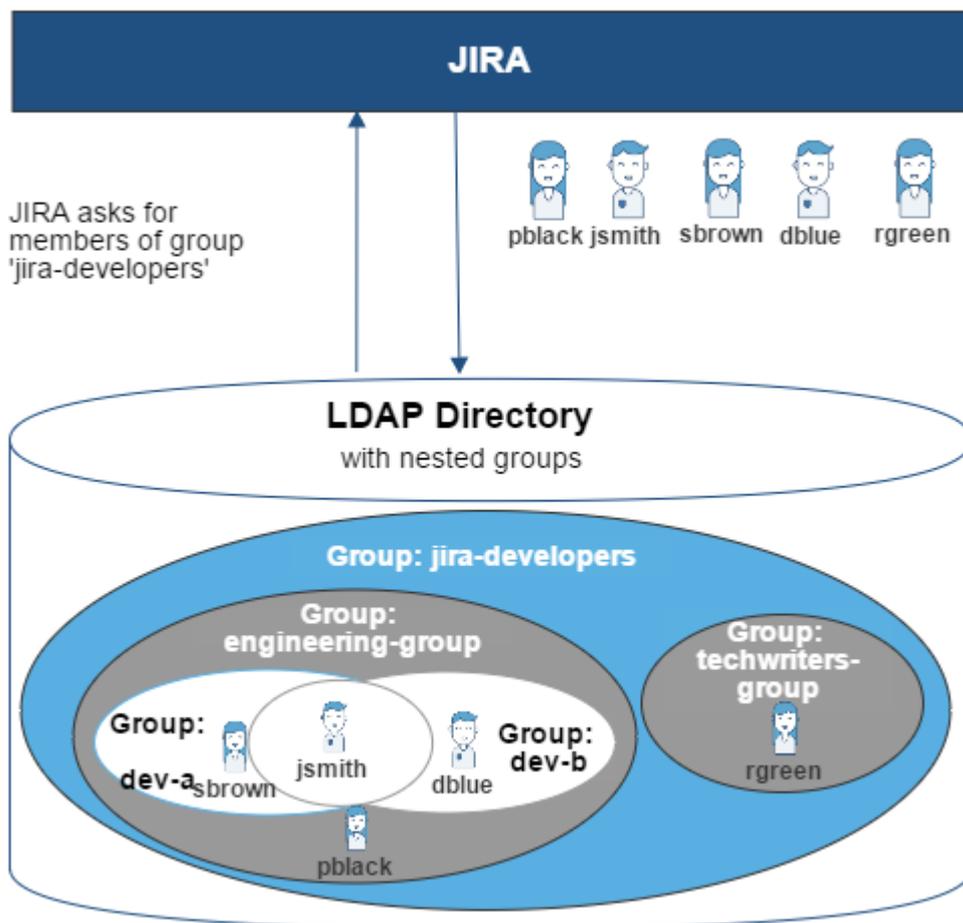
Group memberships are now:

- **jira-developers** — sub-groups: **engineering-group**, **techwriters-group**
- **engineering-group** — sub-groups: **dev-a**, **dev-b**; users: **pblack**
- **dev-a** — users: **jsmith**, **sbrown**
- **dev-b** — users: **jsmith**, **dblue**
- **techwriters-group** — users: **rgreen**

When the JIRA application requests a list of users in the **jira-developers** group, it receives the following list:

- **pblack**
- **jsmith**
- **sbrown**
- **dblue**
- **rgreen**

Diagram: Sub-groups as members of the jira-developers group



Notes

- **Possible impact on performance.** Enabling nested groups may result in slower user searches.
- **Definition of nested groups in LDAP.** In an LDAP directory, a nested group is a child group entry whose DN (Distinguished Name) is referenced by an attribute contained within a parent group entry. For example, a parent group **Group One** might have an `objectClass=group` attribute and one or more `member=DN` attributes, where the DN can be that of a user *or* that of a group elsewhere in the LDAP tree:

```
member=CN=John Smith,OU=Users,OU=OrgUnitA,DC=sub,DC=domain
member=CN=Group
Two,OU=OrgUnitBGroups,OU=OrgUnitB,DC=sub,DC=domain
```

Related topics

Configuring user directories

- Configuring the internal directory
- Connecting to an LDAP directory
- Connecting to an internal directory with LDAP authentication
- Connecting to Crowd or another JIRA application for user management
- Managing multiple directories
- Migrating users between user directories
- Synchronizing data from external directories

User management limitations and recommendations

This page describes the optimal configurations and limitations that apply to user management in JIRA.

On this page:

- Recommendations for connecting to LDAP
- Recommendations for connecting to another JIRA server

General recommendations

- **Avoid duplicate usernames across directories.** If you are connecting to more than one user directory, we recommend that you ensure the usernames are unique to one directory. For example, we do not recommend that you have a user `jsmith` in both 'Directory1' and 'Directory2'. The reason is the potential for confusion, especially if you swap the order of the directories. Changing the directory order can change the user that a given username refers to.
- **Be careful when deleting users in remote directories.** If you are connecting to an LDAP directory, a Crowd directory or a remote JIRA directory, please take care when deleting users from the remote directory. If you delete a user that is associated with data in JIRA, this will cause problems in JIRA. We recommend that you perform all user management in JIRA, because the JIRA UI will prevent the deletion of a user if there are issues assigned to the user, reported by the user or the user is a project lead.

Recommendations for connecting to LDAP

Please consider the following limitations and recommendations when connecting to an LDAP user directory.

Optimal Number of Users and Groups in your LDAP Directory

The connection to your LDAP directory provides powerful and flexible support for connecting to, configuring and managing LDAP directory servers. To achieve optimal performance, a background synchronization task loads the required users and groups from the LDAP server into the application's database, and periodically fetches updates from the LDAP server to keep the data in step. The amount of time needed to copy the users and groups rises with the number of users, groups, and group memberships. For that reason, we recommended a maximum number of users and groups as described below.

This recommendation affects connections to LDAP directories:

- Microsoft Active Directory
- All other LDAP directory servers

The following LDAP configurations are **not** affected:

- Internal directories with LDAP authentication
- LDAP directories configured for 'Authentication Only, Copy User On First Login'

Please choose one of the following solutions, depending on the number of users, groups and memberships in your LDAP directory.

Your environment	Recommendation
Up to 10 000 (ten thousand) users, 1000 (one thousand) groups, and 20 (twenty) groups per user	Choose the ' LDAP ' or ' Microsoft Active Directory ' directory type. You can make use of the full synchronization option. Your application's database will contain all the users and groups that are in your LDAP server.

More than the above	Use LDAP filters to reduce the number of users and groups visible to the synchronization task.
---------------------	--

Our Test Results

We performed internal testing of synchronization with an AD server on our local network consisting of 10 000 users, 1000 groups and 200 000 memberships.

We found that the initial synchronization took about 5 minutes. Subsequent synchronizations with 100 modifications on the AD server took a couple of seconds to complete.

Please keep in mind that a number of factors come into play when trying to tune the performance of the synchronization process, including:

- **Size of userbase.** Use LDAP filters to keep this to the minimum that suits your requirements.
- **Type of LDAP server.** We currently support change detection in AD, so subsequent synchronizations are much faster for AD than for other LDAP servers.
- **Network topology.** The further away your LDAP server is from your application server, the more latent LDAP queries will be.
- **Database performance.** As the synchronization process caches data in the database, the performance of your database will affect the performance of the synchronization.
- **JVM heap size.** If your heap size is too small for your userbase, you may experience heavy garbage collection during the synchronization process which could in turn slow down the synchronization.

Redundant LDAP is Not Supported

The LDAP connections do not support the configuration of two or more LDAP servers for redundancy (automated failover if one of the servers goes down).

Specific Notes for Connecting to Active Directory

When the application synchronizes with Active Directory (AD), the synchronization task requests only the changes from the LDAP server rather than the entire user base. This optimizes the synchronization process and gives much faster performance on the second and subsequent requests.

On the other hand, this synchronization method results in a few limitations:

1. **Externally moving objects out of scope or renaming objects causes problems in AD.** If you move objects out of scope in AD, this will result in an inconsistent cache. We recommend that you do not use the external LDAP directory interface to move objects out of the scope of the sub-tree, as defined on the application's directory configuration screen. If you do need to make structural changes to your LDAP directory, manually synchronize the directory cache after you have made the changes to ensure cache consistency.
2. **Synchronizing between AD servers is not supported.** Microsoft Active Directory does not replicate the uSNChanged attribute across instances. For that reason, we do not support connecting to different AD servers for synchronization. (You can of course define multiple different directories, each pointing to its own respective AD server.)
3. **Synchronizing with AD servers behind a load balancer is not supported.** As with synchronizing between two different AD servers, Microsoft Active Directory does not replicate the uSNChanged attribute across instances. For that reason, we do not support connecting to different AD servers even when they are load balanced. You will need to select one server (preferably one that is local) to synchronize with instead of using the load balancer.
4. **You must restart the application after restoring AD from backup.** On restoring from backup of an AD server, the uSNChanged timestamps are reverted to the backup time. To avoid the resulting confusion, you will need to flush the directory cache after a Active Directory restore operation.
5. **Obtaining AD object deletions requires administrator access.** Active Directory stores deleted objects in a special container called cn=Deleted Objects. By default, to access this container you need to connect as an administrator and so, for the synchronization task to be aware of deletions, you must use administrator credentials. Alternatively, it is possible to change the permissions on the cn=Deleted Objects container. If you wish to do so, please see [this Microsoft KB article](#).
6. **The User DN used to connect to AD must be able to see the uSNChanged attribute.** The synchronization task relies on the uSNChanged attribute to detect changes, and so must be in the appropriate AD security groups to see this attribute for all LDAP objects in the subtree.

Recommendations for connecting to another JIRA server

Please consider the following limitations and recommendations when connecting to a JIRA server for user management.

Single Sign-On Across Multiple Applications is Not Supported

When you connect to a JIRA application for user management, you will not have single sign-on across the applications connected in this way. JIRA, when acting as a directory manager, does not support SSO.

Custom Application Connectors are Not Supported

JIRA applications, Confluence, FishEye, Crucible and Bamboo can connect to a JIRA server for user management. Custom application connectors will need to use the new REST API.

Custom Directories are Not Supported

Earlier versions of JIRA supported OSUser Providers. It was therefore possible write a special provider to obtain user information from any external user directory. This is no longer the case.

Load on your JIRA instance

If your JIRA instance is already under high load, then using it as a User Server will increase that load.

JIRA Cloud applications not supported

You cannot use JIRA Cloud applications to manage standalone users. Cloud users and users within your self-hosted Atlassian applications need to be managed separately.

Recommendations

Your environment	Recommendation
<p>If all the following are true:</p> <ul style="list-style-type: none"> • Your JIRA application is not under high load. • You want to share user and group management across just a few applications, such as one JIRA Software server and one Confluence server, or two JIRA servers. • You do not need single sign-on (SSO) between your JIRA application and Confluence, or between two JIRA servers. • You do not have custom application connectors. Or, if you do have them, you are happy to convert them to use the new REST API. • You are happy to shut down all your servers when you need to upgrade your JIRA application. 	<p>Your environment meets the optimal requirements for using a JIRA application for user management.</p>

<p>If one or more of the following are true:</p> <ul style="list-style-type: none"> • If your JIRA application is already under high load. • You want to share user and group management across more than 5 applications. • You need single sign-on (SSO) across multiple applications. • You have custom applications integrated via the Crowd SOAP API, and you cannot convert them to use the new REST API. • You are not happy to shut down all your servers when you need to upgrade JIRA. 	<p>We recommend that you install Atlassian Crowd for user management and SSO.</p>
<p>If you are considering creating a custom directory connector to define your own storage for users and groups...</p>	<p>Please see if one of the following solutions will work for you:</p> <ul style="list-style-type: none"> • If you have written a custom provider to support a specific LDAP schema, please check the supported LDAP schemas to see if you can use one of them instead. • If you have written a custom provider to support nested groups, please consider enabling nested groups in the supported directory connectors instead. • If you have written a custom provider to connect to your own database, please consider loading the data into the application's database instead. • If you need to keep the custom directory connection, please consider whether Atlassian Crowd meets your requirements. See the documentation on Creating a Custom Directory Connector.

Related topics

[Connecting to an LDAP directory](#)
[Connecting to Crowd or another JIRA server for user management](#)
[Configuring user directories](#)

Configuring user directories

A user directory is a place where you store information about users and groups. User information includes the person's full name, username, password, email address and other personal information. Group information includes the name of the group, the users that belong to the group, and possibly groups that belong to other groups.

The **internal** directory stores user and group information in the JIRA database. You can also connect to **external** user directories, and to Atlassian **Crowd** and **JIRA** as directory managers.

On this page:

- [Configuring user directories in JIRA](#)
- [Connecting to a directory](#)
- [Updating directories](#)

Configuring user directories in JIRA

1. Log in as a user with the '**JIRA Administrators**' [global permission](#).
2. Choose



> **User Management** > **User Directories**.

Connecting to a directory

You can add the following types of directory servers and directory managers:

- JIRA's internal directory. See [Configuring the internal directory](#).
- Microsoft Active Directory. See [Connecting to an LDAP directory](#).
- Various other LDAP directory servers. See [Connecting to an LDAP Directory](#).
- An LDAP directory for delegated authentication. See [Connecting to an Internal Directory with LDAP Authentication](#).
- Atlassian Crowd. See [Connecting to Crowd or another JIRA server for user management](#).
- Another JIRA server. See [Connecting to Crowd or another JIRA server for user management](#).

You can add as many external user directories as you need. Note that you can define the **order** of the directories. This determines which directory JIRA will search first, when looking for user and group information. See [Managing multiple directories](#).

Updating directories

Limitations when Editing Directories

You cannot edit, disable or remove the directory your user belongs to. This precaution is designed to prevent administrators from locking themselves out of the application by changing the directory configuration in a way that prevents them logging in or removes their administration permissions.

This limitation applies to all directory types. For example:

- You cannot disable the internal directory if your user is an internal user.
- You cannot disable or remove an LDAP or a Crowd directory if your user comes from that directory.

In some situations, reordering the directories will change the directory that the current user comes from, if a user with the same username happens to exist in both. This behavior can be used in some cases to create a copy of the existing configuration, move it to the top, then remove the old one. Note, however, that duplicate usernames are not a supported configuration.

You cannot remove the internal directory. This precaution aligns with the recommendation below that you always keep an administrator account active in the internal directory.

Recommendations

The recommended way to edit directory configurations is to log in as an internal user when making changes to external directory configuration.

 We recommend that you keep either an administrator or system administrator user active in your internal directory for troubleshooting problems with your user directories.

Enabling, disabling, and removing directories

You can enable or disable a directory at any time. If you disable a directory, your configuration details will remain but the application will not recognize the users and groups in that directory.

You have to disable a directory before you can remove it. Removing a directory will remove the details from the database.

[Screenshot: Configuring user directories](#)

User Directories

The table below shows the user directories currently configured for JIRA.

The order of the directories is the order in which they will be searched for users and groups. Changes to users and groups will be made in the first directory where JIRA has permission to make changes. It is recommended that users only exist in a single directory.

Directory Name	Type	Order	Operations
JIRA Internal Directory	Internal	↑ ↓	
LDAP server	OpenLDAP (Read-Write)	↑ ↓	Disable Edit Synchronise Last synchronised at 17/01/11 10:31 AM (took 72s).

[Add Directory](#)

In situations where users are unable to change their passwords, check that a Delegated Authentication Directory is not the highest in the order of User Directories. As a workaround, you can change the order of User Directories, or alternatively use a connection to a LDAP directory instead.

Related topics

- [Configuring the internal directory](#)
- [Connecting to an LDAP directory](#)
- [Connecting to an internal directory with LDAP authentication](#)
- [Connecting to Crowd or another JIRA application for user management](#)
- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)
- [User management](#)

Configuring the internal directory

The internal directory stores user and group information in the JIRA database.

The internal directory is enabled by default at installation. When you create the first administrator during the setup procedure, that administrator's username and other details are stored in the internal directory.

If needed, you can configure one or more additional user directories. This is useful if you want to grant access to users and groups that are stored in a corporate directory or other directory server.

On this page:

- [Settings](#)
- [Diagram of possible configuration](#)

Settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.

Diagram of possible configuration

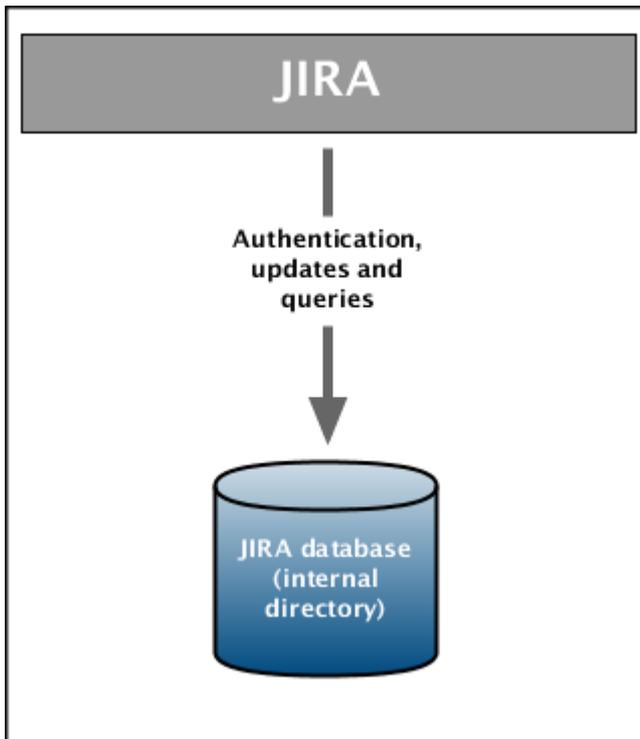


Diagram above: JIRA using its internal directory for user management.

Related topics

Configuring user directories

- [Configuring the internal directory](#)
- [Connecting to an LDAP directory](#)
- [Connecting to an internal directory with LDAP authentication](#)
- [Connecting to Crowd or another JIRA application for user management](#)
- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)

Connecting to an LDAP directory

You can connect your JIRA application to an LDAP directory for authentication, user and group management. You will need to log in as a user with the 'JIRA System Administrators' [global permission](#) to access the Settings menu below.

An LDAP directory is a collection of data about users and groups. LDAP (Lightweight Directory Access Protocol) is an Internet protocol that web applications can use to look up information about those users and groups from the LDAP server.

We provide built-in connectors for the most popular LDAP directory servers:

- [Microsoft Active Directory](#)
- [Apache Directory Server \(ApacheDS\)](#)
- [Apple Open Directory](#)
- [Fedora Directory Server](#)
- [Novell eDirectory](#)
- [OpenDS](#)
- [OpenLDAP](#)
- [OpenLDAP Using Posix Schema](#)
- [Posix Schema for LDAP](#)
- [Sun Directory Server Enterprise Edition \(DSEE\)](#)
- [A generic LDAP directory server](#)

When to use this option: Connecting to an LDAP directory server is useful if your users and groups are stored in a corporate directory. When configuring the directory, you can choose to make it read only, read only with local groups, or read/write. If you choose read/write, any changes made to user and group information in the application will also update the LDAP directory.

Learn more about [synchronizing data from external directories](#).

On this page:

- [Connecting to an LDAP Directory in JIRA](#)
- [Server settings](#)
- [Schema settings](#)
- [Permission settings](#)
 - [Adding user to groups automatically](#)
- [Advanced settings](#)
- [User schema settings](#)
- [Group schema settings](#)
- [Membership schema settings](#)
- [Diagrams of some possible configurations](#)

Connecting to an LDAP Directory in JIRA

1. Choose



> **User Management.**

2. Choose **User Directories**.
3. **Add** a directory and select one of these types:
 - **'Microsoft Active Directory'** – This option provides a quick way to select AD, because it is the most popular LDAP directory type.
 - **'LDAP'** – You will be able to choose a specific LDAP directory type on the next screen.
4. Enter the values for the settings, as described below.
5. Save the directory settings.
6. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen. Here is a summary of how the directory order affects the processing:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

For details, see [Managing multiple directories](#).

Notes:

- For this configuration, every time user logs in (i.e. first and subsequent times), the user's data in JIRA will be updated from the user's data in LDAP. This includes username, display name, email and group

memberships. However for group memberships, only the following applies:

- direct groups only (i.e. not nested groups) are synchronized from LDAP.
- only groups that are already present in JIRA are synchronized, i.e. groups are not added/removed, and group hierarchies are not synchronized.

Learn more about [synchronizing data from external directories](#).

Server settings

Setting	Description
Name	Enter a meaningful name to help you identify the LDAP directory server. Examples: <ul style="list-style-type: none"> • Example Company Staff Directory • Example Company Corporate LDAP
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for many of the options on the rest of screen. Examples: <ul style="list-style-type: none"> • Microsoft Active Directory • OpenDS • And more.
Hostname	The host name of your directory server. Examples: <ul style="list-style-type: none"> • ad.example.com • ldap.example.com • opensds.example.com
Port	The port on which your directory server is listening. Examples: <ul style="list-style-type: none"> • 389 • 10389 • 636 (for example, for SSL)
Use SSL	Check this if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: <ul style="list-style-type: none"> • cn=administrator,cn=users,dc=ad,dc=example,dc=com • cn=user,dc=domain,dc=name • user@domain.name <div style="border: 1px solid red; padding: 10px; margin-top: 10px;"> <p>By default, all users can read the uSNChanged attribute; however, only administrators or users with relevant permissions can access the Deleted Objects container. The specific privileges required by the user to connect to LDAP are "Bind" and "Read" (user info, group info, group membership, update sequence number, deleted objects), which the user can obtain by being a member of the Active Directory's built-in administrators group.</p> <p>Note that the incremental sync will fail silently if the Active Directory is accessed by a user without these privileges. This has been reported as CWD-3093.</p> </div>

Password	<p>The password of the user specified above.</p> <p>Note: Connecting to an LDAP server requires that this application log in to the server with the username and password configured here. As a result, this password cannot be one-way hashed - it must be recoverable in the context of this application. The password is currently stored in the database in plain text without obfuscation. To guarantee its security, you need to ensure that other processes do not have OS-level read permissions for this application's database or configuration files.</p>
----------	---

Schema settings

Setting	Description
Base DN	<p>The root distinguished name (DN) to use when running queries against the directory server. Examples:</p> <ul style="list-style-type: none"> o=example,c=com cn=users,dc=ad,dc=example,dc=com For Microsoft Active Directory, specify the base DN in the following format: dc=domain1,dc=local. You will need to replace the domain1 and local for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the the LDAP structure of your server.
Additional User DN	<p>This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example:</p> <ul style="list-style-type: none"> ou=Users
Additional Group DN	<p>This value is used in addition to the base DN when searching and loading groups. If no value is supplied, the subtree search will start from the base DN. Example:</p> <ul style="list-style-type: none"> ou=Groups

If no value is supplied for **Additional User DN** or **Additional Group DN** this will cause the subtree search to start from the base DN and, in case of huge directory structure, could cause performance issues for login and operations that rely on login to be performed.

Permission settings

Note: You can only assign LDAP users to local groups when 'External User Management' is not selected.

Setting	Description
Read Only	LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens.
Read Only, with Local Groups	<p>LDAP users, groups and memberships are retrieved from your directory server and can only be modified via your directory server. You cannot modify LDAP users, groups or memberships via the application administration screens. However, you can add groups to the internal directory and add LDAP users to those groups.</p> <p>Note for Confluence users: Users from LDAP are added to groups maintained in Confluence's internal directory the first time they log in. This is only done once per user. There is a known issue with Read Only, with Local Groups in Confluence that may apply to you. See</p> <p>CONFSERVER-28621 - User Loses all Local Group Memberships If LDAP Sync is Unable to find the User, but the User appears again in subsequent syncs CLOSED</p>

Read/Write	LDAP users, groups and memberships are retrieved from your directory server. When you modify a user, group or membership via the application administration screens, the changes will be applied directly to your LDAP directory server. Please ensure that the LDAP user specified for the application has modification permissions on your LDAP directory server.
------------	---

Adding user to groups automatically

Setting	Description
Default Group Memberships	<p><i>Option available in Confluence 3.5 and later, and JIRA 4.3.3 and later.</i> This field appears if you select the 'Read Only, with Local Groups' permission. If you would like users to be automatically added to a group or groups, enter the group name(s) here. To specify more than one group, separate the group names with commas.</p> <p><i>In Confluence 3.5 to Confluence 3.5.1:</i> Each time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added locally.</p> <p><i>In Confluence 3.5.2 and later, and JIRA 4.3.3 and later:</i> The first time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added locally. On subsequent logins, the username will <i>not</i> be added automatically to any groups. This change in behavior allows users to be removed from automatically-added groups. In Confluence 3.5 and 3.5.1, they would be re-added upon next login.</p> <p>Please note that there is no validation of the group names. If you mis-type the group name, authorization failures will result – users will not be able to access the applications or functionality based on the intended group name.</p> <p>Examples:</p> <ul style="list-style-type: none"> • confluence-users • confluence-users , jira-administrators , jira-core-users

Advanced settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Some directory servers allow you to define a group as a member of another group. Groups in such a structure are called <i>nested groups</i> . Nested groups simplify permissions by allowing sub-groups to inherit permissions from a parent group.
Manage User Status Locally	If true, you can activate and deactivate users in Crowd independent of their status in the directory server.
Filter out expired users	If true, user accounts marked as expired in ActiveDirectory will be automatically removed. For cached directories, the removal of a user will occur during the first synchronization after the account's expiration date. Note: This is available in Embedded Crowd 2.0.0 and above, but not available in the 2.0.0 m04 release.
Use Paged Results	Enable or disable the use of the LDAP control extension for simple paging of search results. If paging is enabled, the search will retrieve sets of data rather than all of the search results at once. Enter the desired page size – that is, the maximum number of search results to be returned per page when paged results are enabled. The default is 1000 results.

Follow Referrals	Choose whether to allow the directory server to redirect requests to other servers. This option uses the node referral (JNDI lookup <code>java.naming.referral</code>) configuration setting. It is generally needed for Active Directory servers configured without proper DNS, to prevent a 'javax.naming.PartialResultException: Unprocessed Continuation Reference(s)' error.
Naive DN Matching	<p>If your directory server will always return a consistent string representation of a DN, you can enable naive DN matching. Using naive DN matching will result in a significant performance improvement, so we recommend enabling it where possible.</p> <p>This setting determines how your application will compare DNs to determine if they are equal.</p> <ul style="list-style-type: none"> • If this checkbox is selected, the application will do a direct, case-insensitive, string comparison. This is the default and recommended setting for Active Directory, because Active Directory guarantees the format of DNs. • If this checkbox is not selected, the application will parse the DN and then check the parsed version.
Enable Incremental Synchronization	<p>Enable incremental synchronization if you only want changes since the last synchronization to be queried when synchronizing a directory.</p> <p>Please be aware that when using this option, the user account configured for synchronization must have read access to:</p> <ul style="list-style-type: none"> • The <code>uSNChanged</code> attribute of all users and groups in the directory that need to be synchronized. • The objects and attributes in the Active Directory deleted objects container. <p>If at least one of these conditions is not met, you may end up with users who are added to (or deleted from) the Active Directory not being respectively added (or deleted) in the application.</p> <p>This setting is only available if the directory type is set to "Microsoft Active Directory".</p>
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.
Read Timeout (seconds)	The time, in seconds, to wait for a response to be received. If there is no response within the specified time period, the read attempt will be aborted. A value of 0 (zero) means there is no limit. The default value is 120 seconds.
Search Timeout (seconds)	The time, in seconds, to wait for a response from a search operation. A value of 0 (zero) means there is no limit. The default value is 60 seconds.
Connection Timeout (seconds)	<p>This setting affects two actions. The default value is 0.</p> <ul style="list-style-type: none"> • The time to wait when getting a connection from the connection pool. A value of 0 (zero) means there is no limit, so wait indefinitely. • The time, in seconds, to wait when opening new server connections. A value of 0 (zero) means that the TCP network timeout will be used, which may be several minutes.

User schema settings

Setting	Description
User Object Class	<p>This is the name of the class used for the LDAP user object. Example:</p> <ul style="list-style-type: none"> • <code>user</code>

User Object Filter	<p>The filter to use when searching user objects. Example:</p> <ul style="list-style-type: none"> • <code>(&(objectCategory=Person)(sAMAccountName=*))</code> <p>More examples can be found in our knowledge base. See How to write LDAP search filters.</p>
User Name Attribute	<p>The attribute field to use when loading the username. Examples:</p> <ul style="list-style-type: none"> • <code>cn</code> • <code>sAMAccountName</code> <p>NB: In Active Directory, the 'sAMAccountName' is the 'User Logon Name (pre-Windows 2000)' field. The User Logon Name field is referenced by 'cn'.</p>
User Name RDN Attribute	<p>The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example:</p> <ul style="list-style-type: none"> • <code>cn</code>
User First Name Attribute	<p>The attribute field to use when loading the user's first name. Example:</p> <ul style="list-style-type: none"> • <code>givenName</code>
User Last Name Attribute	<p>The attribute field to use when loading the user's last name. Example:</p> <ul style="list-style-type: none"> • <code>sn</code>
User Display Name Attribute	<p>The attribute field to use when loading the user's full name. Example:</p> <ul style="list-style-type: none"> • <code>displayName</code>
User Email Attribute	<p>The attribute field to use when loading the user's email address. Example:</p> <ul style="list-style-type: none"> • <code>mail</code>
User Password Attribute	<p>The attribute field to use when loading a user's password. Example:</p> <ul style="list-style-type: none"> • <code>unicodePwd</code>
User Unique ID Attribute	<p>The attribute used as a unique immutable identifier for user objects. This is used to track username changes and is optional. If this attribute is not set (or is set to an invalid value), user renames will not be detected — they will be interpreted as a user deletion then a new user addition.</p> <p>This should normally point to a UUID value. Standards-compliant LDAP servers will implement this as 'entryUUID' according to RFC 4530. This setting exists because it is known under different names on some servers, e.g. 'objectGUID' in Microsoft Active Directory.</p>

Group schema settings

Setting	Description
Group Object Class	<p>This is the name of the class used for the LDAP group object. Examples:</p> <ul style="list-style-type: none"> • <code>groupOfUniqueNames</code> • <code>group</code>

Group Object Filter	The filter to use when searching group objects. Example: <ul style="list-style-type: none"> • <code>(&(objectClass=group)(cn=*))</code>
Group Name Attribute	The attribute field to use when loading the group's name. Example: <ul style="list-style-type: none"> • <code>cn</code>
Group Description Attribute	The attribute field to use when loading the group's description. Example: <ul style="list-style-type: none"> • <code>description</code>

Membership schema settings

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: <ul style="list-style-type: none"> • <code>member</code>
User Membership Attribute	The attribute field to use when loading the user's groups. Example: <ul style="list-style-type: none"> • <code>memberOf</code>
Use the User Membership Attribute, when finding the user's group membership	Check this if your directory server supports the group membership attribute on the user. (By default, this is the 'memberOf' attribute.) <ul style="list-style-type: none"> • If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the list of groups to which a given user belongs. This will result in a more efficient retrieval. • If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search. • If the Enable Nested Groups checkbox is selected, your application will ignore the Use the User Membership Attribute option and will use the members attribute on the group for the search.
Use the User Membership Attribute, when finding the members of a group	Check this if your directory server supports the user membership attribute on the group. (By default, this is the 'member' attribute.) <ul style="list-style-type: none"> • If this checkbox is selected, your application will use the group membership attribute on the user when retrieving the members of a given group. This will result in a more efficient search. • If this checkbox is not selected, your application will use the members attribute on the group ('member' by default) for the search.

Diagrams of some possible configurations

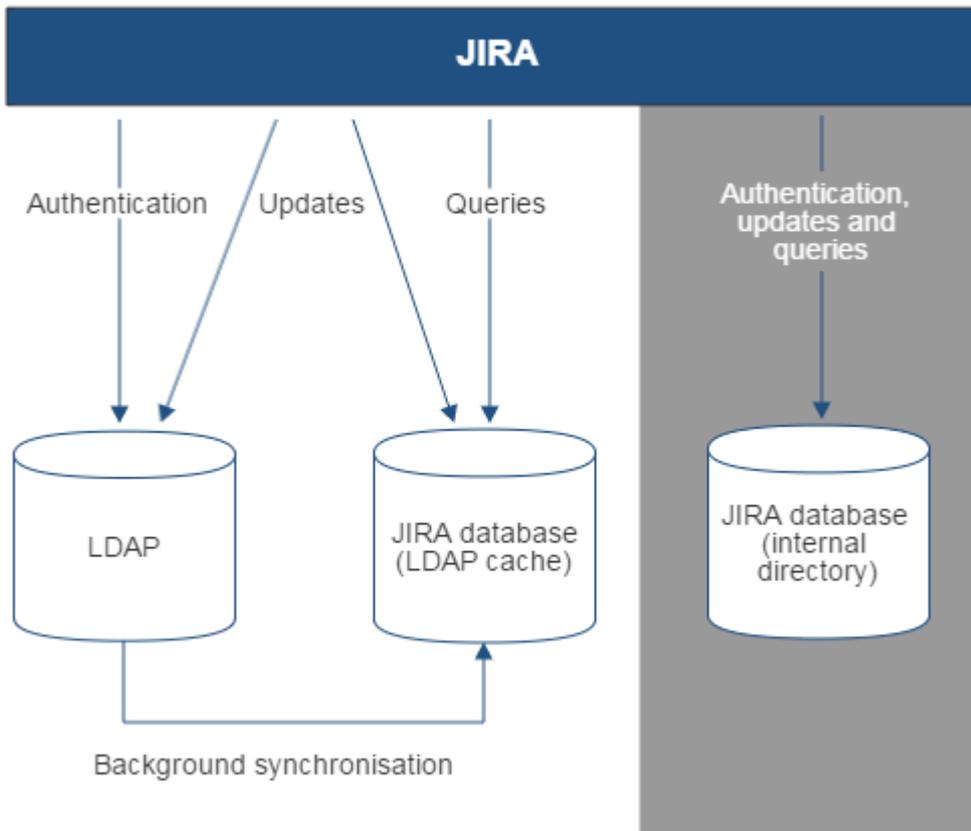


Diagram above: JIRA connecting to an LDAP directory.

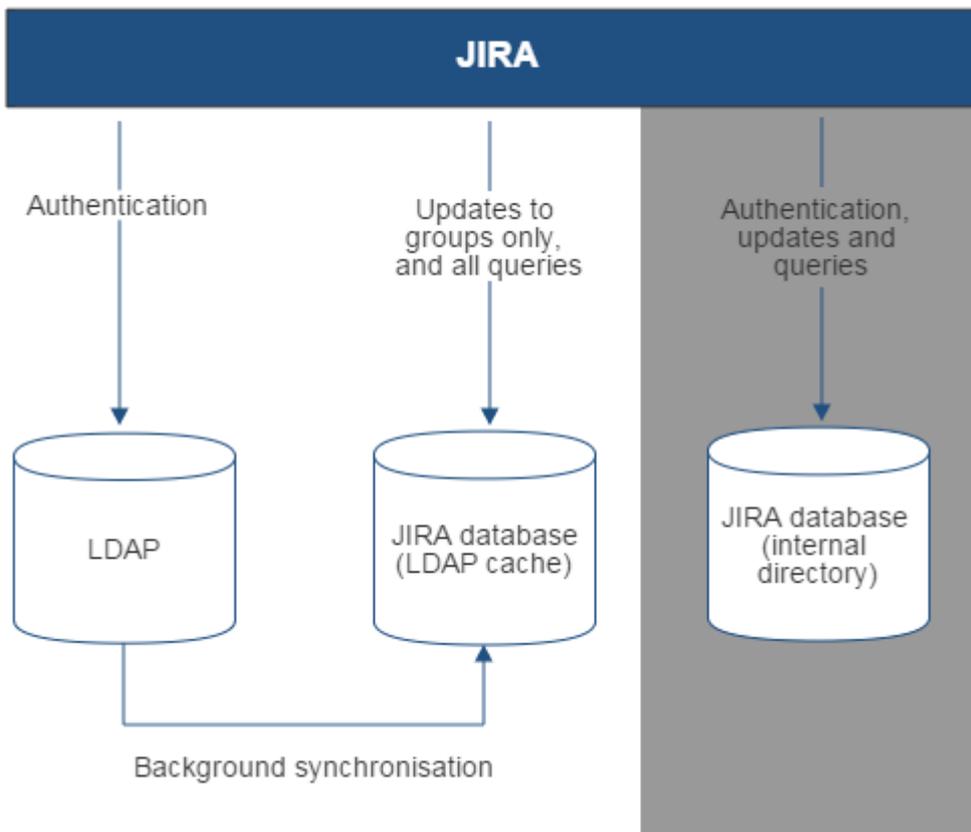


Diagram above: JIRA connecting to an LDAP directory with permissions set to read only and local groups.

Related topics

[Configuring user directories](#)

- Configuring the internal directory
- Connecting to an LDAP directory
 - Configuring an SSL connection to Active Directory
 - Reducing the number of users synchronized from LDAP to JIRA applications
- Connecting to an internal directory with LDAP authentication
- Connecting to Crowd or another JIRA application for user management
- Managing multiple directories
- Migrating users between user directories
- Synchronizing data from external directories

Configuring an SSL connection to Active Directory

Atlassian applications allow the use of SSL within our applications, however Atlassian Support does not provide assistance for configuring it. Consequently, Atlassian **can not guarantee providing any support for it**.

- If assistance with conversions of certificates is required, please consult with the vendor who provided the certificate.
- If assistance with configuration is required, please raise a question on [Atlassian Answers](#).

If you want to configure a read/write connection with Microsoft Active Directory, you will need to install an SSL certificate, generated by your Active Directory server, onto your JIRA server and then install the certificate into your JVM keystore.

On this page:

- Prerequisites
- Step 1. Install the Active Directory Certificate Services
- Step 2. Obtain the Server Certificate
- Step 3. Import the Server Certificate

There's a [Confluence SSL plugin](#) that facilitates this process.

Updating user, group, and membership details in Active Directory requires that your Atlassian application be running in a JVM that trusts the AD server. To do this, we generate a certificate on the Active Directory server, then import it into Java's `keystore`.

Prerequisites

To generate a certificate, you need the following components installed on the Windows Domain Controller to which you're connecting.

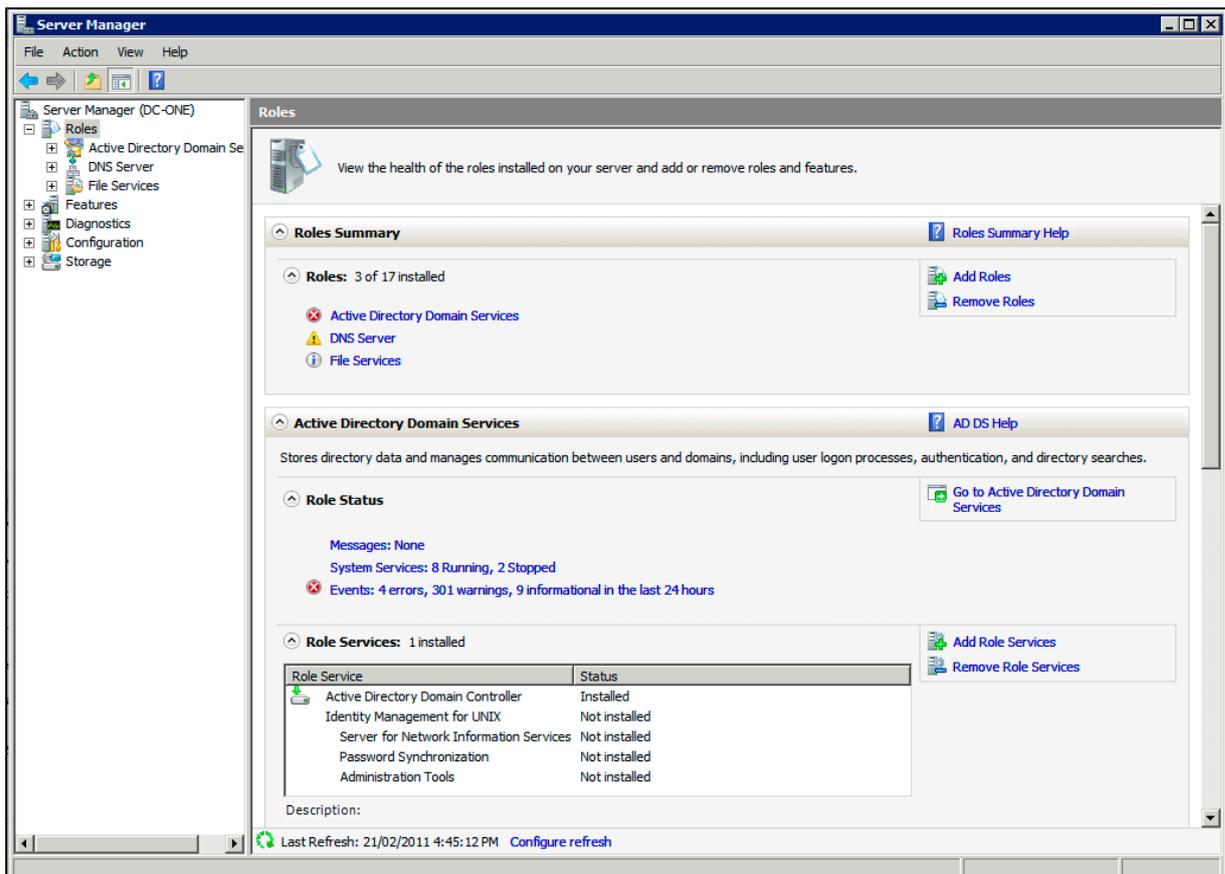
Required Component	Description
Internet Information Services (IIS)	This is required before you can install Windows Certificate Services.
Windows Certificate Services	This installs a certification authority (CA) which is used to issue certificates. Step 1, below, explains this process.

Windows 2000 Service Pack 2	Required if you are using Windows 2000
Windows 2000 High Encryption Pack (128-bit)	Required if you are using Windows 2000. Provides the highest available encryption level (128-bit).

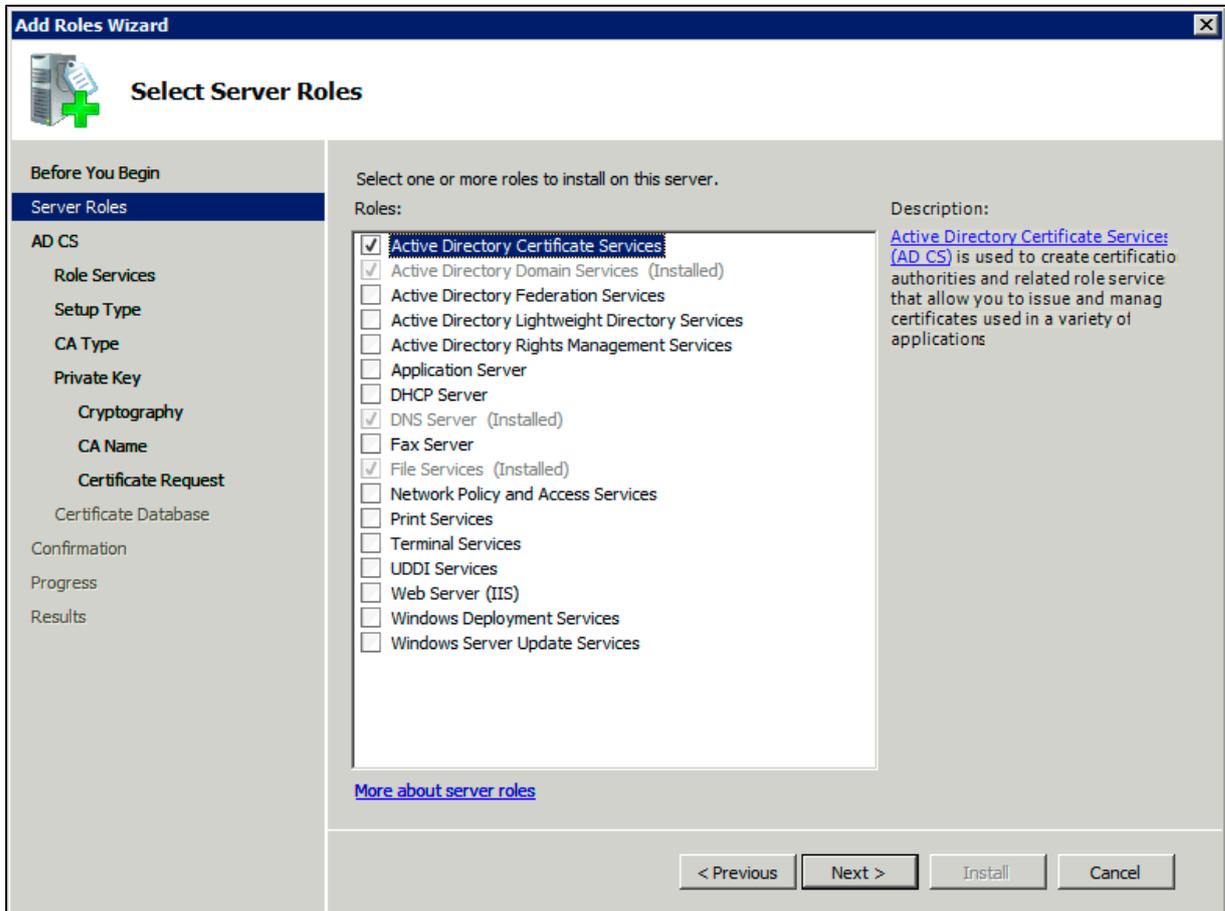
Step 1. Install the Active Directory Certificate Services

If Certificate Services are already installed, skip to step 2, below. The screenshots below are from Server 2008, but the process is similar for Server 2000 and 2003.

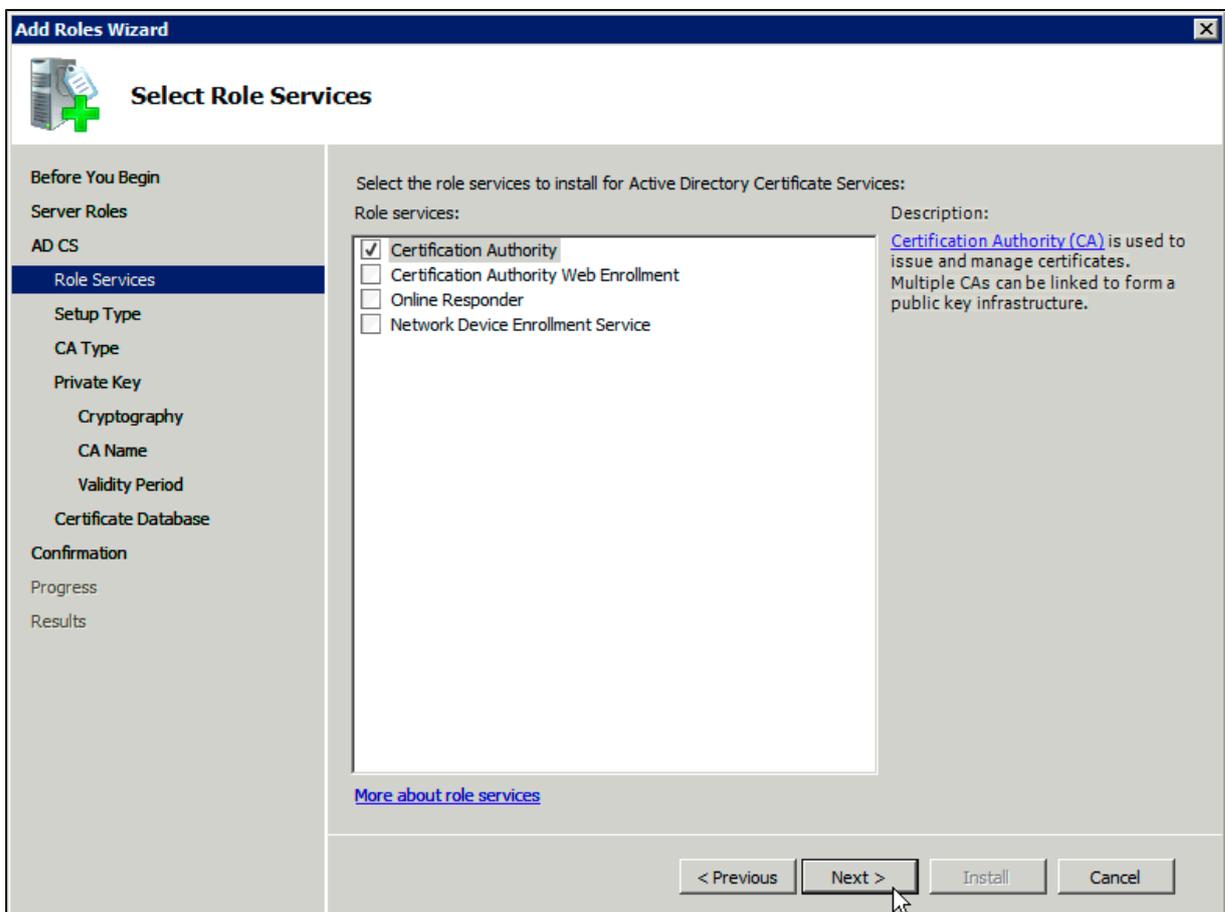
1. Log in to your Active Directory server as an administrator.
2. Click **Start**, point to **Administrative Tools**, and then click **Server Manager**.
3. In the **ROLES SUMMARY** section, click **Add Roles**.



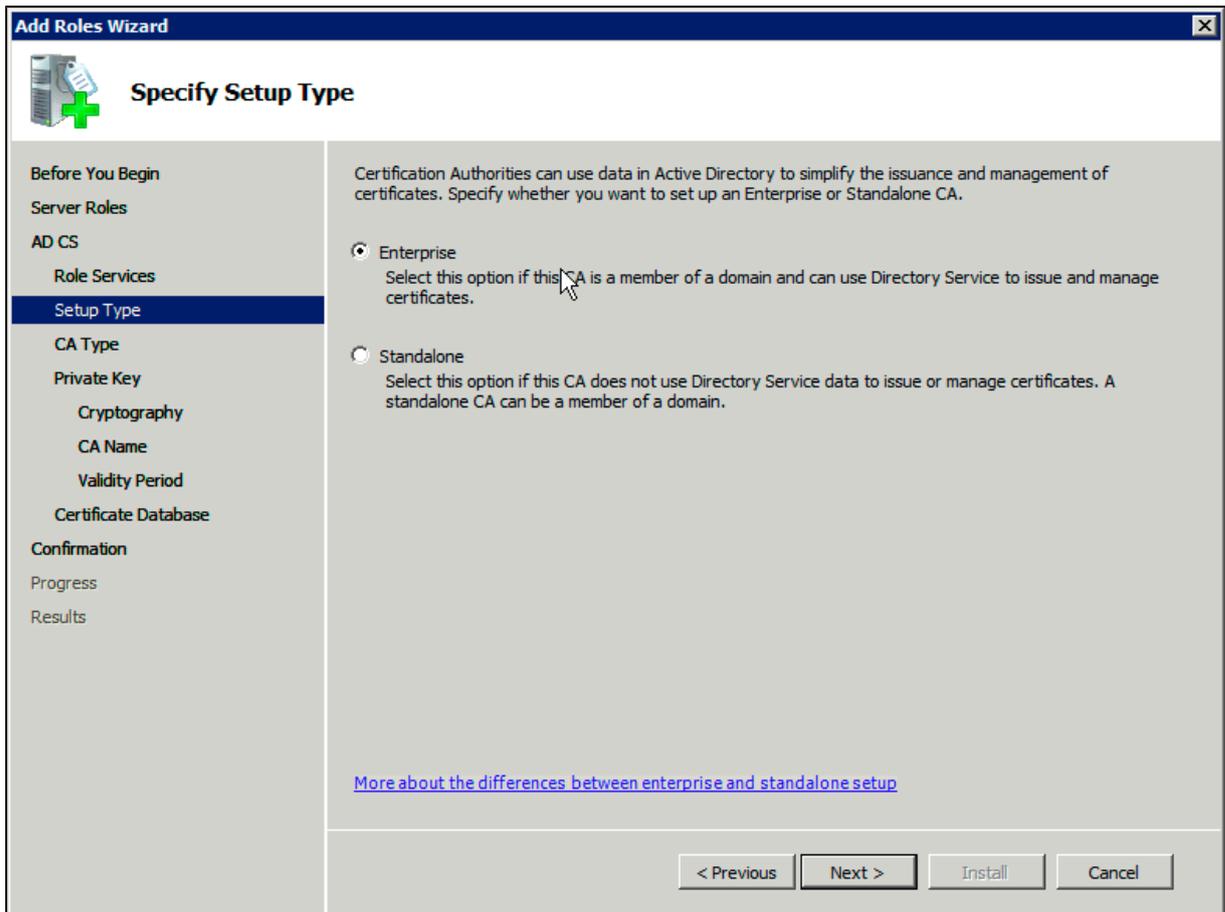
4. On the **Select Server Roles** page, select the **Active Directory Certificate Services** check box. Click **Next** twice.



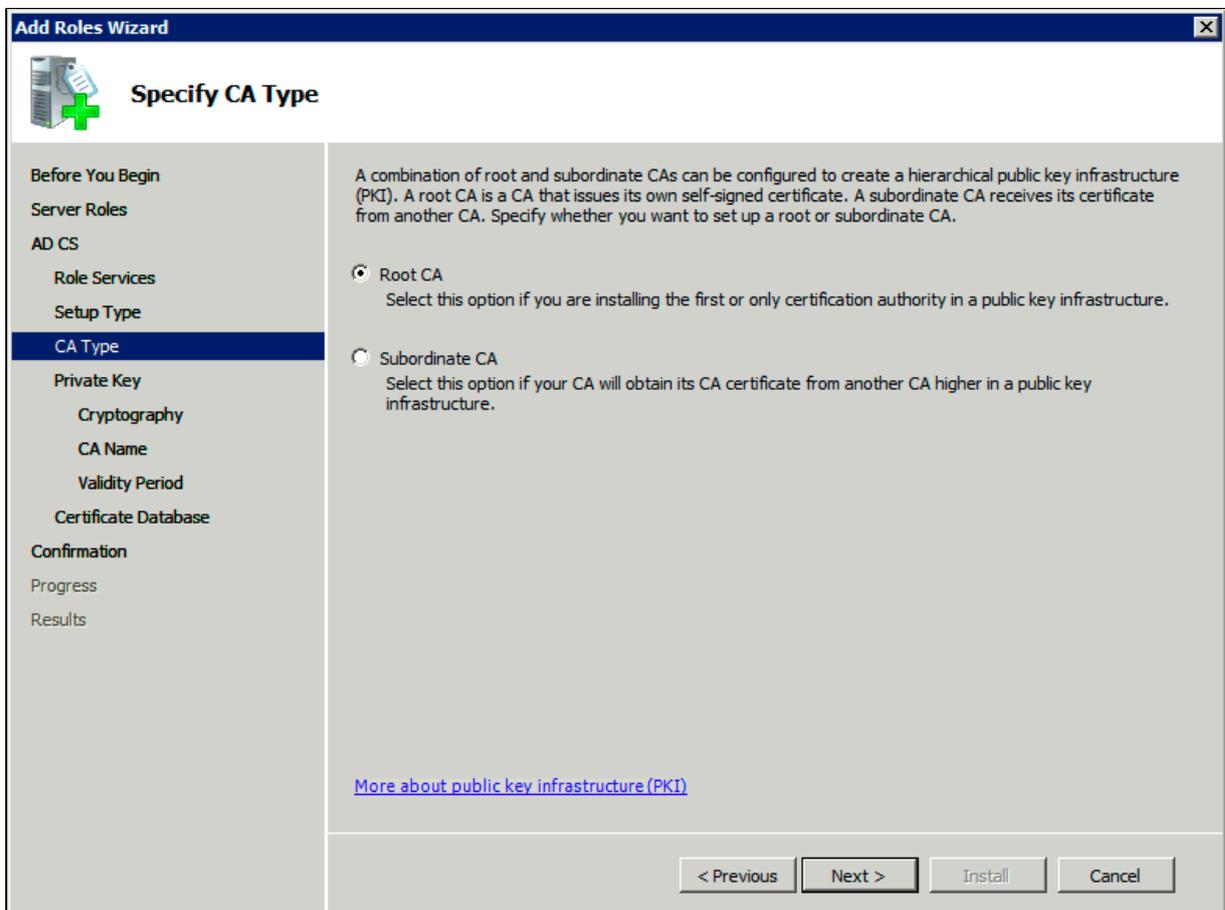
5. On the **Select Role Services** page, select the **Certification Authority** check box, and then click **Next**



6. On the **Specify Setup Type** page, click **Enterprise**, and then click **Next**.

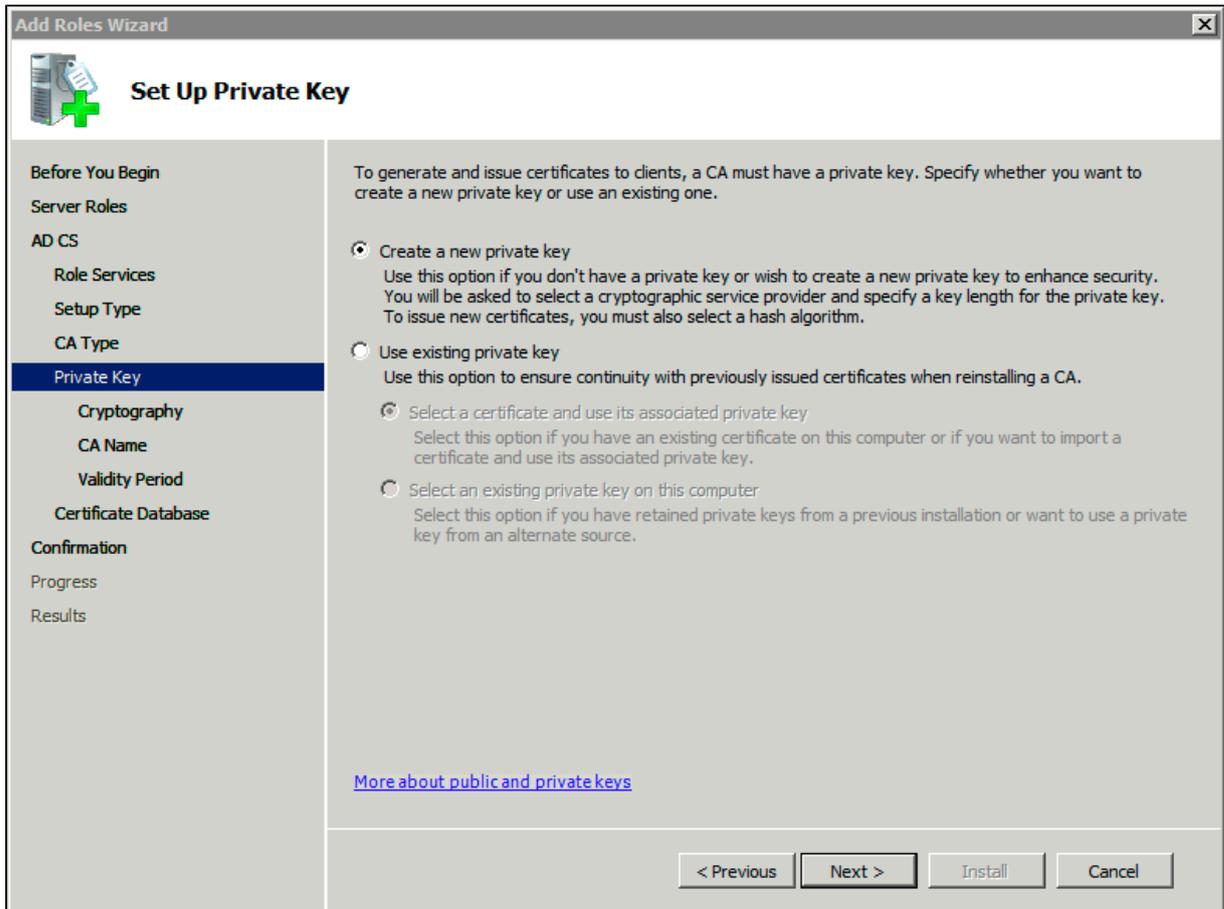


7. On the **Specify CA Type** page, click **Root CA**, and then click **Next**.



8. On the **Set Up Private Key** and **Configure Cryptography for CA** pages, you can configure optional

configuration settings, including cryptographic service providers. However, the default values should be fine. Click **Next** twice.



9. In the **Common name for this CA** box, type the common name of the CA, and then click **Next**.

Add Roles Wizard ✕

 **Configure CA Name**

Before You Begin

Server Roles

AD CS

Role Services

Setup Type

CA Type

Private Key

Cryptography

CA Name

Validity Period

Certificate Database

Confirmation

Progress

Results

Type in a common name to identify this CA. This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.

Common name for this CA:

Distinguished name suffix:

Preview of distinguished name:

[More about configuring a CA name](#)

10. On the **Set Validity Period** page, accept the default values or specify other storage locations for the certificate database and the certificate database log, and then click **Next**.

Add Roles Wizard ✕

 **Set Validity Period**

Before You Begin

Server Roles

AD CS

Role Services

Setup Type

CA Type

Private Key

Cryptography

CA Name

Validity Period

Certificate Database

Confirmation

Progress

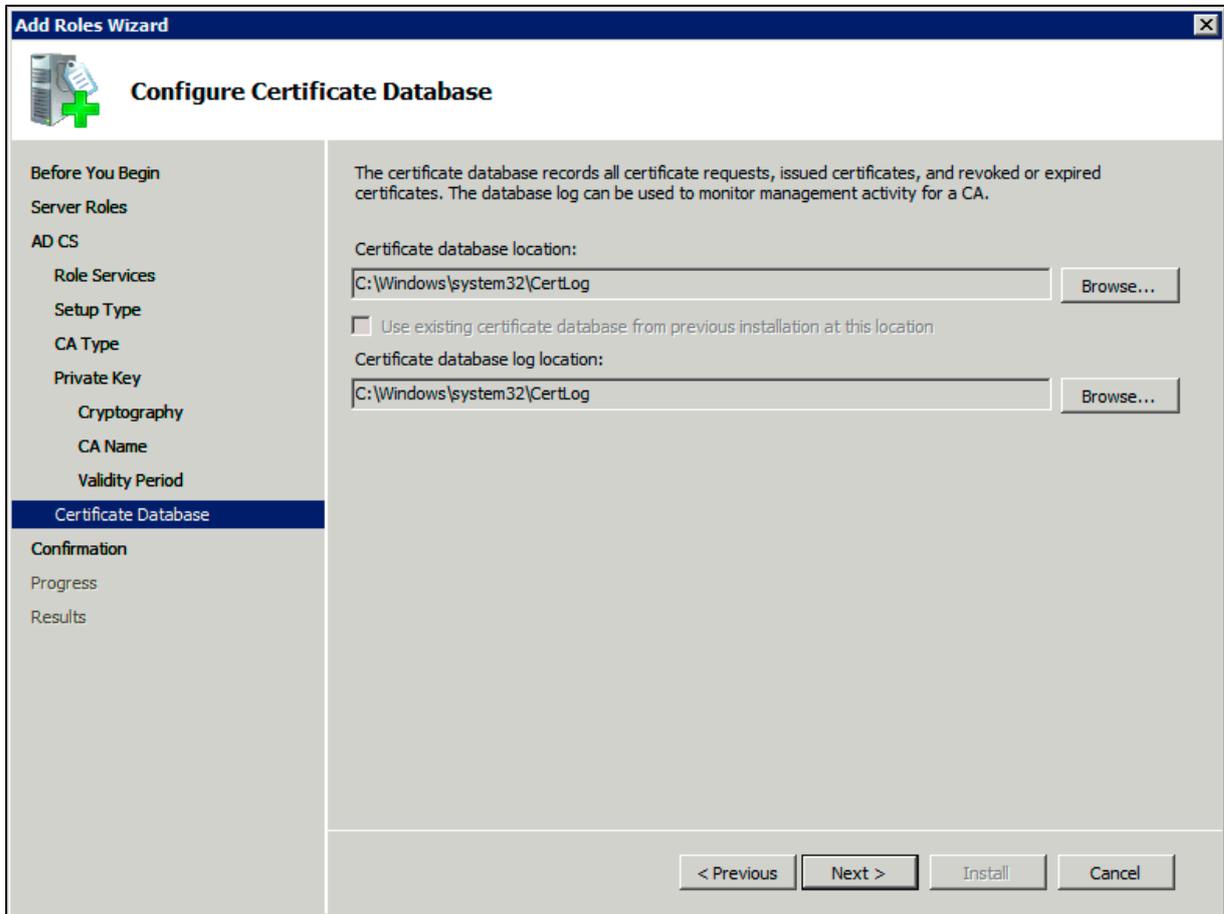
Results

A certificate will be issued to this CA to secure communications with other CAs and with clients requesting certificates. The validity period of a CA certificate can be based on a number of factors, including the intended purpose of the CA and security measures that you have taken to secure the CA.

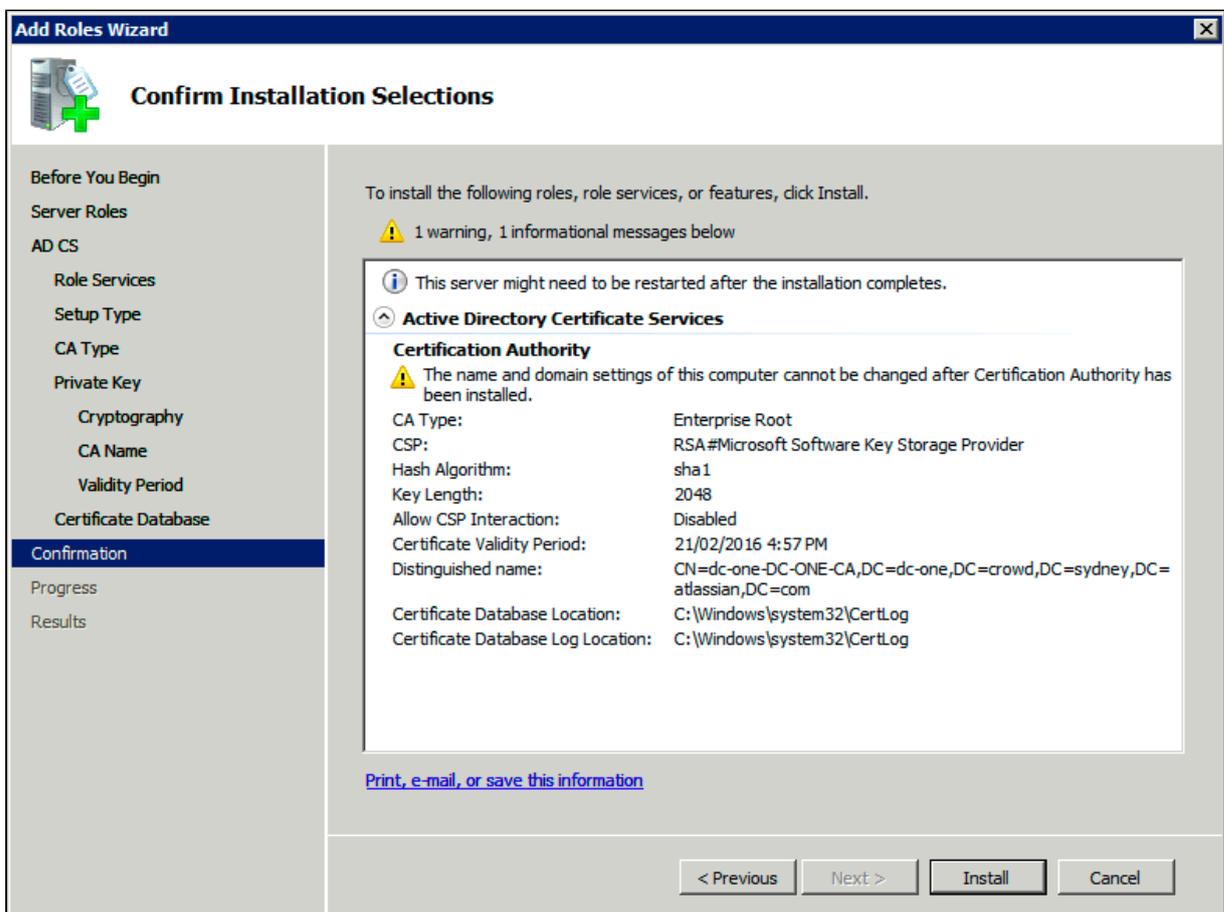
Select validity period for the certificate generated for this CA:

CA expiration Date: 21/02/2016 4:57 PM
 Note that CA will issue certificates valid only until its expiration date.

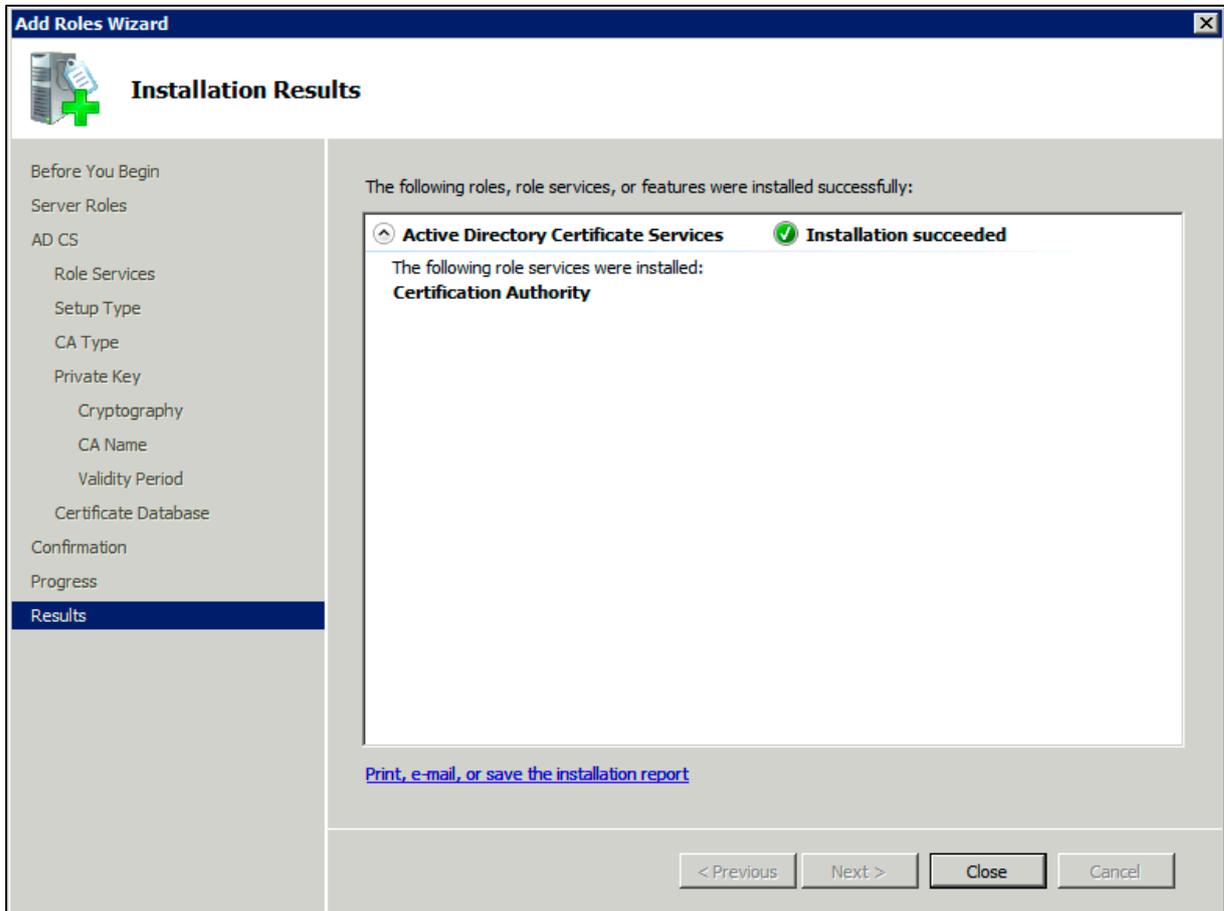
[More about setting the certificate validity period](#)



11. After verifying the information on the **Confirm Installation Selections** page, click **Install**.



12. Review the information on the results screen to verify that the installation was successful.



Step 2. Obtain the Server Certificate

The steps above describe how to install the certification authority (CA) on your Microsoft Active Directory server. Next, you will need to add the Microsoft Active Directory server's SSL certificate to the list of accepted certificates used by the JDK that runs your application server.

The Active Directory certificate is automatically generated and placed in root of the C:\ drive, matching a file format similar to the tree structure of your Active Directory server. For example: c:\ad2008.ad01.atlassian.com_ad01.crt.

You can also export the certificate by executing this command on the Active Directory server:

```
certutil -ca.cert client.crt
```

You might still fail to be authenticated using the certificate file above. In this case, Microsoft's [LDAP over SSL \(LDAPS\) Certificate](#) page might help. Note that you need to:

1. Choose "No, do not export the private key" in step-10 of [Exporting the LDAPS Certificate and Importing for use with AD DS](#) section
2. Choose "DER encoded binary X.509 (.CER)" in step-11 of [Exporting the LDAPS Certificate and Importing for use with AD DS](#) section. This file will be used in the following step.

Step 3. Import the Server Certificate

For an application server to trust your directory's certificate, the certificate must be imported into your Java runtime environment. The JDK stores trusted certificates in a file called a keystore. The default keystore file is called `cacerts` and it lives in the `jre\lib\security` sub-directory of your Java installation.

In the following examples, we use `server-certificate.crt` to represent the certificate file exported by your directory server. You will need to alter the instructions below to match the name actually generated.

Once the certificate has been imported as per the below instructions, you will need to restart the application

to pick up the changes.

Windows

1. Navigate to the directory in which Java is installed. It's probably called something like C:\Program Files\Java\jdk1.5.0_12.

```
cd /d C:\Program Files\Java\jdk1.5.0_12
```

2. Run the command below, where `server-certificate.crt` is the name of the file from your directory server:

```
keytool -importcert -keystore .\jre\lib\security\cacerts -file
server-certificate.crt
```

3. `keytool` will prompt you for a password. The default keystore password is `changeit`.
4. When prompted `Trust this certificate? [no]:` enter `yes` to confirm the key import:

```
Enter keystore password:  changeit
Owner: CN=ad01, C=US
Issuer: CN=ad01, C=US
Serial number: 15563d6677a4e9e4582d8a84be683f9
Valid from: Tue Aug 21 01:10:46 ACT 2007 until: Tue Aug 21
01:13:59 ACT 2012
Certificate fingerprints:
    MD5:  D6:56:F0:23:16:E3:62:2C:6F:8A:0A:37:30:A1:84:BE
    SHA1:
73:73:4E:A6:A0:D1:4E:F4:F3:CD:CE:BE:96:80:35:D2:B4:7C:79:C1
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

You may now change **'URL'** to use LDAP over SSL (i.e. `ldaps://<HOSTNAME>:636/`) and use the **'Secure SSL'** option when connecting your application to your directory server.

UNIX

1. Navigate to the directory in which the Java used by JIRA is installed. If the default JAVA installation is used, then it would be

```
cd $JAVA_HOME
```

2. Run the command below, where `server-certificate.crt` is the name of the file from your directory server:

```
sudo keytool -importcert -keystore ./jre/lib/security/cacerts
-file server-certificate.crt
```

3. `keytool` will prompt you for a password. The default keystore password is `changeit`.
4. When prompted `Trust this certificate? [no]:` enter `yes` to confirm the key import:

```

Password:
Enter keystore password:  changeit
Owner: CN=ad01, C=US
Issuer: CN=ad01, C=US
Serial number: 15563d6677a4e9e4582d8a84be683f9
Valid from: Tue Aug 21 01:10:46 ACT 2007 until: Tue Aug 21
01:13:59 ACT 2012
Certificate fingerprints:
    MD5:  D6:56:F0:23:16:E3:62:2C:6F:8A:0A:37:30:A1:84:BE
    SHA1:
73:73:4E:A6:A0:D1:4E:F4:F3:CD:CE:BE:96:80:35:D2:B4:7C:79:C1
Trust this certificate? [no]:  yes
Certificate was added to keystore

```

You may now change **'URL'** to use LDAP over SSL (i.e. `ldaps://<HOSTNAME>:636/`) and use the **'Secure SSL'** option when connecting your application to your directory server.

Mac OS X

1. Navigate to the directory in which Java is installed. This is usually

```
cd /Library/Java/Home
```

2. Run the command below, where `server-certificate.crt` is the name of the file from your directory server:

```
sudo keytool -importcert -keystore ./jre/lib/security/cacerts
-file server-certificate.crt
```

3. `keytool` will prompt you for a password. The default keystore password is `changeit`.
4. When prompted `Trust this certificate? [no]:` enter `yes` to confirm the key import:

```

Password:
Enter keystore password:  changeit
Owner: CN=ad01, C=US
Issuer: CN=ad01, C=US
Serial number: 15563d6677a4e9e4582d8a84be683f9
Valid from: Tue Aug 21 01:10:46 ACT 2007 until: Tue Aug 21
01:13:59 ACT 2012
Certificate fingerprints:
    MD5:  D6:56:F0:23:16:E3:62:2C:6F:8A:0A:37:30:A1:84:BE
    SHA1:
73:73:4E:A6:A0:D1:4E:F4:F3:CD:CE:BE:96:80:35:D2:B4:7C:79:C1
Trust this certificate? [no]:  yes
Certificate was added to keystore

```

You may now change **'URL'** to use LDAP over SSL (i.e. `ldaps://<HOSTNAME>:636/`) and use the **'Secure SSL'** option when connecting your application to your directory server.

Related topics

[Connecting to an LDAP directory](#)
[Configuring user directories](#)

Reducing the number of users synchronized from LDAP to JIRA applications

If you have connected JIRA applications to an LDAP directory for authentication, user and group management, you may want configure your applications to synchronize a subset of users from LDAP rather than all users. There are two reasons for why you might make this change:

- Improving performance — If you have performance issues during synchronization process, you may be able to improve this by synchronizing a subset of data instead. See this knowledge base article for more information: [Performance Issues with Large LDAP Repository - 100,000 users or more](#).
- Reducing your user count — You can synchronize a subset of users to JIRA applications from LDAP to reduce your user count. This will allow you to count less users against your JIRA application licenses. However, synchronizing a subset of users to JIRA applications from LDAP is not the recommended method for reducing your JIRA application user count. We recommend that you reduce the JIRA application user count by deactivating the users within JIRA. Check [this page](#) for more info on removing users from JIRA.

Procedure

The procedure for configuring JIRA applications to synchronize a different number of users from LDAP depends on how you initially set up your LDAP directory. For example, if you have all your JIRA application users in one organizational unit and your non-JIRA application users in another organizational unit, then you can simply configure JIRA applications to only synchronize users against a particular DN (distinguished name). However, if your setup is not so simple (e.g. you have your JIRA application users and non-JIRA application users in the same node), you will need to define an LDAP filter to synchronize the relevant users. Both of these methods are outlined below.

Synchronizing against Base DN, Additional User DN and Additional Group DN

1. Log in as a user with the [JIRA Administrators global permission](#).
2. Select **Administration > Users > User Directories**.
3. Update the **Base DN** field, and optionally the **Additional User DN** and/or **Additional Group DN** to query against the directory server as desired.
4. For example, if you have configured all of your JIRA application users in the jira-users organizational unit only, for your company at mycompany.example.com, your configuration would look like this:
 - **Base DN** — `dc=mycompany,dc=example,dc=com`
 - **Additional Group DN** — `ou=jira-users`

Defining an LDAP filter

1. Log in as a user with the [JIRA Administrators global permission](#).
Select **Administration > Users > User Directories**.
2. Update **User Object Filter** and/or **Group Object Filter** fields as desired. The syntax for LDAP filters is not simple and your query will depend on how you have set up your LDAP directory.
3. For example, if you have configured only JIRA application groups to have 'jira' in the CN, you can use a wildcard search in your filter to find them by setting the **Group Object Filter** = `(objectCategory=group)(cn=*jira*)`
More information on defining LDAP filters is available in the pages linked in the *Related Topics* section below.

Related topics:

[Performance Issues with Large LDAP Repository - 100,000 users or more](#)

[Unable to create issues due to exceeded number of licenses](#)

[How to write LDAP search filters](#)

[MSDN guide to LDAP search filter syntax](#)

Connecting to an internal directory with LDAP authentication

You can connect your JIRA application to an LDAP directory for delegated authentication. This means that JIRA will have an internal directory that uses

LDAP for authentication only. There is an option to create users in the internal directory automatically when they attempt to log in, as described in the settings section.

If you decide to use an LDAP directory for delegated authentication, you're unable to use [nested groups](#).

You will need to log in as a user with the 'JIRA System Administrators' [global permission](#) to access the Settings menu below.

On this page:

- [Overview](#)
- [Connecting JIRA to an internal directory with LDAP authentication](#)
- [Server settings](#)
 - [Copying users on first login](#)
- [Schema settings](#)
- [User schema settings \(used when copying users on first login\)](#)
- [Group schema settings \(used when enabling 'synchronize group memberships'\)](#)
- [Diagrams of possible configurations](#)

Overview

An internal directory with LDAP authentication offers the features of an internal directory while allowing you to store and check users' passwords in LDAP only. Note that the 'internal directory with LDAP authentication' is separate from the default 'internal directory'. On LDAP, all that the application does is to check the password. The LDAP connection is read only. Every user in the internal directory with LDAP authentication must map to a user on LDAP, otherwise they cannot log in.

When to use this option: Choose this option if you want to set up a user and group configuration within your application that suits your needs, while checking your users' passwords against the corporate LDAP directory. This option also helps to avoid the performance issues that may result from downloading large numbers of groups from LDAP.

Connecting JIRA to an internal directory with LDAP authentication

To connect to an internal directory but check logins via LDAP:

1. Choose



> User Management.

2. Choose **User Directories**.
3. **Add** a directory and select type '**Internal with LDAP Authentication**'.
4. Enter the values for the settings, as described below.
5. Save the directory settings.
6. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen. We recommend that the 'Internal Directory with LDAP Authentication' is at the top of the list. Here is a summary of how the directory order affects the processing:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.
 For details, see [Managing multiple directories](#).
7. Add your users and groups in JIRA. See [Managing users](#) and [Managing groups](#).

Server settings

Setting	Description
Name	A descriptive name that will help you to identify the directory. Examples: <ul style="list-style-type: none"> • Internal directory with LDAP Authentication • Corporate LDAP for Authentication Only
Directory Type	Select the type of LDAP directory that you will connect to. If you are adding a new LDAP connection, the value you select here will determine the default values for some of the options on the rest of screen. Examples: <ul style="list-style-type: none"> • Microsoft Active Directory • OpenDS • And more.
Hostname	The host name of your directory server. Examples: <ul style="list-style-type: none"> • ad.example.com • ldap.example.com • opensds.example.com
Port	The port on which your directory server is listening. Examples: <ul style="list-style-type: none"> • 389 • 10389 • 636 (for example, for SSL)
Use SSL	Check this box if the connection to the directory server is an SSL (Secure Sockets Layer) connection. Note that you will need to configure an SSL certificate in order to use this setting.
Username	The distinguished name of the user that the application will use when connecting to the directory server. Examples: <ul style="list-style-type: none"> • cn=admin, cn=users, dc=ad, dc=example, dc=com • cn=user, dc=domain, dc=name • user@domain.name
Password	The password of the user specified above.

Copying users on first login

Setting	Description
---------	-------------

Copy User on Login	<p>This option affects what will happen when a user attempts to log in. If this box is checked, the user will be created automatically in the internal directory that is using LDAP for authentication when the user first logs in and their details will be synchronized on each subsequent log in. If this box is not checked, the user's login will fail if the user wasn't already manually created in the directory.</p> <p>If you check this box the following additional fields will appear on the screen, which are described in more detail below:</p> <ul style="list-style-type: none"> • Default Group Memberships • Synchronize Group Memberships • User Schema Settings (described in a separate section below)
Update User attributes on Login	<p>Whenever your users authenticate to the application, their attributes will be automatically updated from the LDAP server into the application. After you select this option, you won't be able to modify or delete your users directly in the application.</p> <ul style="list-style-type: none"> • If you need to modify a user, do it on the LDAP server; it will be updated in the application after authenticating. • If you need to delete a user, do it on the LDAP server, but also in the application. If you delete the user only on the LDAP server, it will be rejected from logging in to the application, but it won't be set as inactive, which will affect your license. You'll need to disable the Update User attributes on Login option to delete the user, and then enable it again.
Default Group Memberships	<p>This field appears if you check the Copy User on Login box. If you would like users to be automatically added to a group or groups, enter the group name(s) here. To specify more than one group, separate the group names with commas. Each time a user logs in, their group memberships will be checked. If the user does not belong to the specified group(s), their username will be added to the group(s). If a group does not yet exist, it will be added to the internal directory that is using LDAP for authentication.</p> <p>Please note that there is no validation of the group names. If you mis-type the group name, authorization failures will result – users will not be able to access the applications or functionality based on the intended group name.</p> <p>Examples:</p> <ul style="list-style-type: none"> • confluence-users • bamboo-users , jira-administrators , jira-core-users
Synchronize Group Memberships	<p>This field appears if you select the Copy User on Login checkbox. If this box is checked, group memberships specified on your LDAP server will be synchronized with the internal directory each time the user logs in.</p> <p>If you check this box the following additional fields will appear on the screen, both described in more detail below:</p> <ul style="list-style-type: none"> • Group Schema Settings (described in a separate section below) • Membership Schema Settings (described in a separate section below)

Schema settings

Setting	Description
---------	-------------

Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: <ul style="list-style-type: none"> • <code>o=example,c=com</code> • <code>cn=users,dc=ad,dc=example,dc=com</code> • For Microsoft Active Directory, specify the base DN in the following format: <code>dc=domain1,dc=local</code>. You will need to replace the <code>domain1</code> and <code>local</code> for your specific configuration. Microsoft Server provides a tool called <code>ldp.exe</code> which is useful for finding out and configuring the the LDAP structure of your server.
User Name Attribute	The attribute field to use when loading the username. Examples: <ul style="list-style-type: none"> • <code>cn</code> • <code>sAMAccountName</code>

User schema settings (used when copying users on first login)

Setting	Description
Additional User DN	This value is used in addition to the base DN when searching and loading users. If no value is supplied, the subtree search will start from the base DN. Example: <ul style="list-style-type: none"> • <code>ou=Users</code>
User Object Class	This is the name of the class used for the LDAP user object. Example: <ul style="list-style-type: none"> • <code>user</code>
User Object Filter	The filter to use when searching user objects. Example: <ul style="list-style-type: none"> • <code>(&(objectCategory=Person)(sAMAccountName=*))</code>
User Name RDN Attribute	The RDN (relative distinguished name) to use when loading the username. The DN for each LDAP entry is composed of two parts: the RDN and the location within the LDAP directory where the record resides. The RDN is the portion of your DN that is not related to the directory tree structure. Example: <ul style="list-style-type: none"> • <code>cn</code>
User First Name Attribute	The attribute field to use when loading the user's first name. Example: <ul style="list-style-type: none"> • <code>givenName</code>
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: <ul style="list-style-type: none"> • <code>sn</code>
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: <ul style="list-style-type: none"> • <code>displayName</code>
User Email Attribute	The attribute field to use when loading the user's email address. Example: <ul style="list-style-type: none"> • <code>mail</code>

Group schema settings (used when enabling 'synchronize group memberships')

Setting	Description
---------	-------------

Group Object Class	This is the name of the class used for the LDAP group object. Examples: <ul style="list-style-type: none"> • <code>groupOfUniqueNames</code> • <code>group</code>
Group Object Filter	The filter to use when searching group objects. Example: <ul style="list-style-type: none"> • <code>(&(objectClass=group)(cn=*))</code>
Group Name Attribute	The attribute field to use when loading the group's name. Example: <ul style="list-style-type: none"> • <code>cn</code>
Group Description Attribute	The attribute field to use when loading the group's description. Example: <ul style="list-style-type: none"> • <code>description</code>

Diagrams of possible configurations

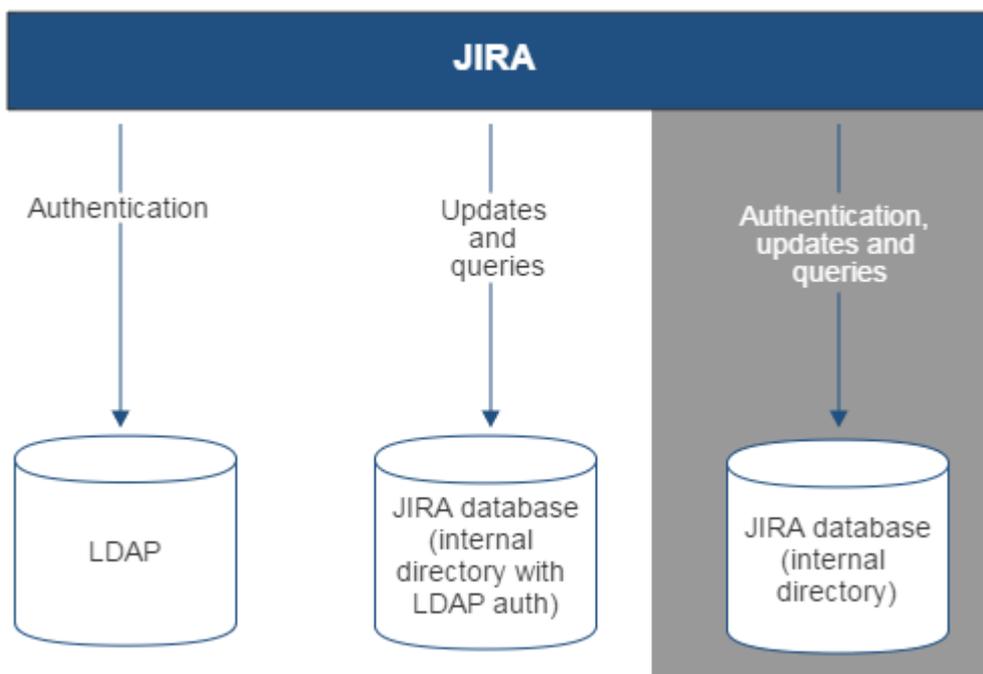


Diagram above: JIRA connecting to an LDAP directory for authentication only.

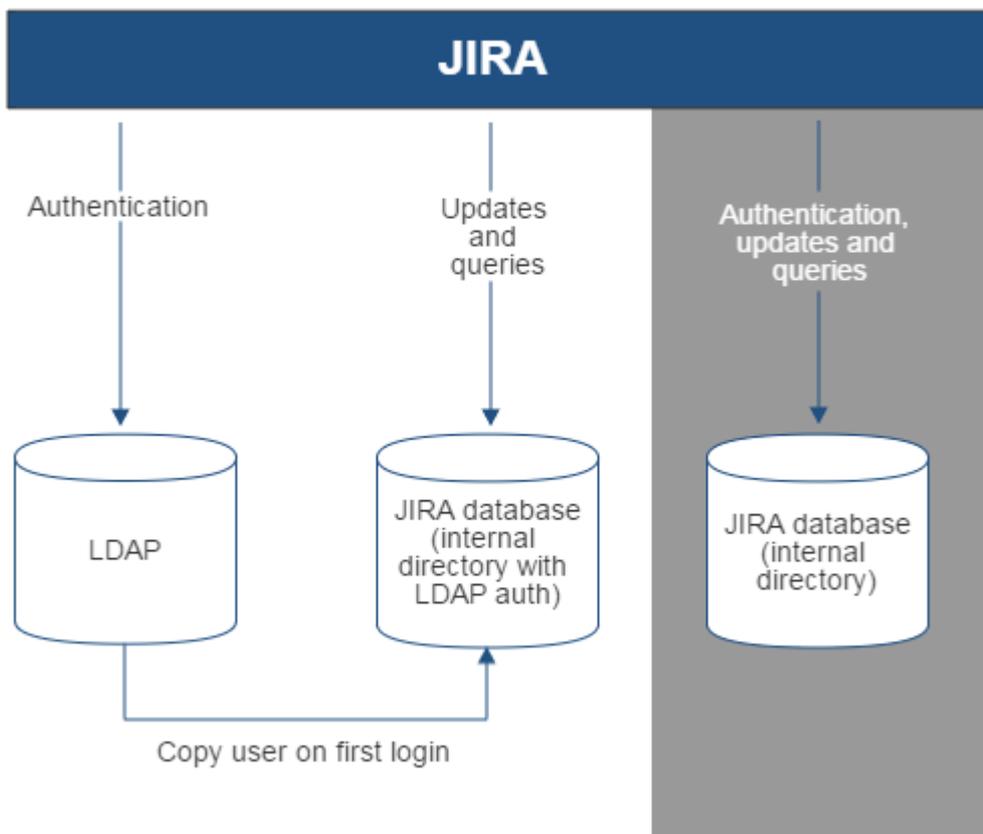


Diagram above: JIRA connecting to an LDAP directory for authentication only, with each user copied to the internal directory when they first log in to JIRA.

Related topics

Configuring user directories

- [Configuring the internal directory](#)
- [Connecting to an LDAP directory](#)
- [Connecting to an internal directory with LDAP authentication](#)
- [Connecting to Crowd or another JIRA application for user management](#)
- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)

Connecting to Crowd or another JIRA application for user management

You can connect your JIRA application to Atlassian Crowd or to another JIRA server application (version 4.3 or later) for management of users and groups, and for authentication (verification of a user's login). You will need to log in as a user with the 'JIRA System Administrators' global permission to access the Settings menu below.

On this page:

- [Connecting a JIRA application to Crowd](#)
- [Connecting JIRA applications to another server](#)
- [Diagrams of some possible configurations](#)

Connecting a JIRA application to Crowd

Atlassian Crowd is an application security framework that handles authentication and authorization for your web-based applications. With Crowd you can integrate multiple web applications and user directories, with support for single sign-on (SSO) and centralized identity management. The Crowd Administration Console provides a web interface for managing directories, users and their permissions. See the [Administration Guide](#).

When to use this option: Connect to Crowd if you want to use the full Crowd functionality to manage your directories, users and groups. You can connect your Crowd server to a number of directories of all types that Crowd supports, including custom directory connectors.

To connect a JIRA application to Crowd:

1. Go to your **Crowd Administration Console** and define the JIRA application to Crowd. See the Crowd documentation: [Adding an Application](#).
2. Choose  **> User Management.**
3. Choose **User Directories.**
4. **Add** a directory and select type '**Atlassian Crowd**'. Enter the settings as described below.
5. Save the directory settings.
6. Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen. Here is a summary of how the directory order affects the processing:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.
 For details, see [Managing multiple directories](#).
7. If required, configure JIRA to use Crowd for single sign-on (SSO) too. See the Crowd documentation: [Integrating Crowd with Atlassian Jira](#).

Notes:

- If you have JIRA-Crowd-LDAP, every time user logs in (i.e. first and subsequent times), the user's data in JIRA/Crowd will be updated from the user's data in LDAP. This includes username, display name, email and group memberships. However for group memberships, only the following applies:
 - direct groups only (i.e. not nested groups) are synchronized from LDAP.
 - only groups that are already present in the JIRA application are synchronized, i.e. groups are not added/removed, and group hierarchies are not synchronized.

Settings in JIRA applications for the Crowd directory type

Setting	Description
Name	A meaningful name that will help you to identify this Crowd server amongst your list of directory servers. Examples: <ul style="list-style-type: none"> • Crowd Server • Example Company Crowd
Server URL	The web address of your Crowd console server. Examples: <ul style="list-style-type: none"> • http://www.example.com:8095/crowd/ • http://crowd.example.com
Application Name	The name of your application, as recognized by your Crowd server. Note that you will need to define the application in Crowd too, using the Crowd administration Console. See the Crowd documentation on adding an application .
Application Password	The password which the application will use when it authenticates against the Crowd framework as a client. This must be the same as the password you have registered in Crowd for this application. See the Crowd documentation on adding an application .

Crowd permissions

Setting	Description
Read Only	The users, groups and memberships in this directory are retrieved from Crowd and can only be modified via Crowd. You cannot modify Crowd users, groups or memberships via the application administration screens.
Read/Write	The users, groups and memberships in this directory are retrieved from Crowd. When you modify a user, group or membership via the application administration screens, the changes will be applied directly to Crowd. Please ensure that the application has modification permissions for the relevant directories in Crowd. See the Crowd documentation: Specifying an Application's Directory Permissions .

Advanced Crowd settings

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if the user directory or directories in Crowd support nested groups. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.
Enable Incremental Synchronization	Enable or disable incremental synchronization. Only changes since the last synchronization will be retrieved when synchronizing a directory. Note that full synchronization is always executed when restarting Fisheye.
Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.

Connecting JIRA applications to another server

Subject to certain limitations, you can connect a number of Atlassian applications to a single JIRA application for centralized user management.

When to use this option: You can connect to a server running **JIRA 4.3** or later, **JIRA Software 7.0** or later, **JIRA Core 7.0** or later, or **JIRA Service Desk 3.0** or later. Choose this option as an alternative to Atlassian Crowd, for simple configurations with a limited number of users.

Let's assume that you have two JIRA application servers, called for example '**JIRA instance 1**' and '**JIRA instance 2**'. You want JIRA instance 2 to manage your users and groups. JIRA instance 1 will delegate user management to JIRA instance 2.

To connect JIRA instance 1 to use JIRA instance 2 for user management:

- Configure JIRA instance 2 to recognize JIRA instance 1:
 - Choose  **> User Management.**
 - Choose **User Directories.**
 - Add** an application.
 - Enter the **application name** and **password** that JIRA instance 1 will use when accessing JIRA instance 2.
 - Enter the **IP address** or addresses of JIRA instance 1. Valid values are:
 - A full IP address, e.g. 192.168.10.12.
 - A wildcard IP range, using CIDR notation, e.g. 192.168.10.1/16. For more information, see the introduction to [CIDR notation on Wikipedia](#) and [RFC 4632](#).
 - Save** the new application.
- Configure JIRA instance 1 to delegate user management:
 - Choose



> User Management.

- Choose **User Directories**.
- **Add** a directory and select type '**Atlassian JIRA**'.
- Enter the settings as described below. When asked for the **application name** and **password**, enter the values that you defined in the settings on JIRA instance 2.
- Save the directory settings.
- Define the **directory order** by clicking the blue up- and down-arrows next to each directory on the '**User Directories**' screen. Here is a summary of how the directory order affects the processing:
 - The order of the directories is the order in which they will be searched for users and groups.
 - Changes to users and groups will be made only in the first directory where the application has permission to make changes.

For details, see [Managing multiple directories](#).

Settings for the JIRA application directory type

Setting	Description
Name	A meaningful name that will help you to identify this Jira server in the list of directory servers. Examples: <ul style="list-style-type: none"> • Jira Service Desk Server • My Company Jira
Server URL	The web address of your Jira server. Examples: <ul style="list-style-type: none"> • http://www.example.com:8080 • http://jira.example.com
Application Name	The name used by your application when accessing the Jira server that acts as user manager. Note that you will also need to define your application to that Jira server, via the ' Other Applications ' option in the 'Users, Groups & Roles' section of the 'Administration' menu.
Application Password	The password used by your application when accessing the Jira server that acts as user manager.

Permissions for the JIRA application directory type

Setting	Description
Read Only	The users, groups and memberships in this directory are retrieved from the Jira server that is acting as user manager. They can only be modified via that JIRA server.
Read/Write	The users, groups and memberships in this directory are retrieved from the Jira server that is acting as user manager. When you modify a user, group or membership via the application administration screens, the changes will be applied directly to Jira.

Advanced Settings for the JIRA application directory type

Setting	Description
Enable Nested Groups	Enable or disable support for nested groups. Before enabling nested groups, please check to see if nested groups are enabled on the JIRA server that is acting as user manager. When nested groups are enabled, you can define a group as a member of another group. If you are using groups to manage permissions, you can create nested groups to allow inheritance of permissions from one group to its sub-groups.

Synchronization Interval (minutes)	Synchronization is the process by which the application updates its internal store of user data to agree with the data on the directory server. The application will send a request to your directory server every x minutes, where 'x' is the number specified here. The default value is 60 minutes.
------------------------------------	--

Diagrams of some possible configurations

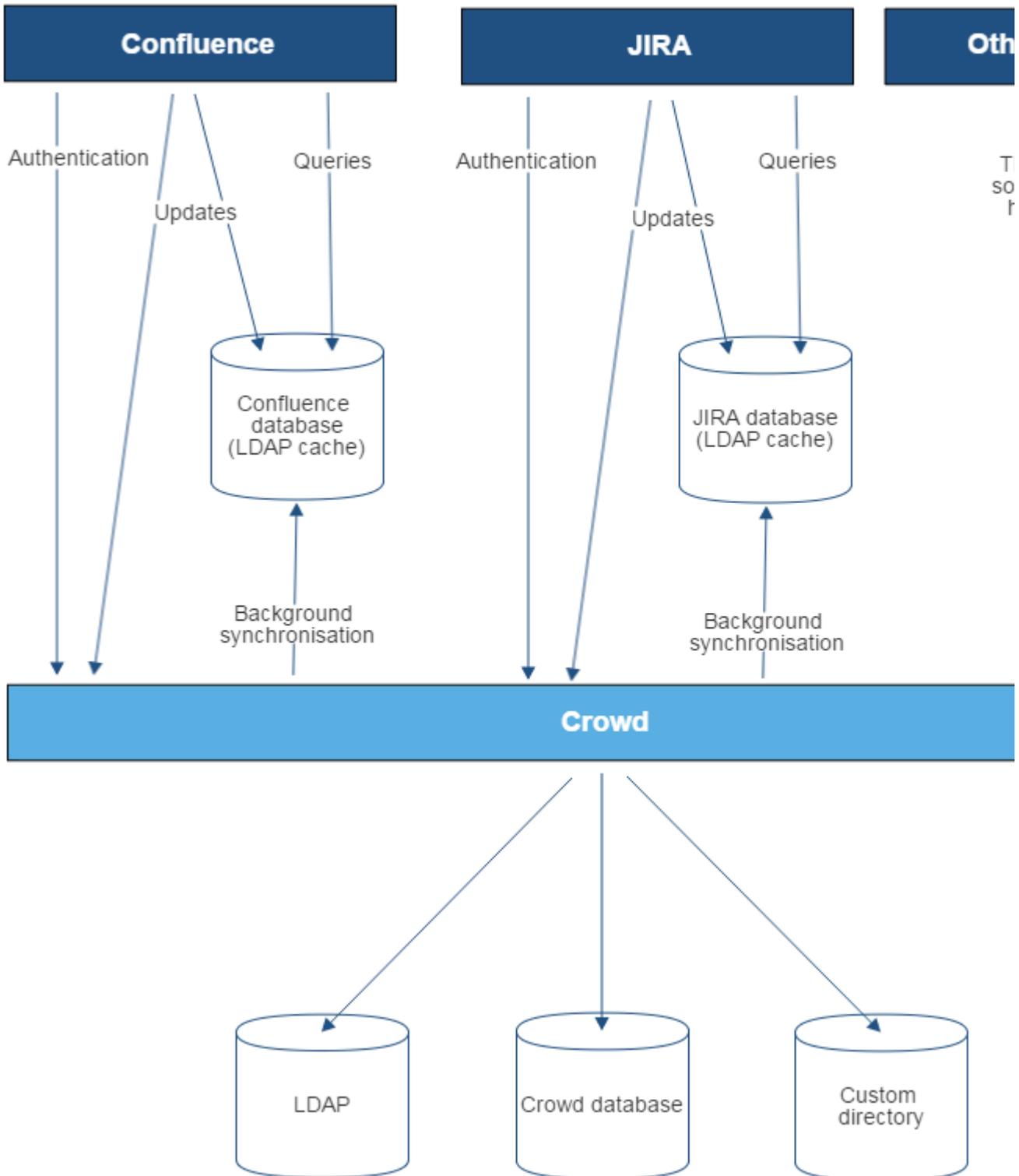


Diagram above: Confluence, JIRA and other applications connecting to Crowd for user management.

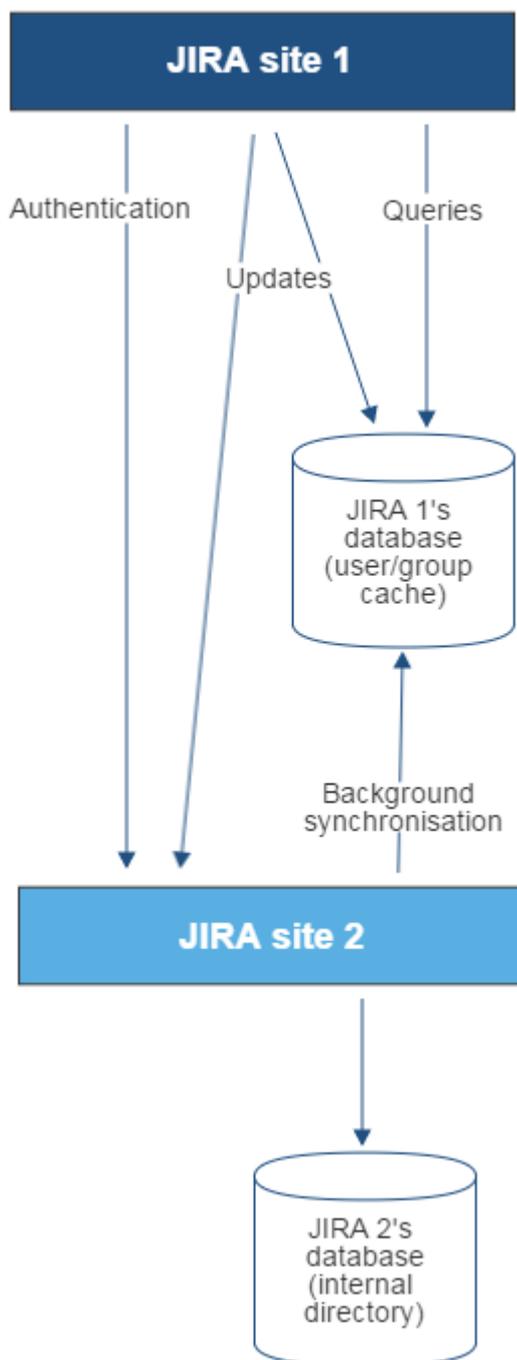


Diagram above: One JIRA site connecting to another for user management. JIRA site 2 does the user management, storing the user data in its internal directory.

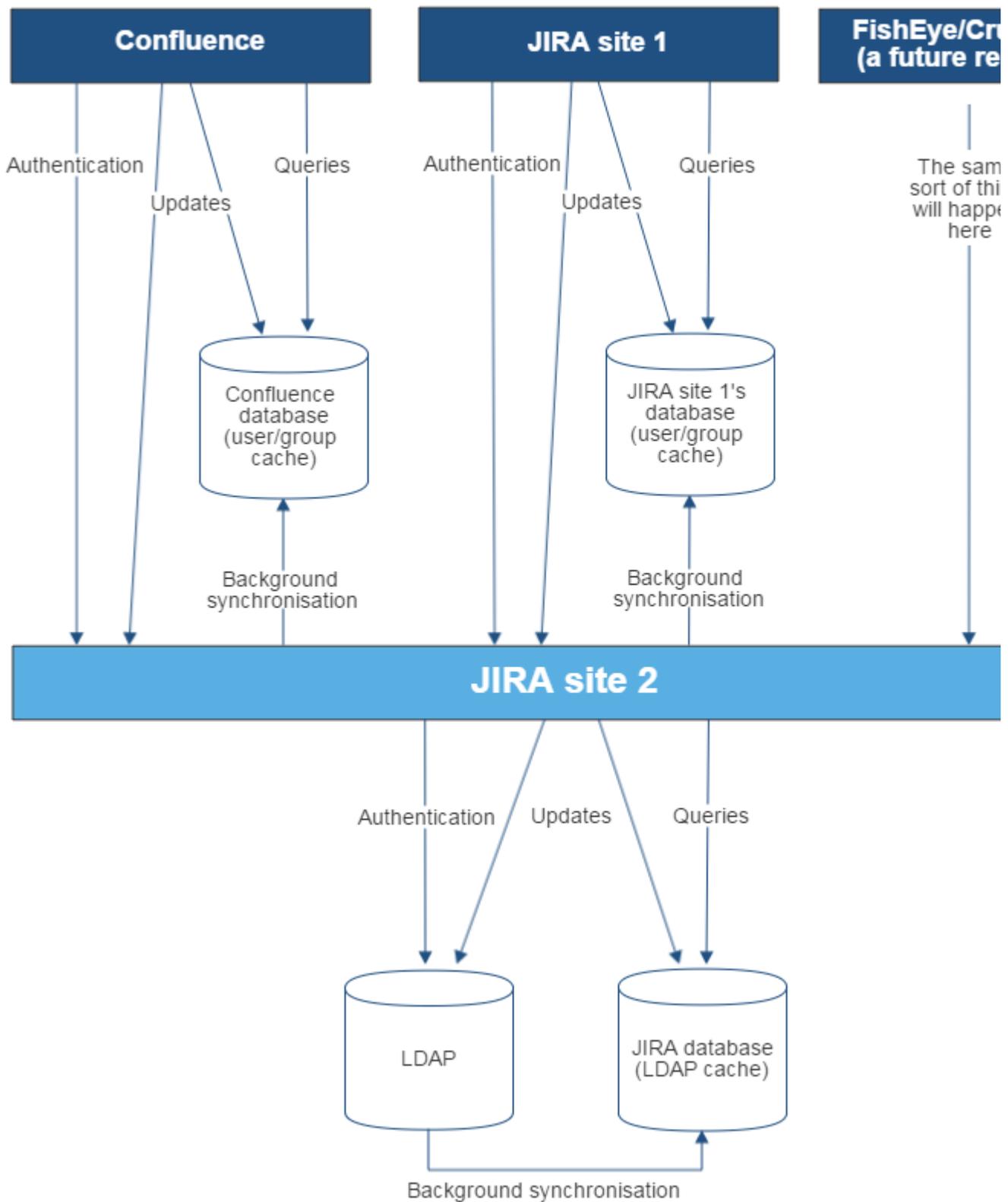


Diagram above: A number of applications connecting to JIRA (site 2) for user management, with JIRA in turn connecting to an LDAP server.

Related topics

Configuring user directories

- Configuring the internal directory
- Connecting to an LDAP directory
- Connecting to an internal directory with LDAP authentication
- Connecting to Crowd or another JIRA application for user management

- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)

Managing multiple directories

This page describes what happens when you have defined more than one user directory in JIRA. For example, you may have an internal directory and you may also connect to an LDAP directory server and/or other types of user directories. When you connect to a new directory server, you also need to define the **directory order**.

Avoid duplicate usernames across directories. If you are connecting to more than one user directory, we recommend that you ensure the usernames are unique to one directory. For example, we do not recommend that you have a user `jsmith` in both 'Directory1' and 'Directory2'. The reason is the potential for confusion, especially if you swap the order of the directories. Changing the directory order can change the user that a given username refers to.

Here is a summary of how the directory order affects the processing:

- The order of the directories is the order in which they will be searched for users and groups.
- Changes to users and groups will be made only in the first directory where the application has permission to make changes.

On this page:

- [Configuring the Directory Order](#)
- [Effect of Directory Order](#)
 - [Login](#)
 - [Permissions](#)
 - [Updating Users and groups](#)

Configuring the Directory Order

You can change the order of your directories as defined to JIRA. Select '**User Directories**' from the JIRA administration menu and click the blue up- and down-arrows next to each directory.

Directory Name	Type	Order
JIRA Internal Directory	Internal	↑ ↓
LDAP server	OpenLDAP (Read-Write)	↑ ↓

In situations where users are unable to change their passwords, check that a Delegated Authentication Directory is not the highest in the order of User Directories. As a workaround, you can change the order of User Directories, or alternatively use a connection to a LDAP directory instead.

Notes:

- Please read the rest of this page to understand what effect the directory order will have on authentication (login) and permissions in JIRA, and what happens when you update users and groups in JIRA.

Effect of Directory Order

This section summarizes the effect the order of the directories will have on login and permissions, and on the updating of users and groups.

Login

The directory order is significant during the authentication of the user, in cases where the same user exists in multiple directories. When a user attempts to log in, the application will search the directories in the order specified, and will use the credentials (password) of the *first occurrence of the user* to validate the login attempt.

Permissions

The directory order is significant when granting the user permissions based on group membership. If the same username exists in more than one directory, the application will look for group membership only in the first directory where the username appears, based on the directory order.

Example:

- You have connected two directories: The Customers directory and the Partners directory.
- The Customers directory is first in the directory order.
- A username `jsmith` exists in both the Customers directory and the Partners directory.
- The user `jsmith` is a member of group `G1` in the Customers directory and group `G2` in the Partners directory.
- The user `jsmith` will have permissions based on membership of `G1` only, not `G2`.

Updating Users and groups

If you update a user or group via the application's administration screens, the update will be made in the first directory where the application has write permissions.

Example 1:

- You have connected two directories: The Customers directory and the Partners directory.
- The application has permission to update both directories.
- The Customers directory is first in the directory order.
- A username `jsmith` exists in both the Customers directory and the Partners directory.
- You update the email address of user `jsmith` via the application's administration screens.
- The email address will be updated in the Customers directory only, not the Partners directory.

Example 2:

- You have connected two directories: A read/write LDAP directory and the internal directory.
- The LDAP directory is first in the directory order.
- Since you can create users in both directories, you can choose the directory in which you want to perform the update.

Related topics

Configuring user directories

- [Configuring the internal directory](#)
- [Connecting to an LDAP directory](#)
- [Connecting to an internal directory with LDAP authentication](#)
- [Connecting to Crowd or another JIRA application for user management](#)
- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)

Migrating users between user directories

Organizations will often migrate to or from LDAP engines, such as Active Directory or OpenLDAP, as they grow or acquire new companies, and need to migrate users into the same LDAP engine. As changes occur outside of JIRA, they will also need to be reflected within the JIRA user directories:

- JIRA can have multiple user directories (e.g. JIRA Internal, Delegated LDAP, LDAP Connector).
- The difference between the two is a connector will periodically synchronize user details against LDAP and can add/delete users and groups during that process. A delegated directory can only add users/groups upon the user's first login.
-  You can easily identify this by looking for the **Synchronize** option.
- Each directory will have **unique** users, groups and group memberships. This means there can be multiple users of the same username with different group memberships.

- Project Roles are global across all user directories.
- If you have the same user in multiple directories, the [effect of directory order](#) will apply. This means that if you add a new user directory and then change the order, so it is before your existing directory, your users will be selected from that directory first.
- When deactivating a user in LDAP, it will be deactivated in JIRA.
- When deleting a user in LDAP, it will be deleted in JIRA if it is not needed, or deactivated if it is (e.g. the user has comments).
- You can set up a User Directory with different [permissions settings](#) that will allow you to administer the groups in either LDAP, JIRA, or both.

This guide describes how to migrate users between the different user directories, as described in [Configuring user directories](#). You will need to log in as a user with the 'JIRA System Administrators' [global permission](#) to access the Settings menu.

On this page:

- Using the 'migrate users from one directory to another' functionality
- Migrating users by changing the directory order
- Migrating users manually

Using the 'migrate users from one directory to another' functionality

This functionality allows for the following scenarios:

- Migrate **all users** from JIRA Internal to Delegated LDAP
- Migrate **all users** from Delegated LDAP to JIRA Internal
- Migrate **all users** from Delegated LDAP to Delegated LDAP

However, it cannot be used for any of the following scenarios:

- Migrating a specific set of users or one single user from one directory to another
- Connector user directories — these can be easily identified, as they have a Synchronize option
- Migrating groups only
- Migrating users without their groups

It also has the following features:

- If you, the currently logged-in user, are in the directory to be migrated from, your user data will **not** be migrated.
- Users and groups will not be migrated if they already exist in the target directory. For example, consider a user that exists in JIRA Internal and JIRA Delegated LDAP but has different groups in JIRA Internal: when migrating from JIRA Internal to the JIRA Delegated LDAP, that user will be skipped and the groups will not be migrated.

To migrate users:

1. If the username needs to be changed as part of the migration, rename them (see [Managing users](#) for instructions).
2. Choose  **> User Management.**
3. Choose **User Directories.**
4. Choose **Additional Configuration & Troubleshooting** (section) **> Migrate users from one directory to another.**
5. *This option will not appear if there are no valid directories to migrate from/to.*

Administration Search JIRA admin

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

USER MANAGEMENT

- Users
- Groups
- Roles

Global Permissions

- User Preferences
- Password Policy
- JIRA User Server

USER DIRECTORIES

User Directories

Shared Filters

Shared Dashboards

User Directories ⓘ

The table below shows the user directories currently configured for JIRA.

The order of the directories is the order in which they will be searched for users and groups. Changes to users and groups will be made in the first directory where JIRA has permission to make changes. It is recommended that each user exist only in a single directory.

Directory Name	Type	Order	Operations
JIRA Internal Directory You cannot edit this directory because you are logged in through it, please log in as a locally authenticating user to edit it.	Internal	1	Edit
Active Directory (Connector) Last synchronised at 26/11/13 5:43 PM (took 0s). Full synchronisation completed successfully.	Microsoft Active Directory (Read Only, with Local Groups)	2	Disable Edit Test Synchronise
Active Directory (Delegated)	Microsoft Active Directory	3	Disable Edit

[Add Directory](#)

Additional Configuration & Troubleshooting

- Migrate users from one directory to another
- Directory Configuration Summary

6. Select the from and to directories and migrate the users:

Administration Search JIRA admin

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

USER MANAGEMENT

- Users
- Groups
- Roles

Global Permissions

- User Preferences
- Password Policy
- JIRA User Server

USER DIRECTORIES

User Directories

Shared Filters

Shared Dashboards

Migrate users from one directory to another ⓘ

⚠ The from and to directories will be disabled during the migration process. Some users may not be able to log in during the migration. Passwords will not be migrated.

⚠ If you, the currently logged in user, are in the directory to be migrated from, your user data will not be migrated.

From directory* JIRA Internal Directory
Users will be moved from this directory

To directory* Active Directory (Delegated)
Users will be moved to this directory

[Migrate users](#) [Cancel](#)

7. You will be shown a message telling you whether the migration was successful or not. In these example screenshots, only 61 out of 62 users could be migrated, as the user doing the migration was logged into the JIRA Internal Directory.

Administration Search JIRA admin

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

USER MANAGEMENT

- Users
- Groups
- Roles

Global Permissions

- User Preferences
- Password Policy
- JIRA User Server

USER DIRECTORIES

User Directories

Shared Filters

Shared Dashboards

Migrate users from one directory to another ⓘ

✔ 61 of 62 users successfully migrated.

From directory* JIRA Internal Directory
Users will be moved from this directory

To directory* Active Directory (Delegated)
Users will be moved to this directory

[Return to the User Directories list](#)

Migrating users by changing the directory order

This method is only applicable if moving users from the JIRA Internal Directory into an LDAP Connector and when LDAP will manage all their groups. Migrating users in this method will not move across any groups as the groups are separate from the JIRA Internal Directory to the LDAP Connector.

1. Add the LDAP Connector, as detailed in [Connecting to an LDAP directory](#).
2. Move the new user directory, so that it is ordered before the JIRA Internal Directory:

Administration Search JIRA admin

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

USER MANAGEMENT

- Users
- Groups
- Roles

Global Permissions

User Preferences

Password Policy

JIRA User Server

USER DIRECTORIES

User Directories

Shared Filters

Shared Dashboards

User Directories

The table below shows the user directories currently configured for JIRA.

The order of the directories is the order in which they will be searched for users and groups. Changes to users and groups will be made in the first directory where JIRA has permission to make changes. It is recommended that each user exist only in a single directory.

Directory Name	Type	Order	Operations
Active Directory (Connector)	Microsoft Active Directory (Read Only, with Local Groups)	1	Disable Edit Test Synchronise Last synchronised at 29/11/13 6:22 PM (took 0s). Incremental synchronisation completed successfully.
JIRA Internal Directory	Internal	2	Edit
Active Directory (Delegated)	Microsoft Active Directory	3	Disable Edit

Additional Configuration & Troubleshooting

- Migrate users from one directory to another
- Directory Configuration Summary

When users login, they will login to the LDAP Connector rather than the JIRA Internal Directory provided the usernames are identical.

Migrating users manually

If the user migration does not fall into the above scenario, you can migrate users by modifying the database. See this knowledge base article for instructions on how to do this: [Migrate local group memberships between directories](#). When

JRASERVER-27868 - Migrating users from one directory to another (part 2)

FUTURE CONSIDERATION

is completed,

JIRA will handle this in product.

Synchronizing data from external directories

For certain directory types, JIRA stores a cache of directory information (users and groups) in the application database, to ensure fast recurrent access to user and group data. A synchronization task runs periodically to update the internal cache with changes from the external directory.

On this page:

- Affected Directory Types
- How it Works
- Finding the Time Taken to Synchronize
- Manually Synchronizing the Cache
- Configuring the Synchronization Interval

Affected Directory Types

Data caching and synchronization apply to the following user directory types:

- LDAP** (Microsoft Active Directory and all supported LDAP directories) where permissions are set to **read only**.

- **LDAP** (Microsoft Active Directory and all supported LDAP directories) where permissions are set to **read only, with local groups**.
- **LDAP** (Microsoft Active Directory and all supported LDAP directories) where permissions are set to **read/write**.
- **Atlassian Crowd**.
- **Atlassian JIRA**.

Data caching and synchronization do not occur for the following user directory types:

- **Internal Directory with LDAP Authentication**.
- **Internal Directory**.

How it Works

Here is a summary of the caching functionality:

- The caches are held in the application database.
- When you connect a new external user directory to the application, a synchronization task will start running in the background to copy all the required users, groups and membership information from the external directory to the application database. This task may take a while to complete, depending on the size and complexity of your user base.
- Note that a user will not be able to log in until the synchronization task has copied that user's details into the cache.
- A periodic synchronization task will run to update the database with any changes made to the external directory. The default synchronization interval, or polling interval, is one hour (60 minutes). You can change the synchronization interval on the directory configuration screen.
- You can manually synchronize the cache if necessary.
- If the external directory permissions are set to read/write: Whenever an update is made to the users, groups or membership information via the application, the update will also be applied to the cache and the external directory immediately.
- All authentication happens via calls to the external directory. When caching information from an external directory, the application database does not store user passwords.
- All other queries run against the internal cache.

Finding the Time Taken to Synchronize

The '**User Directories**' screen shows information about the last synchronization operation, including the length of time it took.

Manually Synchronizing the Cache

You can manually synchronize the cache by clicking '**Synchronize**' on the '**User Directories**' screen. If a synchronization operation is already in progress, you cannot start another until the first has finished.

Screen snippet: User directories, showing information about synchronization

OpenLDAP	OpenLDAP (Read-Write)	 	Disable Edit Synchronise Last synchronised at 14/01/11 3:07 PM (took 65s).
Crowd	Atlassian Crowd	 	Disable Edit Synchronise Last synchronised at 14/01/11 2:39 PM (took 0s).

Configuring the Synchronization Interval

You can set the '**Synchronization Interval**' on the directory configuration screen. The synchronization interval is the period of time to wait between requests for updates from the directory server.

The length you choose for your synchronization interval depends on:

- The length of time you can tolerate stale data.
- The amount of load you want to put on the application and the directory server.
- The size of your user base.

If you synchronize more frequently, then your data will be more up to date. The downside of synchronizing more frequently is that you may overload your server with requests.

If you are not sure what to do, we recommend that you start with an interval of 60 minutes (this is the default setting) and reduce the value incrementally. You will need to experiment with your setup.

Related topics

Configuring user directories

- [Configuring the internal directory](#)
- [Connecting to an LDAP directory](#)
- [Connecting to an internal directory with LDAP authentication](#)
- [Connecting to Crowd or another JIRA application for user management](#)
- [Managing multiple directories](#)
- [Migrating users between user directories](#)
- [Synchronizing data from external directories](#)

Configuring projects

JIRA projects are a way of grouping issues together and a way of applying the same sets of configurations to issues. These configurations, such as workflow, issue types and screens, can be changed on a per project basis, so that each project can have a different set of configurations. Setting up a JIRA project effectively will enable your users to manage and complete their work quicker and more efficiently. This section of the documentation will take you through all the technical aspects of setting up your project, and give you information and tips on how to get the most out of your project.

Search the topics in 'Configuring projects':

Defining a project	Learn more about creating, configuring and deleting a project. Find out what elements make up the configuration of a project, and how to change them.
Configuring issues	Learn more about configuring your issue's fields, statuses, priorities and security, so that you can make your issues more effective for your organization.
Configuring permissions	Learn more about configuring permissions, both specific to your individual project, and applicable to JIRA as a whole.
Managing versions	Learn more about versions, how to create, edit and delete a version, and how to use them to further group issues in your project.
Managing components	Learn more about components, when and why to use them, and how to create, edit and delete them.
Screens, schemes and fields	Learn more about how issue screens and schemes are set up and maintained, how to configure your issue's fields, and how to create notification schemes for your project.
Using the issue collector	Learn more about how to configure and use the issue collector to get the most out of your projects.
Working with workflows	Learn more about workflows and your project. Workflows define how your issues are managed in your project, and you can configure the workflow to perform specific actions when you work on your issues.

Defining a project

This page tells you how to **add a new project**, **configure an existing project** or **convert an existing project to another project type**.

A JIRA project is a collection of issues. Your team could use a JIRA project to coordinate the development of a product, track a project, manage a help desk, and more, depending on your requirements. A JIRA project can also

be configured and customized to suit the needs of you and your team.

On this page:

- Before you begin
- Creating a project
- Convert a project type
- Re-index a project
- Delete a project
- Configuring a project
- A note about project administrators

Before you begin

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**. A JIRA administrator is able to create projects for all applications installed, but if the administrator does not have application access for that application, they will not be able to view the project after they have created it.

Creating a project

1. Click **Projects** (in header) > **Create project**.
2. Follow the wizard to create the project.

About the project types:

- Depending on which JIRA applications you have installed, you may have more than one project type available.
- Each project type has a specific set of features.
- All users on the JIRA instance will be able to see all projects, but what features they see and what actions they can take are determined by their application access and the [project specific permissions](#).

About shared configurations:

- When you create a new project from a template, that project is created with its own set of schemes. These schemes are:
 - a permission scheme (default)
 - a notification scheme (default)
 - an issue security scheme
 - a workflow scheme
 - an issue type scheme
 - an issue type screen scheme
 - a field configuration scheme (default)
- Sometimes you may wish to share schemes among your projects, so that editing one scheme changes that scheme in several projects at once.
- You can select **Create with shared configuration** to select an existing project and to use that project's schemes. Note that when you're sharing schemes, any change to the scheme will affect all the projects using that scheme.

About the project details:

- The *project key* will be used as the prefix of this project's issue keys (e.g. 'TEST-100'). Choose one that is descriptive and easy to type.

- The *project lead* is a unique project role. Choose the person who manages the project as the project lead. If there is only one user in your JIRA system, the Project Lead will default to that person and this field will not be available.
- If you're creating a project using a project type related to an application you currently do not have access to, JIRA will display a checkbox that will allow you to grant yourself access to that application. This will add you to the default group of that application, and you will count as a user for that license.

Convert a project type

At some point you may wish to convert an existing project to a different project type. For instance, you can convert a JIRA Software project to a JIRA Core project at the end of a JIRA Software evaluation period, or when your team grows. You can only convert to project types of JIRA applications that you have installed. Note that a project administrator may also change the project type.

1. Choose  > **Projects**, and select the relevant project..
2. Select **Details** in the **Project settings** menu.
3. Change the project type, and click **Save details**. Only project types for applications you have installed will be available.

You can review more on project types and what your users will see on the [project type and application overview](#) page.

Re-index a project

To provide fast searching, JIRA creates an index of the text entered into issue fields. It's 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. You [regenerate the index for your entire JIRA instance](#), or you can do it on a per project basis. Follow these instructions to re-index a single project.

1. Choose  > **Projects**, and select the relevant project..
2. Select **Re-index project** in the **Project settings** menu.
3. Click **Start project re-index**.

Delete a project

If you're thinking about deleting a project from your JIRA instance, remember that you can't reinstate it from within JIRA. Once the project's been deleted, it can only be recovered by reinstating a backup or an XML copy, and this is no trivial task. Make sure you're comfortable deleting the project before proceeding. Deleting a project will only delete the issues, versions and components associated with the project. It won't delete any of the associated schemes, workflows or issues types, or any content that could be shared with another project.

1. Choose  > **Projects**, and select the relevant project..
2. Select **Delete project** in the **Project settings** menu.
3. Click **Delete** to begin deleting the project.
4. Acknowledge the project has been deleted.

Configuring a project

1. Navigate to the settings page for the project by doing either of the following:
 - Choose  > **Projects**, and select the relevant project.

- Navigate to the desired project's summary via the **Project** drop-down, and click the **Project settings** link.
2. Use the links on the left to navigate between the different project settings. Read the sections below for a description of each setting.

[Project details](#) | [Issue types](#) | [Workflows](#) | [Screens](#) | [Fields](#) | [Settings](#) | [Roles](#) | [Versions](#) | [Components](#) | [Permissions](#) | [Notifications](#) | [Development tools](#)

Project details

Click **Details** in the **Project settings** sidebar, and edit the project details as desired. Once you've completed your edits, don't forget to click the **Save** button. Note the following:

- Editing the project key: This is not a simple task. Read this page before you edit the project key: [Editing a project key](#).
- Using the Wiki Style Renderer in the project description: You can use the [Wiki Style Renderer](#) to display rich text (HTML) in your project description.
- Choosing a project avatar: If you don't want to use a project avatar, you can upload a transparent pixel. This effectively loads the transparent pixel, which means you won't see an image.

About project categories:

Categories can be viewed/created via



> **Projects > Project Categories.**

Why are categories useful? JIRA can search for all the issues in a particular project category (e.g. `category = "buildeng"` in an advanced search), and can display projects sorted by the project category. A JIRA project can only belong to one category. Please note that a project category is not part of a project hierarchy. Also, JIRA does not support sub-projects or parent projects.

Issue types

JIRA enables you to keep track of different types of things — bugs, tasks, helpdesk tickets, etc — by using different *issue types*. You can also configure each issue type to act differently, e.g. to follow a different process flow or track different pieces of information.

Click either **Issue Types** in the left menu or one of the issue types under it, e.g. **Bug, Task, Story**, etc:

- **Issue Types:** Click this to configure which issue types apply to this project (choose an [issue type scheme](#) or [edit the existing scheme](#)). You can also configure the workflow, fields and screens for the issue type in the project, but it is easier to do this by clicking one of the issue types.
- **One of the issue types (e.g. Bug, Task, Story):** Click this to configure the workflow/screen for the issue type in the project. The workflow screen (**Workflow** tab) shows the [workflow designer](#). The screen (**View** tab) shows the [screen designer](#).

Workflows

Your JIRA issues can follow a process that mirrors your team's practices. A *workflow* defines the sequence of steps (or statuses) that an issue will follow, e.g. Open, In Progress, Resolved. You can configure how issues will transition between statuses, e.g. who can transition them, under what conditions, and which screen will be displayed for each transition.

- **Workflow Scheme** — the project's [workflow scheme](#) determines which [workflows](#) (issue state transitions) apply to issue types in this project.

Screens

JIRA allows you to display particular pieces of issue information at particular times, by defining *screens*. A screen is simply a collection of fields. You can choose which screen to display when an issue is being created, viewed, edited, or transitioned through a particular step in a workflow.

- **Screen Scheme** — the project's [screen scheme](#) determines which [screens](#) are displayed for different

issue operations (view, edit, create);

OR

- **Issue Type Screen Scheme** — the project's [issue type screen scheme](#) determines which [screens](#) are displayed for different issue operations (view, edit, create), for different issue types.

Fields

JIRA enables you to define field behavior: each field can be required/optional, rich text/plain text, hidden/visible. You define this behavior by using a *field configuration*.

- **Field Configuration Scheme** — the project's [field configuration scheme](#) determines which [field configuration](#) applies to issue types in this project. (A [field configuration](#) determines each field's overall visibility, requiredness, formatting (wiki/rich-text or plain) and help-text).

Settings

- **Application Links** (Configure project links) — if you have linked your JIRA instance to other Atlassian applications, like Confluence, FishEye or other JIRA instances, you will be able to link this JIRA project to areas of those applications that contain information relating to your project or team. For example, Confluence spaces, FishEye repositories, JIRA projects (in another JIRA instance), etc. This allows you to take advantage of integration points between these applications. See [Using AppLinks to link to other applications](#) for information about application links and project links.

Roles

Different people may play different roles in different projects — the same person may be a leader of one project but an observer of another project. JIRA enables you to allocate particular people to specific roles in your project.

- **Project Lead** — user fulfilling the role of project leader. Used as the 'Default Assignee' (except for JIRA Software projects where it is set to 'Unassigned'), and potentially elsewhere in JIRA (e.g. 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'. There are also [default component assignees](#).
 By default, new projects also have their 'Default Assignee' set to 'Unassigned.' You can change this here if you want to set it to be a specific role, i.e. 'Project Lead.'
- **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.

Versions

Issues can be grouped in JIRA by allocating them to versions. For example, if you are using JIRA to manage the development of a product or manage the build of a house, you may want to define different *versions* to help you track which issues relate to different phases of your product or build (e.g. 1.0, 1.1, 1.2, 2.0, 2.0.1). JIRA can help you manage, release and archive your versions. Versions can also have a Release Date, and will automatically be highlighted as "overdue" if the version is unreleased when this date passes.

- **Versions** — versions defined in the project. See the [version management](#) page for details.

Components

You may want to define various *components* to categorize and manage different issues. For a software development project, for example, you might define components called "Database", "Usability", "Documentation" (note that issues can belong to more than one component). You can choose a Default Assignee for each component, which is useful if you have different people leading different sub-teams in your project.

- **Components** — logical groups that this project's issues can belong to. See the [component management](#) page for details.

Permissions

JIRA allows you to control who can access your project, and exactly what they can do (e.g. "Work on Issues", "Comment on Issues", "Assign Issues"), by using *project permissions*. You can also control access to individual issues by using *security levels*. You can choose to grant access to specific users, or groups, or roles (note that roles are often the easiest to manage).

- **Permission Scheme** — the project's [permission scheme](#) determines who has permission to view or change issues in this project.
- **Issue Security Scheme** — the project's [issue security scheme](#) determines what visibility levels issues in this project can have.

Notifications

JIRA can notify the appropriate people when a particular event occurs in your project (e.g. "Issue Created", "Issue Resolved"). You can choose specific people, or groups, or roles to receive *email notifications* when different events occur. (Note that roles are often the easiest to manage.)

- **Notification Scheme** — the project's [notification scheme](#) determines who receives email notifications of changes to issues in this project.
- **Email** — specifies the 'From' address for emails sent from this project. Only available if an SMTP email server has been configured in JIRA.

 Please note, the **Default Notification Scheme** (shipped with JIRA) is associated with all new projects by default. This means that if you have an outgoing (SMTP) mail server set up, that email notifications will be sent as soon as there is any activity (e.g. issues created) in the new project.

Development tools

The Development tools section is only available on JIRA Software projects, and can only be viewed by JIRA Software users. It gives you an overview of the development tools that are connected and which users can use the integration features between them:

- **View permission** - This section lists which users can see the development tools integration features (like the **Create Branch** link) on the view issue screen, as well as other development-related information, like commits, reviews and build information. This ability is controlled by the "View Development Tools" project permission.
- **Applications** - This section shows which development tools are connected to JIRA via application links and are eligible to use the development tool features in JIRA.

A note about project administrators

A project administrator in JIRA is someone who has the project-specific **Administer Projects** [project permission](#), but not necessarily the **JIRA Administrator** [global permission](#).

Without the **JIRA Administrator** [global permission](#), however, project administrators can do the following:

- Edit the project name
- Edit the project description
- Edit the project avatar image
- Edit the project URL
- Edit the project lead
- Edit [project role membership](#)
- Change the project type
- Define [project components](#)
- Define [project versions](#)
- View, but not select nor edit the project's schemes (notification scheme, permission scheme, etc)

Changing the project category of a JIRA project requires **JIRA Administrator** [global permission](#).

Editing a project key

Editing a project key is not a trivial task. You should choose key that will suit your long-term needs when creating a project, rather than rely on editing the

project key after the project is created. However, there are situations where you need to change the key for an existing project, e.g. change of product name.

The instructions on this page show you how to change the project key and describe the implications of such a change.

- [Before you begin](#)
- [Editing the project key](#)
- [Notes for change management](#)
- [Related topics](#)
- [Notes for developers](#)

Before you begin

- Your desired project key must conform to the project key format restrictions specified in your JIRA applications. By default, the project key format must be at least 2 characters long and contain only uppercase letters. You can change the project key format to enforce different restrictions. See [Changing the project key format](#) for instructions.
- Perform this change during a low usage period — JIRA applications will start a [background re-index](#) when you save your updated project key. This can have a performance impact on your instance. Note, you cannot choose a 'Lock JIRA and rebuild index'. The background index will be faster anyway, as it is limited to issues for the project.
- Communicate changes to your users — Ensure that you are aware of the consequences of changing the project key, and have adequately prepared your users for the changes. See the [Changes](#) section below.

Editing the project key

1. Choose



> **Projects**, and select the relevant project..

2. Locate the project that you'd like to change.
3. Select **Edit** in the **Actions** column of the project you want to change.
4. Edit the project key, and click **Save details**. Only project types for applications you have installed will be available.

This will start a project re-index, which you need to acknowledge when it finishes.

Note:

- If you update any other project detail fields, you'll see the changes immediately. You won't need to wait for the re-index to finish.
- If you cancel the background re-index, you will have trouble searching for issues related to the project. If you do need to cancel it, you can run it again later to fix these problems.

Post-update tasks

- **Fix the project entity links** — When you connected JIRA to another Atlassian application, entity links would have been automatically created between your JIRA projects and the relevant "projects" in other applications, e.g. Confluence spaces. If you change the key of a JIRA project, you will need to fix the project entity links, as described in [Creating links between projects](#).
- **Updating JIRA Software agile board filters** - If your JIRA Software agile boards use the old project key, the board filters may need to be updated to reflect the new project key. Otherwise the board might not display issues from the renamed project.

Notes for change management

While editing the project key is a major change, in most cases, your JIRA project will work as you'd expect with a new key. There are a few cases that you should be aware of, which are listed below. We recommend reviewing these and advising your users accordingly.

- The old project key can be used in JQL queries — Users won't have to update issue filters that reference the old project key.
- If you use Confluence with JIRA, the JIRA issue macros in Confluence will continue to work. Please note, if you don't see the change straight away, allow some time for the cache to refresh.
- You won't be able to create a new project with the old project key. However, you can change the renamed project back to the old project key. If you delete the project, all associated keys will be freed and you'll be able to re-use them.
- Links will work, whether they are inside JIRA or from external sources. However, link aliases will not be updated — For example, if you have a link to an issue 'EXAMPLE-1' in the description of an issue, and you change the project key 'EXAMPLE' to 'DEMO', then the alias 'EXAMPLE-1' will not be updated to 'DEMO-1'. The link will still direct you to DEMO-1 though.
- If you are using a gadget with a global filter, you will need to update the filter after the project is renamed.
- All attachments will be accessible after the project key change. Please note however, that the directory that they are stored in (under the `<JIRA Home>\data\attachments` directory) will be retain the old project key. For example, if you change a project's key from TEST to DEMO, the attachments will be stored under `<JIRA Home>\data\attachments\TEST`.
- If you export a renamed project, then import it, it will have the updated project key, i.e. the original project key will not be retained. In fact, all historical keys for that project will be removed. There is a workaround for this that involves changing data directly in your database, see this [Answers post](#).

Related topics

Changing the maximum project key length — You can change the maximum characters allowed for a project key. Navigate to the General Configuration page of the JIRA administration console, as described on [Configuring JIRA application options](#), and change the **Maximum project key size** field. **Changing the project key format** — You can change the format of a project key. This restricts the format of a project key when it is [created](#) or [edited](#) (as described above). For instructions, see [Changing the project key format](#).

Notes for developers

- REST API calls will still work with old project key — REST calls that specify an issue key will work with the old issue key after the project key has changed. For example, `/rest/api/issue/EXAMPLE-100` will still work after the project key is changed from EXAMPLE to DEMO.
- We have created a new event, ProjectUpdatedEvent. This event is triggered any time a project's details are changed, including changing the project key.
- If you need to retrieve all issue keys and project keys (historical and current), you can do this via the following:
 - REST:
 - Get all project keys for a project: `/rest/api/2/project/<project key>?expand=projectKeys`
 - Java API:
 - Get all project keys: `com.atlassian.jira.project.ProjectManager#getAllProjectKeys`
 - Get all issue keys for an issue: `com.atlassian.jira.issue.IssueManager#getAllIssueKeys`

Changing the project key format

JIRA provides the ability to specify the format of project keys within the system. This allows you to restrict the format of a project key, when a project key is [created](#) or [edited](#).

A project key format is defined via a regular expression 'rule' that governs the valid project key format. By default, the JIRA project key configuration requires two or more uppercase alphabetical characters — based on the regular expression `([A-Z][A-Z]+)`.

On this page:

- Before you begin
- Configuring the project key format
- Related topics

Before you begin

- Ensure that you choose a **supported project key format**. Only formats that meet all of the following rules are supported:
 - The first character must be a letter,
 - All letters used in the project key must be from the [Modern Roman Alphabet](#) and upper case, and
 - Only letters, numbers or the underscore character can be used.
 Examples:
 - Examples of supported keys: PRODUCT_2013, R2D2, MY_EXAMPLE_PROJECT.
 - Examples of unsupported keys: 2013PROJECT (*first character is not a letter*), PRODUCT-2012 (*hyphens are not supported*).
- You cannot configure the issue key pattern, as JIRA expects this key to conform to specific rules. By default, JIRA issue keys (or issue IDs) are of the format **<project key>-<issue number>**, e.g. ABC-123. For example, you can't show the issue number before the project key.
- If a number of issues have already been created in your JIRA installation, then *changing the project key format is not recommended*. If you must change the project key pattern after issues have already been created, use a regular expression that allows a more 'permissive' project key pattern than the current one (e.g. use a regular expression which will still be valid for existing project keys defined in your JIRA installation).

If you have integrated JIRA with [Bamboo](#), do not change JIRA's default project key format as Bamboo only supports this key format.

Configuring the project key format

The `jira.projectkey.pattern` property allows JIRA administrators to specify a Perl5 regular expression value that defines the rule for a valid project key. Further information on Perl5 is available [here](#).

This property and its regular expression value can be defined through the **Advanced Settings** page. This is described below.

Step 1. Configure a pattern for your project key syntax

1. Navigate to the JIRA Advanced settings page, as described on [Configuring advanced settings](#).
2. Find the `jira.projectkey.pattern` property and click its value to modify it. Below is a list of common examples and patterns:

Pattern Requested	Expression needed	Resulting Issue IDs	Comments
XXYY, where X indicates two fixed letters, Y represents two fixed digits	<code>([A-Z]{2}[0-9]{2})</code>	TQ09-01, TQ09-02, etc.	[A-Z] Any character from A to Z {2} Matches the preceding character 2 times exactly [0-9] Any character (i.e. digit) from 0 to 9

XZ+, where X indicates one fixed letter, Z+ represents one or more letters, digits or underscore characters	([A-Z][A-Z_0-9]+)	ACAT_51-1, AAA5-1330, A_20_A091-15, etc.	[A-Z] Any characters from A to Z [A-Z_0-9] Any character from A to Z, 0 to 9 or the underscore character. + specifies [A-Z_0-9] as one or more characters from A to Z, 0 to 9 or the underscore character.
---	-------------------	--	--

Please note:

- JIRA prepends the regular expression specified with '^' and closes it with '\$' for an exact matching rule within the system.
- The project key only supports uppercase characters, as [stated above](#). Hence, for simplicity, use uppercase characters in your expressions as JIRA will convert any lowercase characters to uppercase ones.

Step 2. Test your regular expression

A variety of tools allow searching using a Regular Expression. Most text editors will allow a Regular Expression search. There are also a variety of websites available to for testing a Regular Expression available from an Internet search.

(Optional) Step 3. Customize the project key description and warning

In addition to the project key format, you can also customize the following properties in the `jira-config.properties` file:

- `jira.projectkey.description` — a configurable description (to match the project key pattern) displayed on project creation
- `jira.projectkey.warning` — if JIRA detects that the project key entered does not match the `jira.projectkey.pattern`, it will throw the error message defined in `jira.projectkey.warning`. You can change this error message, so that when a user keys in the wrong format, they will be informed of the correct pattern to use.

Related topics

- **Changing the maximum project key length** — You can change the maximum characters allowed for a project key. Navigate to the General Configuration page of the JIRA administration console, as described on [Configuring JIRA application options](#), and change the **Maximum project key size** field.
- [Defining a project](#)
- [Editing a project key](#)
- [Configuring advanced settings](#)

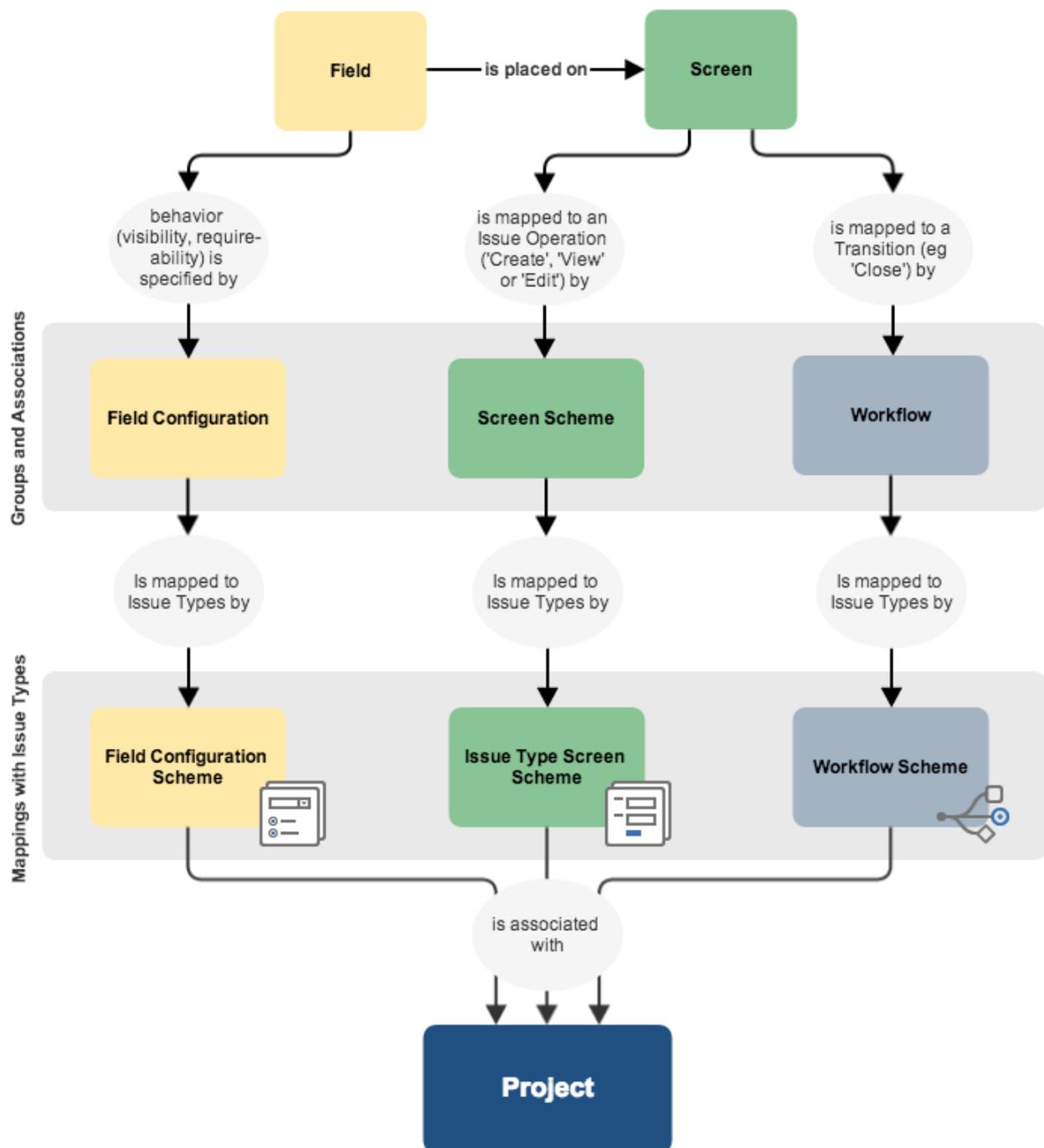
Configuring issues

Overview

To help you tailor JIRA to your organization's needs, JIRA enables you to manipulate the display and behavior 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 [wo](#)

rkflow transition (e.g. 'Resolve Issue', 'Close Issue')



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').
- **Workflow** — defines the steps (i.e. statuses) and transitions to other steps that an issue moves through during its lifecycle. Screens can also be mapped to different transitions of a workflow.

- [Field configuration scheme](#) — associates field configurations with [issue types](#), which in turn is applied to projects. This allows you to specify different behaviors for a field, for each type of issue in a given project.
- [Issue type screen scheme](#) — associates screen schemes with [issue types](#), which in turn is applied to projects. This allows you to specify different screens for a particular operation (e.g. 'Create Issue'), for each type of issue in a given project. 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'.
- [Workflow scheme](#) — associates Workflows with [issue types](#), which in turn is applied to projects. This allows you to specify different workflows for each type of issue in a given project.
- [Issue type scheme](#) — is applied to projects and defines (or *restricts*) which [issue types](#) are available to those projects.
 - **i** If the [field configuration scheme](#), [issue type screen scheme](#), and [workflow scheme](#) associated with a given project contain associations with other issue types that are not specified in the project's [issue type scheme](#), then those other issue types will be ignored by the project since the project's Issue Type Scheme restricts what issue types the project can use.

Related topics

- [Configuring built-in fields](#)
 - [Defining issue type field values](#)
 - [Associating issue types with projects](#)
 - [Defining priority field values](#)
 - [Associating priorities with projects](#)
 - [Defining resolution field values](#)
 - [Defining status field values](#)
 - [Translating resolutions, priorities, statuses, and issue types](#)
- [Issue fields and statuses](#)
- [Configuring issue-level security](#)

Configuring built-in fields

Each issue has a number of built-in fields, and some of the built-in fields can be customized as follows:

- [Defining issue type field values](#)
 - [Associating issue types with projects](#)
- [Defining priority field values](#)
 - [Associating priorities with projects](#)
- [Defining resolution field values](#)
- [Defining status field values](#)
- [Translating resolutions, priorities, statuses, and issue types](#)

Defining issue type field values

JIRA applications ship with a set of default issue types to help you get started. You can add, edit and delete your own custom issue types to suit the needs of your team. The diagram on [Configuring issues](#) shows how issue types relate to other entities in JIRA applications.

Note that you can also:

- Control the set of available issue types for each project — see [Associating issue types with projects](#).
- Control the display order of available issue types and the default issue type for each project — see [Associating issue types with projects](#).
 - **i** *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.
- Associate particular issue types with specific fields, screens and workflow — for details see [Associating field behavior with issue types](#), [Associating screen and issue operation mappings with an issue type](#), and [Managing your workflows](#), respectively.

On this page:

- [Creating an issue type](#)
- [Deleting an issue type](#)
- [Editing an issue type](#)

Tip: You can quickly configure the workflow/screen design of an existing

issue type for a project via the project administration page. See [Defining a project](#) for details.

Creating an issue type

When creating a new issue type in JIRA applications, you can create either a new standard or sub-task issue type. However, to create a sub-task issue type, you must [enable sub-tasks](#).

You can also create sub-tasks on the **Sub-Tasks** page. See [Configuring sub-tasks](#) for details.

1. Choose



> **Issues.**

2. Select **Issue Types** to view all issue types used by your JIRA applications.
3. Select **Add Issue Type** and enter the following details:
 - **Name** — enter a short phrase that best describes your new issue type
 - **Description** — enter a sentence or two to describe when this issue type should be used
 - **Type** — specify whether the issue type you are creating is a **Standard** issue type or a **Sub-Task** issue type. Sub-tasks are associated with individual **Standard** issues. Note that this option will not be available if [sub-tasks are disabled](#).
4. Select **Add** to create your new issue type.
 - **i** Your new issue type will be automatically added to the **Default Issue Type Scheme**. You may want to also add it to other issue type schemes — for more information, see [Associating issue types with projects](#).

Deleting an issue type

Before you begin:

- If any issues of the Issue Type you are about to delete exist in your JIRA installation, please ensure this Issue Type has the following requirements (to ensure JIRA prompts you to choose a new Issue Type for those issues):
 - the same [workflow](#) in all [workflow schemes](#) that are associated with one or more projects.
 - the same [field configuration](#) in all [field configuration schemes](#) that are associated with one or more projects.
 - the same [screen scheme](#) in all [issue type screen schemes](#) that are associated with one or more projects.
- **Alternatively**, you can simply search for all issues that currently use the Issue Type which you are about to delete and perform a bulk move to change those issues to a different Issue Type.

1. Choose



> **Issues.**

2. Select **Issue Types** to open the Issue Types page, which lists all issue types.
3. Click the **Delete** link (in the **Operations** column) for the issue type that you wish to delete.
4. Complete the fields.

Editing an issue type

1. Choose



> **Issues.**

2. Select **Issue Types** to open the Issue Types page, which lists all issue types.
3. Click the **Edit** link (in the **Operations** column) for the issue type that you wish to edit.
4. Edit the **Name**, **Description** and/or **Icon** as described above for creating an issue type.

i Please note: To reorder an Issue Type, or set it as a default, see [Associating issue types with projects](#). (*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.)

Associating issue types with projects

What is an 'issue type scheme'?

An 'issue type scheme' defines a subset of [issue types](#), which:

- restricts the set of available **issue types** for a project, and
 - controls the order of available issue types and the default issue type shown to your users for a project.
- i** The 'default issue type' is the issue type displayed in the selection-box when a user creates an issue.

A single issue type scheme can be 're-used' across multiple projects, so that a group of similar projects (i.e. projects which might be used for similar purposes) can share the same issue type settings.

For example, all projects in your company may fit one of two 'purpose' categories:

- Development-related projects or
- Support-related projects.

On this page:

- What is an 'issue type scheme'?
- Managing issue type schemes
- Choosing a project's issue type scheme
- Using the Issue Type Migration Wizard

Hence, you could create one scheme called *Development Issue Type Scheme* (with issue types *Bug* and *Feature*) and another called *Support Issue Type Scheme* (with issue types *Development Query* and *Support Request*). You can then associate each of these schemes with the appropriate project(s), for which there may be a plethora.

This provides your users with a different set of issue types based on the project they decide to create issues in and furthermore reflects the purpose behind creating these issues.

Your future maintenance workload is minimized, because any change you make to an issue type scheme is made across all projects that are associated with the scheme. In the example above, adding a new issue type to all support-related projects only requires the simple step of adding the issue type to the *Support Issue Type Scheme*.

! Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Managing issue type schemes

1. Choose

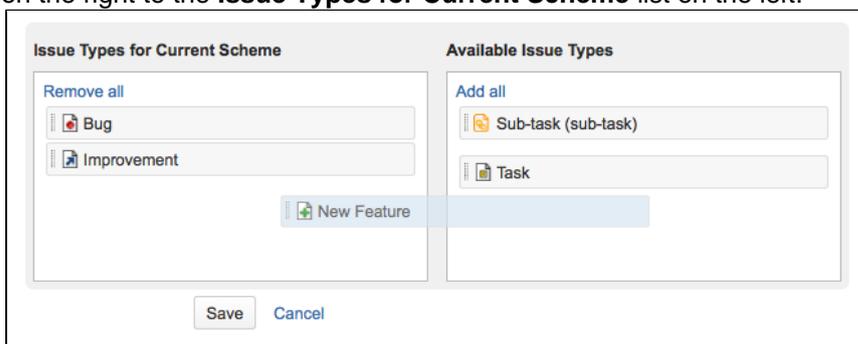


> **Issues.**

2. Select **Issue Types > Issue Type Schemes** to open the Issue Type Schemes page, which displays all existing issue type schemes, their related issue types and their associated projects.

Creating a new issue type scheme

1. Go to the **Issue Type Schemes** tab (see [above](#)).
2. Select **Add Issue Type Scheme** and enter a Scheme Name and Description.
 - i** Ensure that the Scheme Name is meaningful to other administrators, who will be able to reuse the scheme.
3. To add issue types to your scheme, drag and drop an issue type from the **Available Issue Types** list on the right to the **Issue Types for Current Scheme** list on the left:



4. If you need an issue type that does not currently exist, you can easily add this by using the **Add New Issue Type** button and dialog box. This will **add the issue type** to your JIRA system and also add it to **Issue Types for Current Scheme** list on the left.
5. To reorder the issue types, drag and drop them into the preferred positions. *Reordering* issue types

- changes the order in which they are displayed in the selection-box when a user creates an issue.
6. Set the **Default Issue Type** for the new scheme from the drop-down list. **Please Note:**
 - The 'default issue type' is the issue type displayed in the selection-box when a user creates an issue.
 - The issue types in this list depend on the issues in the **Issue Types for Current Scheme** list on the left.
 - The **None** option means that there is no default value. If this option is selected, the system will show the first Issue Type listed in the **Issue Types for Current Scheme**.
 - The **Issue Type** is remembered as long as you keep creating issues in the same project. Once you change projects or log off the system, it goes back to the *default* value.
 7. Click the **Save** button to create your issue type scheme.

Editing an issue type scheme

- Go to the **Issue Type Schemes** tab (see [above](#)).
- Click the **Edit** link (in the **Operations** column) to access and edit the relevant issue type scheme.

i Please note:

- The process of editing a scheme is identical to the creation process. While editing your issue type scheme, you can set the default default issue type and reorder, add or remove issue types.
- If an issue type scheme has been associated with one or more JIRA projects ([below](#)) and:
 - issues of the issue types (defined by this issue type scheme) already exist in any of these JIRA projects and
 - you then want to remove one or more of these issue types from this issue type scheme, you will be prompted to use the [Issue Type Migration Wizard](#) (below). This wizard will move your issues from the original issue type (which will no longer be applicable) to a valid one. If you cancel this process at any time, your changes will not be saved.

Associating an issue type scheme with projects

1. Go to the **Issue Type Schemes** tab (see [above](#)).
2. Click the **Associate** link (in the **Operations** column) for the relevant Issue Type scheme.
3. Using the multi-select **Project** box, choose the JIRA projects that you wish to apply your issue type scheme to.
4. Select **Associate** and all selected projects will change from their current scheme to the selected scheme.

i **Please note:** If a project you are attempting to associate your new issue type scheme with has issues with issue types which have not been added to this new issue type scheme, you will be asked to use the [Issue Type Migration Wizard](#) (below) to migrate the issues to a new issue type (made available by the new issue type scheme).

Choosing a project's issue type scheme

You may want to change a project to use a different set of issue types.

i This is effectively the same as associating an issue type scheme with projects ([above](#)), but is performed from a project's **Project Summary** administration page (and you cannot choose multiple projects in one action).

1. Choose



> **Projects**, and select the relevant project.

2. In the **Issue Types** section, click the name of the current scheme to display the details of the project's issue type scheme.
3. Click the **Actions** drop-down menu and choose **Use a different scheme**.
4. There are three ways you can select your issue type scheme. Select the radio button that is most relevant:
 - a. **Choose an 'existing issue type scheme'** — If you know the name of your scheme (e.g. 'Development 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.
 - b. **Choose a scheme that is the 'same as an existing project'** — Select this option 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. 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.

- c. **Create a new scheme and associate with current project** — Select this option if you cannot 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](#).
5. 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](#).

Using the Issue Type Migration Wizard

The 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 is similar to the bulk move function, except 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 the '**Confirm**' button, the wizard will migrate your issues to the new issue types and then complete your action.

Defining priority field values

An issue's priority defines its importance in relation to other issues, so it helps your users determine which issues should be tackled first. JIRA comes with a set of default priorities, which you can modify or add to. You can also choose different priorities for your projects.

Managing priorities

To manage priorities and complete the actions listed below, go to



> **Issues**, and choose **Priorities**. You need to have the **JIRA Administrators** global permission to get there.

Icon and name	Description	Color	Order	Used by	Actions
Blocker	Blocker		↓	3 schemes	Edit Delete
Highest	Our top priority issues		↑ ↓	1 scheme	Edit Delete
High	High priority issues		↑ ↓	2 schemes	Edit Delete
Medium	Medium priority issues		↑ ↓	2 schemes	Edit Delete
Low	Low priority issues		↑	3 schemes	Edit Delete

1. This is where all **priorities** live.

2. **Icon and name** representing a priority.
3. Priority scheme that a priority is **used by**. You can click it to see the list of schemes.
4. Here you can create a new priority.

Creating priorities

To create a priority:

1. From the Priorities page, select **Add priority**.
2. Enter the name and description for your priority. The name will appear in the drop-down field when a user creates or edits an issue.
3. Choose an icon to represent this priority.
4. Specify a color to represent this priority. You can either type the HTML color code, or click the box at the right of the field to select from a color chart.
5. Select **Add** to create a priority. It will be added to the default priority scheme that contains all priorities. You can later add it to another scheme if you wish.

Associating priorities with projects

To choose priorities for a project, you need to add them to a priority scheme, and then associate this scheme with a project. Until you do that, all projects use the default priority scheme. Treat priority schemes like mappings that allow you to choose a set of priorities and the projects that will use them. Read more about this here: [Associating priorities with projects](#).

Other actions

Action	Description
Translating priorities	You can translate priorities into different languages. See Translating resolutions, priorities, statuses, and issue types .
Editing priorities	To edit a priority, select Edit next to the priority you want to edit.
Deleting priorities	To delete a priority, select Delete next to the priority you want to delete. There are certain restrictions: <ul style="list-style-type: none"> • You can't delete priorities that are used by non-default priority schemes. You can see which schemes and how many of them are using a priority in the Used by column. To delete a priority, first remove it from these schemes.
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. The order on this page applies only to the default priority scheme. <p>To reorder priorities:</p> <ul style="list-style-type: none"> • 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.

Associating priorities with projects

Once you're happy with the priorities available in your JIRA instance, it's time to associate them with some projects. You can choose a different set of priorities for each project by using priority schemes.

What's a priority scheme?

A priority scheme works like a mapping that allows you to associate a subset of priorities with particular projects. You can use it to achieve the following goals:

- restrict the set of available priorities for a project,
- control the order in which priorities are displayed,
- select a default priority that is assigned to all newly created issues in a project.

A single priority scheme can be reused across multiple projects so that a group of similar projects (i.e. projects which might be used for similar purposes) can share the same priorities. It's also easier to add or remove priorities for these projects because all you need to modify is a single priority scheme.

Default priority scheme

JIRA comes with a default priority scheme that contains all priorities, and is associated with all projects until you change it. You can't edit this scheme, but you can associate it as you wish.

Managing priority schemes

To manage priority schemes and complete the actions listed below, go to



> **Issues**, and choose **Priority schemes**. You need to have the **JIRA Administrators** global permission to get there.

Name	Priorities	Projects	Actions
Default scheme	<ul style="list-style-type: none"> ↑ High ↑ Medium ↓ Low 🚨 Urgent 🔴 Important 🟡 Standard 🟢 DEFCON-4 🟢 DEFCON-5 	<ul style="list-style-type: none"> • Project A • Project B • Project C 	Associate
Office	<ul style="list-style-type: none"> 🚨 Urgent 🔴 Important 🟡 Standard 	No projects	Edit Associate
Security	<ul style="list-style-type: none"> 🟢 DEFCON-4 🟢 DEFCON-5 	<ul style="list-style-type: none"> • Project D • Project E 	Edit Associate ...

1. This is where you can find all **priority schemes**.
2. **Priorities** used by a scheme.
3. A list of **projects** that use this set of priorities.
4. **Actions** that you can perform on a scheme: **Edit**, **Associate**, and **Delete**.

Creating priority schemes

Managing priority schemes is difficult if you don't have any. Create a priority scheme, and decide which priorities apply to it.

To create a priority scheme:

1. From the Priority schemes page, select **Add priority scheme**.
2. Enter the name and description for your scheme. Make it meaningful, so that other admins know they can reuse this scheme instead of creating a new one.
3. Add priorities by dragging and dropping them from **Available priorities** to **Selected priorities**. The order on the list matters—that's how your users will see priorities when assigning them to issues.
4. Select a default priority. It will be preselected whenever an issue is created in a project that uses this scheme.

Associating priority schemes

Next, associate your priority scheme with a project.

1. From the Priority schemes page, select **Associate** next to the priority scheme you want to associate.
2. Select the projects that you want this scheme to apply to. If a project is already using a different scheme, it will switch to this one.
3. If some issues in the projects you've selected use priorities that are not available in this scheme, you'll be asked to choose priorities that will replace them. See [Replacing obsolete priorities](#).

Other actions

Apart from creating and associating priority schemes, you can also edit and delete them.

Action	Description								
Editing	<p>To edit a priority scheme, select Edit next to the scheme you want to edit, and then change the name, default priority, or priorities selected for this scheme.</p> <p>If you removed priorities that are currently in use by some issues, you'll be asked to choose priorities that will replace them. See Replacing obsolete priorities.</p>								
Deleting	<p>To delete a priority scheme, select Delete next to the scheme you want to delete. There are certain restrictions:</p> <ul style="list-style-type: none"> • You can't edit, or delete the default priority scheme. • You can't delete priority schemes that are used by some projects. You can see how many projects are using a scheme on the Priorities scheme page (Projects). Associate these projects with a scheme (e.g. the default scheme), and proceed with deleting. 								
Ordering with JQL	<p>It's important to know how JQL works with priorities. If you use JQL to order issues by their priorities (e.g. descending), they will be ordered according to the importance of the priorities on the global list of priorities. The global list is what's displayed on the Priorities page. The order from priority schemes is not relevant for JQL. To put it in an example, let's say the priorities in your JIRA instance are arranged in the following way:</p> <table border="1"> <thead> <tr> <th>Global list (Priorities page)</th> <th>Custom priority scheme</th> </tr> </thead> <tbody> <tr> <td>1. Urgent</td> <td>1. Highest</td> </tr> <tr> <td>2. Highest</td> <td>2. Urgent</td> </tr> <tr> <td>3. Medium</td> <td>3. Medium</td> </tr> </tbody> </table> <p>After appending a JQL query with e.g. <code>ORDER BY priority DESC</code>, the Urgent issues will be at the top of the list (followed by Highest, and then Medium), even if they're in a project that uses the custom scheme. You can change the order in the default scheme on the Priorities page.</p>	Global list (Priorities page)	Custom priority scheme	1. Urgent	1. Highest	2. Highest	2. Urgent	3. Medium	3. Medium
Global list (Priorities page)	Custom priority scheme								
1. Urgent	1. Highest								
2. Highest	2. Urgent								
3. Medium	3. Medium								

Replacing obsolete priorities

When you edit an existing scheme or associate a scheme with projects that were already using a different one, you might be asked to replace obsolete priorities. That's because some issues are using priorities that are not available in the new or edited scheme. An issue can't live without a priority, so you need to replace the old with

the new. It's simply about choosing e.g. **Highest** (new scheme) to replace **Top** (current scheme). Once you select priorities, we'll update all these issues for you.

Defining resolution field values

Resolutions are the ways in which an issue can be closed. JIRA applications ship with a set of default resolutions, but you can add your own as follows.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global** permission.

Defining a new resolution

Don't create a Resolution named "Unresolved"/"None"

Any issue that has the Resolution field set is treated by JIRA applications as "resolved". The Issue Navigator displays Unresolved when no resolution is set for an issue. So adding a resolution named Unresolved/None and setting it in an issue will mean that the issue is seen as resolved. This will lead to confusion and is not recommended.

1. Choose



> **Issues.**

2. Select **Resolutions** to view the existing resolutions, along with a form for adding new resolutions:

Name	Description	Order	Operations
Done	Work has been completed on this issue.	↓	Edit · Delete · Default
Won't Do	This issue won't be actioned.	↑ · ↓	Edit · Delete · Default
Duplicate	The problem is a duplicate of an existing issue.	↑	Edit · Delete · Default

3. Complete the **Add New Resolution** form at the bottom of the page by entering the following details:

- **Name** — enter a short phrase that best describes your new resolution.
- **Description** — enter a sentence or two to describe when this resolution should be used.

i The **View Resolutions** page 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.

Defining status field values

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 applications ships with a set of [default statuses](#) that are used by the [default workflow](#). You can add your own statuses and [customize](#) the workflow. You can also re-order existing statuses, as well as change their names, descriptions and lozenges.

On this page:

- [Defining a new status](#)
- [Re-ordering statuses](#)
- [Deleting a status](#)

For all of the following procedures, you must be logged in as a user with the [JIRA Administrators global permission](#).

Defining a new status

1. Choose



> **Issues.**

2. Select **Statuses** to open the 'Statuses' page.
3. Click **Add Status** and complete the 'Add Status' form:
 - **Name** — specify a short phrase that best describes your new status.
 - **Description** — add a sentence or two to describe what workflow step this status represents.
 - **Category** — choose a category that this status will be grouped into: 'To Do' (blue), 'In Progress' (yellow) or 'Done' (green). Categories help you identify where issues are in their lifecycle, particularly in places where a large number of issues are rolled up, e.g. Version Details page, Sprint Health Gadget. The category is also used to map statuses to columns in JIRA Software, when creating a new board for an existing project.

Next steps:

Now you will need to associate your new status with a workflow 'step'. See [Working with workflows](#).

Re-ordering statuses

You may want to change the order of statuses in JIRA in line with a particular workflow or to highlight key statuses. The order of statuses is reflected on screens (or parts of the screen) in JIRA, where issues are listed or grouped by status. These include the issues summary for a project, search results (when status is one of the columns), and a number of gadgets, like the Issue Statistics gadget (where the Statistic Type is 'Status').

1. Navigate to the 'Statuses' page (described in the 'Defining a new status' section above).
2. Use the up and down arrows in the **Order** column to re-order individual statuses.

Deleting a status

You can only delete statuses that not are being used in workflows, i.e. inactive statuses.

1. Navigate to the 'Statuses' page (described in the 'Defining a new status' section above).
2. Click **Delete** for the status that you want to delete.

Translating resolutions, priorities, statuses, and issue types

Further extending JIRA as an international issue manager, it is possible to easily specify a translated name and description for all values of the following 'issue constants':

- the [issue type](#) field (for either standard and sub-task issue types)
- the [status](#) field
- the [resolution](#) field
- the [priority](#) field

This allows you to specify a translation set for each available language — providing each user with a more complete translation in their own chosen language. The translated field names and descriptions appear

throughout JIRA, e.g. in reports, gadgets and all issue views.

Translating an issue constant

Each issue constant can be configured to have a translation set for each available language in your JIRA system. If no translation has been configured for a particular language, the default issue constant name and description are displayed.

1. Log in as a user with the **JIRA Administrators** global permission.
2. Click the **Translate** link located on any of the following pages:
 - [Defining issue type field values](#) (for standard issue types - click any of the **Translate** links),
 - [Configuring sub-tasks](#) (for sub-task issue types),
 - [View statuses](#),
 - [View resolutions](#), or
 - [View priorities](#).

The relevant issue constant **Translation** page displays the translation set for the currently selected language.

3. To view/update a translation set for a specific language, select the required language from the **View Language Translations** list at the top of the page and click the **View** button to preview the translation:

Note that a translated name and description set can be specified for each type of issue constant.

4. Once all translations have been entered, select **Update**. Note that:
 - The process can be repeated for all of the issue constants — i.e. Issue Type, Status, Resolution and Priority fields.
 - The translated issue constant name and description will be displayed throughout JIRA, e.g. in reports, gadgets and all issue views.

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

Issue fields and statuses

These are the pieces that make up the issues you work on:

- [Issue fields](#)
- [Issue types](#)
- [Issue priorities](#)
- [Issue resolutions](#)

Issue fields

▼ [Expand to view issue fields...](#)

Field	Description
Project	The parent project to which the issue belongs.
Key	A unique identifier for this issue, in the example above: ANGRY-304. (The characters to the left of the hyphen represent the project to which this issue belongs.)
Summary	A brief one-line summary of the issue. For example, "Red Angry Nerd is scary."
Type	See below for a list of types.

Status	The stage the issue is currently at in its lifecycle (workflow). See below for a list of statuses.
Priority	The importance of the issue in relation to other issues. (See below for a list of priorities).
Resolution	A record of the issue's resolution , if the issue has been resolved or closed. (See below for a list of resolutions).
Affects Version(s) <i>(if applicable)</i>	Project version(s) for which the issue is (or was) manifesting.
Fix Version(s) <i>(if applicable)</i>	Project version(s) in which the issue was (or will be) fixed.
Component(s) <i>(if applicable)</i>	Project component(s) to which this issue relates.
Labels <i>(if applicable)</i>	Labels to which this issue relates.
Environment <i>(if applicable)</i>	The hardware or software environment to which the issue relates.
Description	A detailed description of the issue.
Links	A list of links to related issues. (Strikethrough text, like this , indicates that an issue has been resolved .)
Assignee	The person to whom the issue is currently assigned. Note that you cannot assign issues to a user group.
Reporter	The person who entered the issue into the system.
Votes	The number shown indicates how many votes this issue has.
Watchers	number shown indicates how many people are watching this issue.
Due <i>(if applicable)</i>	The date by which this issue is scheduled to be completed.
Created	The time and date on which this issue was entered into JIRA.
Updated	The time and date on which this issue was last edited.
Resolved	The time and date on which this issue was resolved .
Estimate	The Original Estimate of the total amount of time required to resolve the issue, as estimated when the issue was created.
Remaining	The Remaining Estimate , i.e. the current estimate of the remaining amount of time required to resolve the issue.
Logged	The sum of the Time Spent from each of the individual work logs for this issue.
Development *	If you use Bitbucket to manage your code repositories, you can create code branches in your code development tools directly from JIRA issues. See Integrating with development tools for details.
Agile *	Lets you view your issue on your Scrum or Kanban board.

Service Desk **	Lets you view request participants and view the equivalent request in the customer portal
---------------------------	---

* Only available in JIRA Software

projects, and only available to JIRA Software users

** Only available in JIRA Service Desk projects, and only available to JIRA Service Desk users

Issue types

Your default issue types depend on what JIRA application you have installed. We've listed all the default issue types for each application:

▼ [Expand to view JIRA Core issue types...](#)

Type	Description
Task	A task represents work that needs to be done.
Sub-task	A sub-task is a piece of work that is required for a task.

▼ [Expand to view JIRA Software issue types...](#)

Type	Description
Task	A task represents work that needs to be done.
Sub-task	A sub-task is a piece of work that is required for a task.
Story	A user story is the smallest unit of work that needs to be done.
Bug	A bug is a problem which impairs or prevents the functions of a product.
Epic	A big user story that needs to be broken down.

▼ [Expand to view JIRA Service Desk issue types...](#)

Type	Description
IT Help	Requesting help for IT related problems.
Purchase	Requesting hardware or software.
Change	Requesting a change in current IT profile.
Fault	Reporting a fault.
Access	Requesting additional access.

Issue priorities

An issue's priority indicates its relative importance. The default priorities are listed below; note that both the priorities and their meanings can be customized by your administrator to suit your organization.

▼ [Expand to view issue priorities...](#)

Priority	Description
Highest	Highest priority. Indicates that this issue takes precedence over all others.
High	Indicates that this issue is causing a problem and requires urgent attention.
Medium	Indicates that this issue has a significant impact.

Low	Indicates that this issue has a relatively minor impact.
Lowest	Lowest priority.

Issue resolutions

An issue can be completed, or resolved, in many ways. An issue resolution is usually set when the status is changed. The default resolutions are listed below; note that your administrator may have customized these to suit your organization.

▼ [Expand to view JIRA Core issue resolutions...](#)

Resolution	Description
Done	The work is completed.
Won't do	The work will not be done.
Duplicate	This work is being tracked elsewhere.

▼ [Expand to view JIRA Software issue resolutions...](#)

Resolution	Description
Done	The work is completed.
Won't do	The work will not be done.
Duplicate	This work is being tracked elsewhere.
Cannot reproduce	The issue cannot be reproduced.

▼ [Expand to view JIRA Service Desk issue resolutions...](#)

Resolution	Description
Done	The work is completed.
Won't do	The work will not be done.
Duplicate	This work is being tracked elsewhere.

Note that once an issue has been resolved (that is, the issue's Resolution field is filled in), textual references to that issue will show the key in strikethrough text.

Configuring issue-level security

Issue security levels are created within issue security schemes and let you control which user or group of users can view an issue. When an issue security scheme is associated with a project, its security levels can be applied to issues in that project. Sub-tasks will also inherit the security level of their parent issue.

Note, if issue security levels are available but aren't set, the project permissions will then be applied.

On this page:

- Before you begin
- Creating an issue security scheme
- Assigning an issue security scheme to a project
- Deleting an issue security scheme
- Editing an issue security scheme
- Copying an issue security scheme

Before you begin

- Log in as a user with the JIRA Administrators [global permission](#) to configure issue-level security.
- Make sure all users who want to use issue-level security have the project-specific 'Set Issue Security' permission.

Creating an issue security scheme

1. Choose



> **Issues.**

2. Select **Issue Security Schemes** to open the Issue Security Schemes page.
3. Click **Add Issue Security Scheme**.
4. Fill in the requested details and click **Add**.

Adding a security level to an issue security scheme

1. Choose



> **Issues.**

2. Select **Issue Security Schemes** to open the Issue Security Schemes page.
3. Click the scheme name, or the **Security Levels** link in the Actions column, to open the Edit Issue Security Levels page.
4. Fill in the requested details and then click **Add Security Level**.

Setting the default security level for an issue security scheme

You now have the power to select the default security level that will be applied to issues assigned to each security scheme. Some things to keep in mind when setting a default security level:

- If the reporter of an issue does not have the 'Set Issue Security' permission, the issue will be set to the default security level.
- If an issue security scheme doesn't have a default security level, issue security levels will be set to 'None' (anyone can see the issues).

1. Choose



> **Issues**.

2. Select **Issue Security Schemes** to open the Issue Security Schemes page.
3. Click the scheme name, or the **Security Levels** link in the Actions column, to open the Edit Issue Security Levels page.
 - a. To set the default security level, locate the appropriate **Security Level** and click **Default** in the Actions column.
 - b. To remove the default security level, click **Change default security level to "None"** link (near the top of the page).

Adding members to a security level

1. Choose



> **Issues**.

2. Select **Issue Security Schemes** to open the Issue Security Schemes page.
3. Click the name of any scheme or the link **Security Levels** to open the **Edit Issue Security Levels** page.
4. Locate the appropriate security level and click its **Add** link (in the **Actions** column), which opens the **Add User/Group/Project Role to Issue Security Level** page.
5. Select the appropriate user, group or project role, then click the **Add** button.

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 or (multi-)group picker [custom field](#).

6. Repeat steps 4 and 5 until all appropriate users, groups, or project roles have been added to the security level.

Assigning an issue security scheme to a project

1. Choose



> **Projects**, and select the relevant project.

2. Select the name of the project of interest. The **Project Summary** page is displayed.
3. In the **Permissions** section of the **Project Summary** page, click the link corresponding to the **Issues** label to open the **Associate Issue Security Scheme to Project** page. This will either be the name of the project's current issue security scheme, or the word **None**.
4. Select the issue security scheme that you want to associate with this project.
5. If there are no previously secured issues (or if the project did not previously have an issue security scheme), skip the next step.
6. 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.
7. Click the **'Associate'** button to associate the project with the issue security scheme.

If the **Security Level** field is not displayed on the issue's screen after configuring the Issue-Level Security, use the Where is My Field? tool to see why it is not being displayed.

If the **Security Level** field has been hidden on purpose, please see the limitations of doing so in [Hiding or showing a field](#).

Deleting an issue security scheme

It's important to understand that you can't delete a issue security scheme if it is associated with a project. You must first remove any associations between the issue security scheme and projects in your JIRA installation — please refer to [Assigning an Issue Security Scheme](#).

1. Choose  **> Issues.**
2. Select **Issue Security Schemes** to open the Issue Security Schemes page, which lists all the issue security schemes currently available in your JIRA installation.
3. Click the **Delete** link (in the **Actions** column) for the scheme that you want to delete.
4. On the confirmation page, click **Delete** to confirm the deletion. Otherwise, click **Cancel**.

Editing an issue security scheme

You can edit the name and description of an issue security scheme. You can also edit the Default Security Level when editing an issue, and the security level will be applied in the same manner as described in [Setting the default security level for an issue security scheme](#).

1. Choose  **> Issues.**
2. Select **Issue Security Schemes** to open the Issue Security Schemes page, which lists all the issue security schemes currently available in your JIRA installation.
3. Click the **Edit** link (in the **Actions** column) for the scheme that you want to edit.
4. Make your edits, and then click **Update** to confirm the edits. Otherwise, click **Cancel**.

Copying an issue security scheme

1. Choose  **> Issues.**
2. Select **Issue Security Schemes** to open the Issue Security Schemes page, which lists all the issue security schemes currently available in your JIRA installation.
3. Click the **Copy** link (in the **Actions** column) for the scheme that you want to copy. A new scheme will be created with the same security levels and the same users/groups/project roles assigned to them. Your new scheme will be called '**Copy of ...**'. You can edit your new scheme to give it a different name if you wish.

Configuring permissions

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

- permissions within JIRA applications themselves
- security in the external environment

Configuring permissions within JIRA applications

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

There are five types of security levels within JIRA applications:

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

On this page:

- [Configuring permissions within JIRA applications](#)
- [Configuring security in the external environment](#)

In this section:

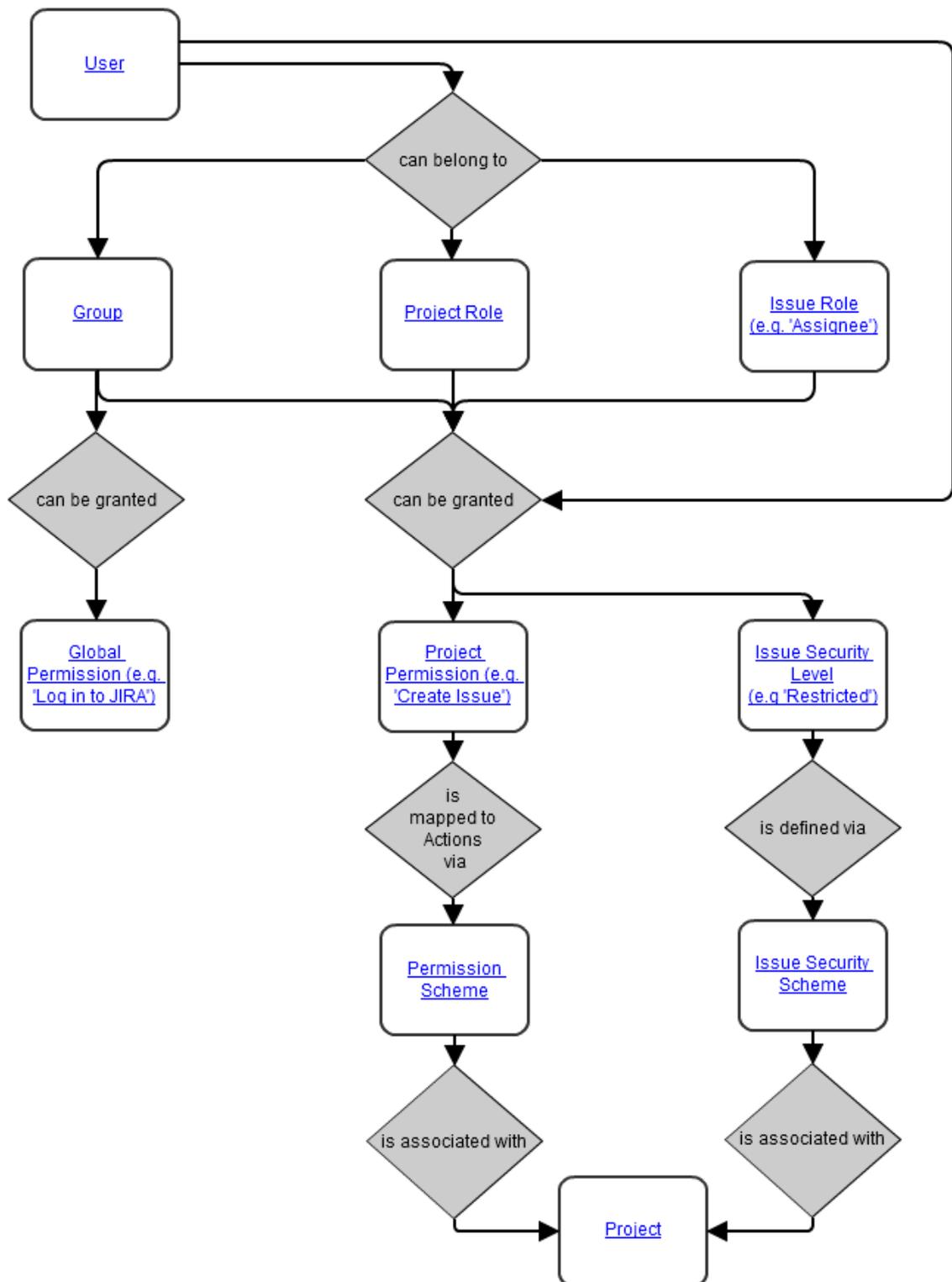
the project's permissions.

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. Does not restrict visibility of progress bar on issue time tracking.

- Managing global permissions
- Managing project permissions
 - Customizing JIRA Service Desk permissions
 - Resolving JIRA Service Desk permission errors
 - Using Manage Sprints permission for advanced cases

- Managing project roles
 - Managing project role membership
- Allowing anonymous access to your project

Diagram: People and permissions



Configuring security in the external environment

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

- File system — you should restrict access to the following directories (but note that the user which your instance is running as will require full access to these directories):
 - [Index directory](#)
 - [Attachments directory](#)
- Database:
 - If you are using an [external database](#) as recommended for production systems (i.e. you are not

using JIRA's internal/bundled H2 database), you should restrict access to the database that your JIRA instance uses.

- If you are using JIRA's internal/bundled H2 database, 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.)
- SSL — if you are running your JIRA instance over the Internet, you may want to consider using [SSL](#).

Managing global permissions

Global permissions are system-wide and are granted to groups of users. You can refer to [project permissions](#) to manage permissions that apply to individual projects.

For all of the following procedures, you must be logged in as a user with the JIRA Administrators or JIRA System Administrators global permission.

On this page:

- [Granting global permissions](#)
- [Removing global permissions](#)
- [About JIRA System Administrators and JIRA Administrators](#)
- [Separating JIRA System Administrators from JIRA Administrators in default JIRA installations](#)
- [Troubleshooting permissions with the JIRA admin helper](#)

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. Note that a user with the JIRA Administrators permission will be able to log in at any time, but may have restricted functions depending on their application access.

Browse Users	Permission to view a list of all JIRA user names and group names, share issues, and @mention people on issues. Used for selecting users/groups in popup screens. Enables auto-completion of user names in most 'User Picker' menus and popups. Note that the Assign User permissions also allows a limited version of this on a per-project basis.
Create Shared Objects	Permission to share a filter or dashboard globally or with groups of users. Also used to control who can create an agile board.
Manage Group Filter Subscriptions	Permission to manage (create and delete) group filter subscriptions.
Bulk Change	Permission to execute the bulk operations within JIRA: - Bulk Edit * - Bulk Move * - Bulk Workflow Transition - Bulk Delete * (* subject to project-specific permissions .) 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 (i.e. anybody can sign up and create issues), a user with the Bulk Change global permission and the Add Comments project permission could comment on <i>all</i> 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).

Granting global permissions

1. Choose



> **System.**

2. Select **Global Permissions** to open the Global Permissions page, which lists JIRA's global permissions.

The **Add Permission** box is shown at the bottom of the list (not displayed in the screen capture above).

3. In the **Permission** drop-down list, select the global permission you wish to grant.
 4. 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**. This is **not** recommended for production systems, or systems that can be accessed from the public Internet such as Cloud.

Please Note:

 - If you have reached your user limit, you will be able to create new users but it won't have login permission.
- JIRA admin doesn't consume a license unless they've been granted specific JIRA application access. See [Licensing and application access](#).

Removing global permissions

1. Choose



> System.

2. Select **Global Permissions** to open the Global Permissions page, which lists JIRA's global permissions.
3. For each global permission in JIRA (indicated on the left of this page), groups which currently have that permission are shown on the right (under the **Users / Groups** column).
4. Locate the global permission you want to remove from a group as well as the group you want to remove that permission from (under **Users / Groups**) and click the **Delete** link next to that group.

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 [separation](#) is useful for organizations 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.

Here is a list of administration tasks that only **JIRA System Administrators** (*not JIRA Administrators*) can perform:

- View or manage tasks from the the [Systems menu](#).
- Configure JIRA's [SMTP mail server](#) for notifications (but *they can* configure [POP/IMAP mail servers](#) for the receipt of email messages that create issue comments and new issues, and fully administer [email notification schemes](#)).
- Configure a CVS source code repository (but *they can* associate a [project](#) with a configured repository).
- Configure [listeners](#).
- Configure [services](#) (except for [POP/IMAP](#) services).
- Configure [issue cloning](#).
- Change the [index](#) path (but *they can* reindex and optimize 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 XML workflows](#) into JIRA.
- Configure [attachments](#) (note that **JIRA Administrators** can set the size limits of attachments, enable thumbnails, and enable ZIP support).
- Add gadgets to the [gadget directory](#).
- Configure [user directories](#) (e.g. LDAP).
- Configure [Application Links](#) that use an authentication type other than OAuth.
- View [user sessions](#).
- 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.
- Upload and/or install an [add-on](#).

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.

Separating JIRA System Administrators from JIRA Administrators in default JIRA installations

By default, the `jira-administrators` [groups](#) has both the **JIRA Administrators** permission *and* the **JIRA System Administrators** permission. Also by default, the user account created during the [JIRA setup wizard](#) is a member of this `jira-administrators` group.

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.

Troubleshooting permissions with the JIRA admin helper

The JIRA admin helper can help you diagnose why a user can or cannot see a certain issue.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

1. Choose  **> System.**
2. Select **Permission helper.**
3. Enter the username of the user (leave blank for anonymous users), an issue key (for example, an issue that the user can/cannot see) and the permission to check.
4. Click **Submit.**

Managing project permissions

Project permissions are created within [permission schemes](#), which are then assigned to specific projects by JIRA Administrators. Project permissions are able to be granted based on:

- Individual users
- Groups
- [Project roles](#)
- Issue roles such as 'Reporter', 'Project Lead', and 'Current Assignee'
- 'Anyone' (e.g. 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 that some permissions are dependent upon others to ensure that users can perform the actions needed. For example, in order for a user to be able to resolve an issue, that user must be granted both the Transition Issue permission and the Resolve Issue permission.

On this page:

- [Permission schemes](#)
- [Creating a permission scheme](#)
- [Adding users, groups, or roles to a permission scheme](#)
- [Deleting users, groups, or roles from a permission scheme](#)
- [Associating a permission scheme with a project](#)
- [Deleting a permission scheme](#)
- [Copying a permission scheme](#)

The following table lists the different types of project permissions and the functions they secure. Note that project permissions can also be used in [workflow conditions](#).

Project permissions overview

Project permissions	Explanation
Administer projects	Permission to administer a project in JIRA. This includes the ability to edit project role membership , project components , project versions , and some project details ('Project Name', 'URL', 'Project Lead', 'Project Description').
Extended project administration	<p>Gives the project administrator the ability to edit workflows and screens under certain conditions.</p> <p>Restrictions for editing the project's workflows...</p> <ul style="list-style-type: none"> • The workflow must not be shared with any other projects, or be a system workflow. • To add a status, the status must already exist in the JIRA instance i.e. the project admin can't create new statuses or edit existing statuses. • To delete a status, the status must not be used by any of the project's issues. • The project admin can create, update or delete transitions, but they can't select or update a screen used by the transition, or edit or view a transition's properties, conditions, validators or post-functions. <p>Restrictions for editing the project's screens...</p> <ul style="list-style-type: none"> • The screen must not be a default system screen. • The screen must not be shared with any other projects, or used as a transition screen in workflows. • The project admin can add, remove and rearrange system fields. • The project admin can add, remove and rearrange existing custom fields, but they cannot create custom fields.
Browse projects	Permission to browse projects, use the Issue Navigator and view individual issues (except issues that have been restricted via issue-level security). Many other permissions are dependent on this permission , e.g. the 'Work On Issues' permission is only effective for users who also have the 'Browse Projects' permission.

Manage sprints (only available to JIRA Software users)	<p>Permission to perform the following sprint-related actions for all projects in a board:</p> <ul style="list-style-type: none"> • Creating sprints • Starting sprints • Completing sprints • Reopening sprints • Reordering future sprints • Adding sprint goals • Deleting sprints • Editing sprint information (sprint name, goal, dates) • Moving the sprint footer <p>Depending on the complexity of your board's filter query, you may need further consideration when configuring the 'Manage Sprints' permission for users. For more information on the impact of complex filters, and ways to simplify your filter query, see Using Manage Sprints permission for advanced cases.</p> <p>▼ Notes on working with sprints</p> <p>In general, sprint actions require the Manage Sprints permission. However, there are some sprint actions (e.g. adding issues to sprints, removing issues from sprints) that require the Schedule Issues and Edit Issues permissions.</p> <p>When adding an issue to a sprint:</p> <ul style="list-style-type: none"> • Sub-tasks cannot be moved independently of their parents. • An issue can only be assigned to one active sprint or future sprint. This means you can't add an issue to both an active sprint and a future sprint at the same time. • You can <i>add any issue to any active or future sprint</i>, even if the issue doesn't match the filter query of the board where the sprint was created. When you do this: <ul style="list-style-type: none"> • the issue is assigned to the sprint, but will not be visible on boards where the filter query excludes it. • any sprint actions (e.g. start sprint, close sprint) that span multiple boards will also <i>affect the sprint in all boards where the sprint is visible</i>. • if the issue doesn't match the filter query of any agile board, the issue will be linked to the sprint but not appear in any board. • A sprint appears in any board — a single board or multiple boards — as long as the issues assigned to the sprint match the filter query of the board or boards. This also applies to completed sprints. <p>See Planning sprints for more information.</p>
View development tools (only available to JIRA Software users)	Permission to view the Development panel , which provides you with just enough information to evaluate the status of an issue's development, at a glance.
View (read-only) workflow	Permission to view the project's 'read-only' workflow when viewing an issue. This permission provides the 'View Workflow' link against the 'Status' field of the 'View Issue' page.
Issue permissions	Explanation
Assign issues	Permission to assign issues to users. Also allows autocompletion of users in the Assign Issue dropdown. (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).

Close issues	Permission to close issues based on the workflow conditions. (This permission is useful where, for example, developers resolve issues and testers close them). Requires the Transition issue and Resolve issue transitions. Also see the Resolve Issues permission.
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).
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.
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).
Link issues	Permission to link issues together. (Only relevant if Issue Linking is enabled).
Modify reporter	Permission to modify the 'Reporter' of an issue. This allows a user to create issues 'on behalf of' someone else. This permission should generally only be granted to administrators.
Move issues	Permission to move issues from one project to another, or 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.
Resolve issues	Permission to resolve and reopen issues based on the workflow condition. This also includes the ability to set the 'Fix For version' field for issues. Requires the Transition issues permission. Also see the Close Issues permission.
Schedule issues	Permission to schedule an issue — that is, to edit the 'Due Date' of an issue. In older versions of JIRA this also controlled the permission to view the 'Due Date' of an issue.
Set issues security	Permission to set the security level on an issue to control who can access the issue. Only relevant if issue security has been enabled .
Transition issues	Permission to transition (change) the status of an issue.
Voters & watchers permissions	Explanation
Manage watcher list	Permission to manage (i.e. view/add/remove users to/from) the watcher list of an issue.
View voters and watchers	Permission to view the voter list and watcher list of an issue. Also see the Manage Watcher List permission.
Comments permissions	Explanation
Add comments	Permission to add comments to issues. Note that this does not include the ability to edit or delete comments.

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.
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.
Attachments permissions	Explanation
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.
Time-tracking Permissions	Explanation
Work on issues	Permission to log work against an issue, i.e. create a worklog entry. (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). Also see the Work On Issues permission.
Delete own worklogs	Permission to delete worklog entries that were added by the user. (Only relevant if time tracking is enabled). Also see the Work On Issues permission.
Edit all worklogs	Permission to edit any worklog entries, regardless of who added them. (Only relevant if time tracking is enabled). Also see the Work On Issues permission.
Edit own worklogs	Permission to edit worklog entries that were added by the user. (Only relevant if time tracking is enabled). Also see the Work On Issues permission.

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 organizations, multiple projects have the same needs regarding access rights. (For example, only the specified project team may be authorized 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.

Creating a permission scheme

1. Choose



> **Issues.**

2. Select **Permission Schemes** to open the Permission Schemes page, which displays a list of all permission schemes in your JIRA system and the projects that use each scheme.
3. Click the '**Add Permission Scheme**' link.
4. In the 'Add Permission Scheme' form, enter a name for the scheme, and a short description of the scheme. Select **Add**.
5. You will return to the 'Permission Schemes' page which now contains the newly added scheme.

Adding users, groups, or roles to a permission scheme

1. Choose



> **Issues.**

2. Select **Permission Schemes** to open the Permission Schemes page, which displays a list of all permission schemes in your JIRA system and the projects that use each scheme.
3. Locate the permission scheme you would like to update, and select **Permissions** in the Operations column to view the scheme.
4. Select the '**Edit**' link for the permission you wish to add to, this displays the 'Grant permission' dialog.
5. Select who to add the selected permission to, and click the '**Grant**' button. The users/groups/roles will now be added to the selected permission. 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 minimize the number of permission schemes in your system.
6. Repeat the last 2 steps until all required users/groups/roles have been added to the permissions.

Deleting users, groups, or roles from a permission scheme

1. Choose



> **Issues.**

2. Select **Permission Schemes** to open the Permission Schemes page, which displays a list of all permission schemes in your JIRA system and the projects that use each scheme.
3. Locate the permission scheme of interest and click its name to show the list of 'Project Permissions' ([above](#)).
4. Click the **Remove** link for the permission you wish to remove the users, groups, or roles from.
5. Select the users, groups, or roles you wish to remove, and click the **Remove** button.

Associating a permission scheme with a project

1. Choose



> **Projects**, and select the relevant project.

2. Select the project of interest to open the **Project Summary** administration page for that project. See [Defining a project](#) for more information.
3. On the lower right, in the **Permissions** section, click the name of the current scheme (e.g. 'Default Permission Scheme') to display the details of the project's current permission scheme.
4. Click the '**Actions**' dropdown menu and choose '**Use a different scheme**'.
5. On the 'Associate Permission Scheme to Project' page, which lists all available permission schemes, select the permission scheme you want to associate with the project.
6. Click the '**Associate**' button to associate the project with the permission scheme.

Deleting a permission scheme

1. Choose



> **Issues.**

2. Select **Permission Schemes** to open the Permission Schemes page, which displays a list of all permission schemes in your JIRA system and the projects that use each scheme.

3. Click the **Delete** link (in the **Operations** column) for the scheme that you want to delete.
4. A confirmation screen will appear. To delete click **Delete** otherwise click **Cancel**.
5. 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.)

Copying a permission scheme

1. Choose  **> Issues**.
2. Select **Permission Schemes** to open the Permission Schemes page, which displays a list of all permission schemes in your JIRA system and the projects that use each scheme.
3. Click the **Copy** link (in the **Operations** column) for the scheme that you want to copy.
4. A new scheme will be created with the same permissions and the same users/groups/roles assigned to them.

Customizing JIRA Service Desk permissions

If you want to customize the permission scheme for your service desk, make sure that you grant permissions to users by granting them:

- to the **Administrators** role for administrators
- to the **Service Desk Team** role for agents
- to the **Service Desk Customer - Portal Access** security type for customers.

If you grant permissions to groups or individual users instead of the roles and security type, some functionality in your service desk might be disabled.

Mandatory permissions by project roles

If you choose to use custom permission schemes, the permissions in the following table are mandatory for the project roles in the typical service desk context. If you configure the permissions for the roles differently than shown in the table and run into problems, you can find the explanation of the problems and how you can fix them on [Resolving JIRA Service Desk permission errors](#).

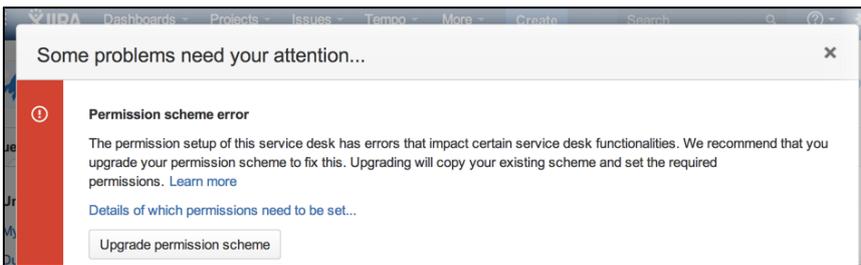
Project role	Mandatory permissions
Administrators	<p>This project role must have the Administer Projects project permission in order to set up and administer a service desk. This permission allows users to manage service desk functionality like creating new request types, setting up new queues, creating SLAs, and generating reports.</p> <p>This project role also must have all the permissions granted to the other users of the service desk in order to see all the functionality they'll be using.</p>
Service Desk Team	<ul style="list-style-type: none"> • Create Issues (This permission gives users the ability to create issues in a Customer Portal.) • Browse Projects (This permission gives users read-only access to the Reports, People, and SLA tabs in a service desk project, as well as access to the project's Customer Portal. Users can also see the Queues tab and work on issues from within the queues.) • Edit Issues • Schedule Issues • Add Comments • Create Attachments

Service Desk Customers	<p>The permissions for customers must be granted to the Service Desk Customer - Portal Access security type, not the Service Desk Customers role. The Portal Access security type lets customers access the customer portal, but not JIRA. The security type reads the role to determine who are customers.</p> <ul style="list-style-type: none"> • Create Issues (Customers can create requests and view requests they have submitted via the customer portal) • Browse Projects (Customers can access the project in the customer portal) • Add Comments (Customers can comment on their requests.) • Create Attachments (Customers can add attachments to their requests) • Assign Issue (This permission is mandatory for the Assignee field to work. The Assignee field is an optional hidden field and it automatically channels issues to certain team members.) <p>In addition, if the service desk project uses an issue security scheme, make sure that it is configured so that service desk users can view issues. Otherwise, customers might be able to create issues but not view them after they've been created. See Configuring issue-level security.</p>
-------------------------------	--

Resolving JIRA Service Desk permission errors

When you create a service desk project, it uses a permission scheme called *JIRA Service Desk Permission Scheme for %ProjectKey%*. If you change this permission scheme, then JIRA Service Desk might display a permission error similar to the following:

- What are permission errors?
- How do I fix permission errors?
- Critical permission errors

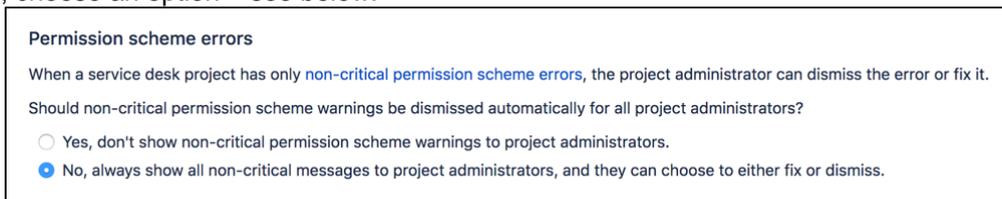


What are permission errors?

JIRA Service Desk considers the differences between your permission scheme and the standard JIRA Service Desk permission scheme as errors in the following two categories:

- **Critical errors (red):** Break core service desk functionality, such as adding agents or allowing customers to log in to the portal. JIRA Service Desk displays a warning until you fix major errors. For a complete list of major errors, see [this table](#).
- **Non-critical errors (yellow):** Differ from the standard permission scheme, but don't impact how JIRA Service Desk works. JIRA administrators can choose whether non-critical permission scheme warnings are dismissed automatically, or if they are always shown to project administrators, so that they can either fix or dismiss them.

- Go to **Administration** () > **Applications > JIRA Service Desk Configuration**, and under **Permission scheme errors**, choose an option – see below:



How do I fix permission errors?

To fix permission errors, you can change the permission scheme yourself, or click the **Fix permissions** button in the error message to have JIRA Service Desk fix the errors for you. When you click **Fix permissions**, JIRA Service Desk corrects the critical and non-critical errors in your permission scheme by doing the

following:

1. Disassociates your permission scheme with the service desk project.
2. Creates a copy of your permission scheme called *%Your permission scheme%1* and associates the scheme with the project.
3. Fixes the errors by:
 - a. Granting standard permissions to the **Administrators** and **Service Desk Team** roles, and the **Service Desk Customer - Portal Access** security type.
 - b. Removing the **Service Desk Customers** role from all the permissions assigned.

The following table describes how JIRA Service Desk might fix a permission scheme:

Custom permission scheme JIRA Service Desk Permission Scheme for Project OA	Fixed permission scheme JIRA Service Desk Permission Scheme for Project OA 1
<p>The following permissions are set up differently from the standard permission scheme:</p> <ul style="list-style-type: none"> • User John Smith has the Browse Projects permission. This is a minor error. • The Service Desk Customers role has the Create Issues permission. This is a major error. • The Service Desk Customer - Portal Access security type does not have the Create Issues permission. This is the major error. 	<p>After you click Fix permissions, the 'JIRA Service Desk Permission scheme for Project OA' permission scheme is dissociated with the project, and a new permission scheme called 'JIRA Service Desk Permission scheme for Project OA 1' will be applied to your service desk.</p> <ul style="list-style-type: none"> • User John Smith will still have the Browse Projects permission. • The Service Desk Customers role is removed from the Create Issues permission. • The Service Desk Customer - Portal Access security type will be granted the Create Issues permission.

Critical permission errors

Critical permission errors break core service desk functionality. JIRA Service Desk displays a warning until you fix them.

Error	Explanation
<p>The Administrators role does not have the following required permissions:</p> <ul style="list-style-type: none"> • Browse Projects • Administer Projects • Edit Issues 	<ul style="list-style-type: none"> • No Browse Projects permission = Administrators cannot access the service desk. • No Administer Projects permission = Administrators cannot modify settings of the service desk. • No Edit Issues permission = Administrators cannot edit issues.
<p>The Service Desk Customer - Portal Access security type does not have the following required permissions:</p> <ul style="list-style-type: none"> • Browse Projects • Create Issues • Add Comments 	<ul style="list-style-type: none"> • No Browse Projects permission = Customers cannot access the Customer Portal of the service desk, that is they cannot log in. • No Create Issues permission = Customers cannot create requests on the Customer Portal. • No Add Comments permission = Customers cannot add comments to their requests.

<p>The Service Desk Customers role is granted any permission directly.</p>	<p>Granting permissions to this role gives customers access to JIRA functions. Customers should only have access to a Customer Portal and permissions should be granted to the Service Desk Customer - Portal Access security type.</p> <p>As a result, administrators will not be able to add any customers to the service desk. Open service desks will become restricted. Public signup will be disabled.</p>
<p>The Service Desk Team role does not have the following required permissions:</p> <ul style="list-style-type: none"> • Browse Projects • Edit Issues 	<ul style="list-style-type: none"> • No Browse Projects permission = Agents cannot see the service desk. • No Edit Issues permission = Agents cannot edit issues.
<p>The Service Desk Team role is granted the Administer Projects permission.</p>	<p>Granting the Administer Projects permission to your agents means that all agents become administrators for your service desk.</p> <p>This is a severe security issue. JIRA Service Desk will disable the functionality of agent management. As a result, administrators will not be able to add any agents.</p>
<p>The Anyone group is granted the Browse Projects permission.</p>	<p>Granting the Browse Project permission to the Anyone group means that anyone can access the project and view all the issues in it.</p>

Using Manage Sprints permission for advanced cases

The 'Manage Sprints' permission (only available to JIRA Software users) is a [project permission](#) that allows users to perform the following sprint-related actions:

- Creating sprints
- Starting sprints
- Completing sprints
- Reopening sprints
- Reordering future sprints
- Deleting future sprints
- Editing sprint information (sprint name and dates)
- Moving the sprint footer

Caveats of the 'Manage Sprints' permission

With this permission, the board's filter query determines the projects that users need to have permission on. Also, permissions are now checked against the filter query of the board from which the sprint originates, not just against the issues within the sprint.

When boards have complex filter queries

A filter query is considered complex when JIRA Software can't determine which projects will be returned by the query. When this happens, JIRA Software will require users to have the 'Manage Sprints' permission for all projects in the instance — essentially, you'll need to manually set users to have this permission for all projects.

To handle this better, consider using [JIRA project roles](#) for the 'Manage Sprints' permission. While project roles are defined at the instance level, they are applied at the project level. Thus, project level permissions can be given to [members of a project role](#), as well as groups, individual users, or through other means of designating a user. In essence, project roles enable you to associate users with particular functions for specific projects.

For example, you can consider doing the following:

1. Create a new [project role](#) called **Sprint Manager**.
2. In the corresponding permission scheme, assign the 'Manage Sprints' permission to the **Sprint Manager** project role.

3. [Associate](#) the permission scheme with the corresponding projects in your instance.
4. [Add](#) the appropriate users to the **Sprint Manager** project role.

Completing these steps will make sure that the appropriate users have the Sprint Manager project role in the corresponding projects — and since the 'Manage Sprints' permission is assigned to the Sprint Manager project role, then these users can perform sprint-related actions.

The following table lists some examples of complex filter queries, and suggestions on simplifying such queries.

Complex filter query	Why the query is complex	How to simplify the query
assignee = someone	These queries return global context results because the results could potentially come from any project in the instance.	Add the project clause into the queries. This will reduce the number of projects JIRA Software will check permissions on.
project = TIS OR issuetype = Bug		
project = TIS OR (issuetype = Bug AND assignee = someone)		
(project = TIS OR assignee = A) AND (project = PMO OR assignee = B)	JIRA Software will evaluate this query as: (project = TIS AND assignee = B) OR (project = TIS AND project = PMO) OR (assignee = A AND project = PMO) OR (assignee = A AND assignee = B) The red parts of the query won't return any results, which makes the query complex. Since the query returns undefined results, the 'Manage Sprints' permission will then be required for all projects in the instance.	Rewrite the query as: (project = TIS AND assignee = B) OR (project = PMO AND assignee = B) With this query, users will be required to have the 'Manage Sprints' permission on <i>only two projects</i> .

In summary, we recommend that queries contain OR clauses in which AND clauses can be sub-clauses, and not the other way around.

Simply put, make sure:

- your OR clauses are outside the brackets, and
- your AND clauses sub-clauses are inside the brackets.

Recommended query format: <clause> OR (<clause> AND <clause>) OR <clause> OR (<clause> AND <clause>)

Complex query format: <clause> AND (<clause> OR <clause>) AND (<clause> OR <clause>)

When boards contain sprints from other boards

With the 'Manage Sprints' permission, permissions are now checked against the filter query of the **origin board** — the board from which the sprint is created — not just against the issues within the sprint.

Depending on the filter query being used, your board might display sprints from other boards. For example, you have the **TIS board** and it's displaying Sprint 3, which was created in another **board** — the **PMO board**. In this case, the **PMO board** is the **origin board** of Sprint 3.

If you're in the **TIS board** and you're closing Sprint 3, the following items are checked:

1. JIRA Software checks if you have the 'Manage Sprints' permission for the projects in the **origin PMO board**.
2. If you have permissions, the sprint will be closed. If Sprint 3 has any incomplete issues, JIRA Software will offer destination options, allowing you to move the incomplete issue to either the Backlog or a future sprint of the **TIS board**, e.g. Sprint 4.
3. If you choose to move the incomplete issue to Sprint 4, the issue is moved to Sprint 4 of the **TIS board**.
4. If Sprint 4 also exists in the **PMO board**, then the incomplete issue will appear in Sprint 4 of the **PMO board**.
However, if Sprint 4 doesn't exist in the **PMO board**, the incomplete issue will be moved to the Backlog of the **PMO board**.

Managing project roles

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

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

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](#)
- [dashboards](#)

On this page:

- [Using project roles](#)
- [Default project roles](#)
- [Viewing project roles](#)
- [Adding a project role](#)
- [Deleting a project role](#)
- [Editing a project role](#)
- [Assigning members to a project role](#)
- [Specifying 'default members' for a project role](#)

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. Every project has a project lead and every project component has a component lead. These individual roles can be used in schemes, issues and workflows, just like project roles. You assign project/component leads when [defining projects](#) or [managing components](#) respectively.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

Using project roles

Project roles enable you to associate users with particular functions. For example, if your organization

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.

Default project roles

When you install JIRA applications, the Administrators role is automatically created, along with project roles specific to each application. You can [create](#), [edit](#), and [delete](#) project roles according to your organization's requirements.

Viewing project roles

1. Choose  **> System**.
2. Select **Project roles** to open the Project Role Browser page.
3. You will then see the Project Role Browser, which contains a list of all the project roles in your JIRA system.
4. To see where a project role is used, click the **View Usage** link. This will display a list of the project role's associated [permission schemes](#), [email notification schemes](#), [issue security levels](#), and [workflow conditions](#).
5. Click any of the **View** links on the 'View Usage for Project Role' screen to see which users/groups are associated with a project role for a particular project.

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.

1. Click on Manage Default Members in the **Operations** column for the newly created Project Role.
2. Click **Edit** under Default Users.
3. Select the User Picker icon to the right of the *Add user(s) to project role* field.
4. Click the Select button at the bottom of this dialog when you are finished adding users and then click the Add button. You now see a list of users on the right that are now included in this Project Role.

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](#)).

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 permission.

If a project role has been used to specify who can view a comment, deleting the project role will mean that no one can see that comment any more.

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.

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.

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

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

Managing project role membership

A JIRA application 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 applications, project roles have specific members for each project. Users may play different roles in different projects.

This page contains instructions for managing membership of *existing project roles*. For information on creating and using project roles, see [Managing project roles](#).

On this page:

- [Viewing project role members](#)
- [Assigning a user or group to a project role](#)
- [Removing a user or group from a project role](#)

For all of the following procedures, you must be logged in to JIRA as a [project administrator](#).

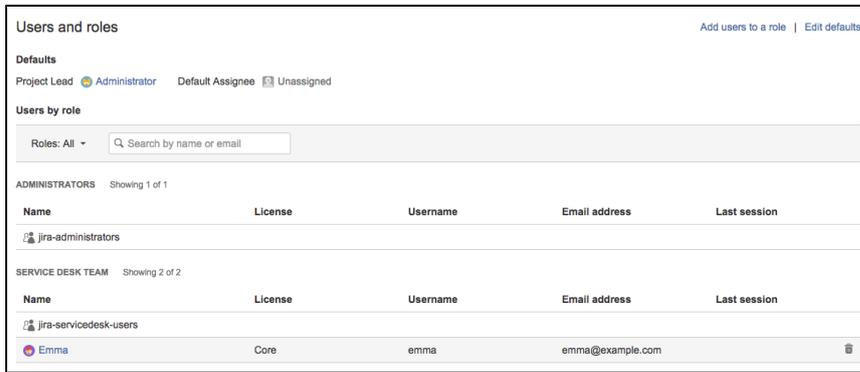
Viewing project role members

1. Choose



> **Projects**, and select the relevant project.

2. Choose **Users and roles** in the left menu to view and manage project role membership:



Assigning a user or group to a project role

1. Open the **Users and roles** page as described [above](#).
2. Select **Add users to a role** from the top right corner.
3. Search for the user or group you wish to add, and select the project role you wish to add them to. Note that the **Browse Users global permission** is required to search for existing users or groups at this step. If you do not have this permission, you will need to specify the exact name or email address.
4. Select **Add**.

Removing a user or group from a project role

1. Open the **Users and roles** page, as described [above](#).
2. Hover over the user or group you wish to remove and select



Note:

- Because group membership can only be edited by users with the **JIRA Administrator global permission**, project administrators may therefore prefer to assign users, rather than groups, to their project roles.
- A project role does not need to have any users or groups assigned to it, although project administrators should be careful with this. Depending on how a project role is used (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.

Allowing anonymous access to your project

JIRA applications can be configured to allow users to create issues without having logged in.

To do this, you need to:

1. Allow users to browse and search issues in the project without logging in.
 - a. Add the Anyone group to the **Browse Project** permission in the permission scheme for the project.
2. Allow users to create issues in that project without logging in.
 - a. Add the Anyone group to the **Create Issue** permission in the permission scheme for the project.
 - b. Set the **Reporter** in the project's field configuration scheme to optional.

Any issue created by a user who is not logged in will display 'Anonymous' for the reporter of the issue. You should also ensure that the anonymous user is able to complete and submit all required fields. For example, if you make the Due Date field required, the anonymous user will also need to have the Schedule Issue permission.

Note that anonymous users will have access equivalent to JIRA Core users. In other words, they can view issues and work in any type of project, but they won't see application-specific features, e.g. agile boards, which are JIRA Software-specific features. See [JIRA applications and project types overview](#) for more information.

Managing components

Components are sub-sections of a project. They are used to group issues

within a project into smaller parts. You can set a [default assignee](#) for a component. This will override the project's default assignee, for issues in that component.

You need to have the project-specific **Administer Projects** [permission](#) or the **JIRA Administrator** [global permission](#) to be able to:

- Add — create a new component against which issues can be aligned
- Select a default assignee — choose who is automatically assigned issues with that component assigned
- Edit — change a component's details
- Delete — remove a component

Once a component has been created for a project, the 'Component' field becomes available for your issues. If you cannot see this field on your issue, your project may not have any components yet, or the field is hidden from view.

On this page:

- [Managing a project's components](#)
- [Adding a new component](#)
- [Editing a component's details](#)
- [Searching for a component](#)
- [Deleting a component](#)

Managing a project's components

1. Choose



> **Projects**, and select the relevant project.

2. Choose **Components** in the project sidebar. The **Components** page is displayed, showing a list of components and each component's details. From here you can manage the project's components as described below.

Adding a new component

1. The Add Component form is located at the top of the 'Components' screen.
2. Enter the **Name** for the component. Optionally, enter a **Description**, and select a **Component Lead** and **Default Assignee** (see [options](#) below).
3. Click **Add**.

Selecting a Default Assignee

You can optionally set a Default Assignee for a component. This will override the project's default assignee, for issues in that component.

i If an issue has multiple components, and the default assignees of components clash, the assignee will be set to the default assignee of the component that is first alphabetically.

Default assignee option	Description	Notes
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.	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.	If the component 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.	This option will only be available if "Allow unassigned issues" is enabled in the General Configuration .

Editing a component's details

1. On the 'Components' screen, open the menu in the **Actions** column for the component you want to edit, and select **Edit**.
2. Edit the component's **Name**, **Description**, **Lead**, and **Default Assignee** in the **Edit component** dialog.
3. Click the **Save** button to save your changes.

Searching for a component

If you need to find a component in a long list, it's easiest to search for it. Start typing text into the search box that you know the component contains, and your list will automatically be filtered for you.

Deleting a component

1. On the 'Components' screen, open the menu in the **Actions** column for the component you want to delete, and select **Delete**.
2. You will be prompted to associate any issues assigned to this component with another component if you wish.
3. Click the **Delete** button to delete the component.

Managing versions

Versions are points-in-time for a project. They help you schedule and organize your releases. Once a version is created and issues are assigned to it, you can use several reports, e.g. the Change Log report, when managing the version. The Change Log report, in particular, gives you a review of the released version, and is 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 makes some changes in some reports (e.g. Change Log report) and some issue fields' drop-downs. If you have integrated JIRA applications with [Bamboo](#), you can also trigger builds when releasing a version.
- Rescheduled — re-arrange the order of versions.
- Archived — hide an old version from the Change Log reports, and in the JIRA User Interface.
- Merged — combine multiple versions into one.

On this page:

- [Managing a project's versions](#)
- [Add a new version](#)
- [Add a start date](#)
- [Release a version](#)
- [Archive a version](#)
- [Merge multiple versions](#)
- [Edit a version's details](#)
- [Delete a version](#)
- [Reschedule a version](#)

For all of the following procedures, you must be logged in to JIRA as a [project administrator](#).

Managing a project's versions

1. Choose



> **Projects**, and select the relevant project.

2. Choose **Versions** in the left menu. The **Versions** page is displayed, showing a list of versions and each version's status. From here you can manage the project's versions as described on this page.

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

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

Add a new version

1. The Add Version form is located at the top of the Versions screen.
2. Enter the name for the version. The name can be:
 - simple numeric, e.g. "2.1", or
 - complicated numeric, e.g. "2.1.3", or
 - a word, such as the project's internal code-name, e.g. "Memphis".
3. Optional details such as the version description (text not HTML), start date and release date (i.e. the planned release date for a version) can also be specified.
4. Click the **Add** button. You can drag the new version to a different position by hovering over the 'drag' icon



at the left of the version name.

Add a start date

If specified, the **Start Date** is used by the Version Report. This gives you a more accurate report in cases where you might plan a version many weeks (or even months) in advance, but not actually commence work until closer to the release date.

Release a version

Before you begin: If you have integrated JIRA applications with [Atlassian's Bamboo](#), you can trigger a Bamboo build to run automatically when releasing a version in JIRA. The version will only be released if the build is successful. For more information, see [Running a Bamboo build when releasing a version](#).

1. On the Versions screen, hover over the relevant version to display the cog icon, then select **Release** from the drop-down menu.
2. If there are any issues set with this version as their 'Fix For' version, JIRA applications allow you to choose to change the 'Fix For' version if you wish. Otherwise, the operation will complete without modifying these issues.

To revert the release of a version, simply select **Unrelease** from the drop-down menu.

Archive a version

1. On the Versions screen, hover over the relevant version to display the cog icon, then select **Archive** from the drop-down menu.
2. 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.

To revert the archive of a version, simply select **Unarchive** from the drop-down menu.

Merge multiple versions

Merging multiple versions allows you to move the issues from one or more versions to another version.

1. On the Versions screen, click the **Merge** link at the top right of the screen.
2. The 'Merge Versions' popup will be displayed. On this page are two select lists — both listing all versions.
In the 'Merging From Versions' select list, choose the version(s) whose issues you wish to move. 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. It is only possible to select one version to merge to.
3. Click the **Merge** button. If you are shown a confirmation page, click **Merge** again to complete the operation.

Edit a version's details

1. On the 'Versions' screen, hover over the relevant version to display the pencil icon.
2. This will allow you to edit the version's Name, Description and Release Date.
3. Click the **Update** button to save your changes.

Delete a version

1. On the 'Versions' screen, hover over the relevant version to display the cog icon, then select **Delete** from the drop-down menu.
2. This will bring you to the 'Delete Version: <Version>' confirmation page. From here, you can specify the actions to be taken for issues associated with the version to be deleted. You can either associate these issues with another version, or simply remove references to the version to be deleted.

Reschedule a version

Rescheduling a version changes its place in the order of versions.

- On the 'Versions' screen, click the  icon for the relevant version, and drag it to its new position in the version order.

Creating 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.

Generating Release Notes

1. Select **Projects** in the navigation bar.

2. Select your project from the list, or select **View all projects** and navigate to your project.
3. Select **Versions** in the project sidebar (if you're using JIRA Software, select **Releases** in the project sidebar).
4. Select the **Version** whose release notes you wish to generate by clicking on it.
5. Select **Release Notes**.
6. Click the '**Configure Release Notes**' link to configure the release notes. The 'Configure Release Notes' page will be displayed:
 - Select the required project version for which the release notes will be generated in the '**Please select version**' drop-down.
 - Select the required format of the release notes — HTML and plain text format templates are provided in the '**Please select style**' drop-down.
7. Selecting the '**Create**' button will generate the release notes using the specified template in the specified format. The release notes will be displayed on screen and can be copied and pasted to another application.

Adding a New Format Template

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 Apache 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 of existing ones within the `jira-config.properties` file. For each new template format, corresponding entries must be added to the existing values of the following properties:
 - `jira.releasenotes.templatenames`
 - `jira.releasenotes.templates`

Notes:

- a. Corresponding entries in both of these properties must be in the same order.
- b. If these properties do not exist in your `jira-config.properties` file, then:
 - i. For each of these properties, add the property's name,
 - ii. followed by an '=',
 - iii. followed by the content of the property's corresponding `<default-value/>` element copied from your JIRA installation's `jpm.xml` file.
 - iv. Next, begin adding the corresponding entries for the new format template.

 See [Making changes to the jira-config.properties file](#) for more information.

4. The new format template is available for selection as a release note format template.

Also see the tutorial on [Creating a Custom Release Notes Template Containing Release Comments](#).

Project screens, schemes and fields

Information for each issue is held in the fields that are associated with that issue. You can tailor these fields to suit your organization's needs. The diagram below is a representation of how these fields are associated with an issue, via screens and schemes. A screen is the user's view of an issue, and the screen is mapped to a specific issue operation (such as creating an issue, or editing an issue) via a screen scheme. The screen scheme is then mapped to an issue type via the issue type screen scheme. This configuration is associated with the project, and is applicable to all issues within the project.

Customizing the fields, screens and schemes allows you to unlock the full power of your JIRA application, and ensure that your users are working efficiently and effectively. You can also set up notification schemes which will notify your users when their issues have been updated. The following pages in this section will help you to configure and customize JIRA to suit your needs.

Adding a custom field	Learn more about how custom fields work, and how you can add them to your issues so you make sure you get the information you need on each issue.
Specifying field behavior	Learn more about how you can change how a field behaves, and when it displays, to make sure your users always see and record the information

that's most important.

Defining a screen

[Learn more](#) about how you can change what displays on each screen, and how to associate the screens with issues and issue operations.

Notification schemes

[Learn more](#) about how you can create notification schemes that keep your users updated when there are updates on their issues.

Adding a custom field

JIRA applications let you add custom fields in addition to the built-in fields. When creating a custom field, you can choose between **Standard** and **Advanced** types. For standard types, a preview image is shown for each type, so you can see what you are creating in advance. This ensures that you get the custom field you want, much faster. To configure search templates or add contexts to custom fields, use the **Configure** option on each custom field.

Custom fields are always optional fields. This means that you can create a new custom field without requiring existing issues to be changed. The existing issues will contain no value for the new custom field, even if a default value is defined.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

Adding a field directly to an issue

JIRA Admins can add an existing field or create a custom field while in View Issue with the **Admin > Add field** option. You can even configure the options for that custom field without having to leave the screens you are presented with.

Adding a field using the Add Custom Field button

1. Choose



> **Issues.**

2. Select **Fields > Custom Fields**.
3. Select **Add Custom Field**. Select **All** to make sure you can see all available field types.
4. Select a field and click **Next**.

5. Configure the selection criteria for your field, as shown in the example for the Checkboxes field below:

Configure 'Checkboxes' Field

Name*

Description

Options*

Apples	x
Oranges	x
Pears	x

The **Field Name** will appear as the custom field's title in both entering and retrieving information on issues, whereas the **Field Description** is displayed beneath the data entry field when entering new issues and editing existing issues, but not when browsing issues.

6. Select **Create**.
7. Choose which options to display and select **Submit** to finish adding your new custom field.

Next steps

If you wish to change the context or other variables in your custom field, see [Configuring a custom field](#). You can also find more custom fields from add-ons listed on the [Atlassian Marketplace](#) (e.g. the [JIRA Toolkit](#)). To build your own custom field types, see the [tutorial](#) at the [JIRA Developer Documentation](#).

Configuring a custom field

You can modify each of the custom fields in your JIRA system by changing the following:

- **Name** — the label that appears to the left of the custom field when it is displayed to a user.
- **Description** — the Help text that appears below the custom field when it is displayed in the Simple Search column.
- **Search template** — the mechanism for making a custom field searchable.
- **Default value** — the default value of the custom field when it is first displayed to a user.
- **Options** (for select and multi-select fields only) — the values from which a user can choose. See [below](#).
- **User filtering** (for User Picker fields only) — the set of users from which a user can choose. See [below](#).
- **Context** — the combination of project(s) and issue type(s) for which a given **Default Value** and **Options** will apply. See [below](#).
 - ✔ You can create [multiple contexts](#), allowing you to specify different Default Values and Options for different combinations of projects and/or issue types.
- **Screens** — the screen(s) on which the custom field will appear when an issue is created, edited or transitioned through workflow. See [below](#) (also see [Defining a screen](#)).
- **Renderers** — (*for certain types of fields only*) — see [Configuring renderers](#) and [Specifying field behavior](#).
- **Hide/Show** — see [Specifying field behavior](#).
- **Required/Optional** — see [Specifying field behavior](#).

⚠ Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Viewing custom fields

1. Choose  **> Issues**.
2. Select **Fields > Custom Fields** to open the Custom Fields page.

Editing a custom field

Editing a custom field allows you to change its name (label), description (Help text) and search template.

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Edit**:
 - The **Name** is the label that appears to the left of the custom field when it is displayed to a user.
 - The **Description** is the Help text that appears below the custom field when it is displayed.
 - **Search Templates** are responsible for indexing a custom field, as well as making it searchable via Simple Search and Advanced Search (note that custom fields are not searchable via Quick Search). Every custom field type has a preconfigured search template, but you may select a different template using this procedure.
2. Modify the fields as desired and click **Update**.

On this page:

- [Viewing custom fields](#)
- [Editing a custom field](#)
- [Configuring a custom field](#)
- [Choosing screens](#)
- [Translating a custom field](#)
- [Tips for custom fields](#)
- [Troubleshooting custom fields](#)

Configuring a custom field

The *custom field context* (also known as *custom field configuration scheme*) is **not** related to the *field configuration scheme*, and specifies the following for the custom field:

- Default value
- Options
- The issue types and projects to which the default values and options apply

You can create multiple contexts if you need to associate different default values and options with particular projects or issue types.

Each custom field has a context named *Default Configuration Scheme for ...*, which is created automatically when you add your custom field.

Context

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Configure**.
2. Locate the context named *Default Configuration Scheme for ...* and click the **Edit Configuration** link.
3. Under **Choose applicable issue types**, select the issue type(s) to which you want the default value and options to apply. You can select any issue types if you wish.
4. Under **Choose applicable contexts**, select the project(s) to which you want the default value and options to apply. Note that this will apply to only issues with the selected issue type(s) as above.

Adding a new context

Adding a new context allows you to configure a custom field differently for different combinations of issue types and projects.

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Configure**.
2. Click the **Add new context** link. The 'Add configuration scheme context' page will be displayed (see below).
 - Under 'Add configuration scheme context', enter a 'Label' and 'Description' for your new context — these are used for administrative purposes only and will not be shown to your end-users.
 - Under 'Choose applicable issue types', select the issue type(s) to which you want the default value and options to apply. You can select **Any issue types** if you wish.
 - Under 'Choose applicable contexts', select the project(s) to which you want the default value and options to apply. Note that this will apply to only issues with the selected issue type(s) as above.

A custom field can only have one context per JIRA project. So you cannot have multiple contexts for different issue types in the same project.

Default value

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Configure**.
2. Locate the relevant context (there will usually only be one, named 'Default Configuration Scheme for ...') and click the **Edit Default Value** link in the right-hand column. The '**Set Custom Field Defaults**' page will be displayed and will be particular to the custom field type:
 - (For a select list or multi-select list) Select the appropriate default value from the drop-down list.
 - **i** To clear the default of a select field, click on the current default so it is no longer highlighted and then save, as described here: [Unable to De-select Default Value for Multi Select Custom Field](#).
 - (For a cascading select list) Select the appropriate default values from the drop-down lists (one for each level).
 - (For a date field) Specify a date, or tick the check-box to make the current date the default.
 - (For other types of fields) Type the appropriate default values from the drop-down lists (one for each level).
 - **i** Certain types of custom fields may not allow for defaults to be selected and will not have the **Edit Default Value** link.

Options

You can specify option values for custom fields of the following types:

- Select lists
- Multi select lists
- Cascading select lists
- Radio buttons
- Multi checkboxes

You can add, remove, re-order, sort the options alphabetically, and edit the text of an option value. You can also have HTML in an option value — be sure to use complete tag pairs, and check that the HTML will display correctly.

i These options are case insensitive, so when using a select or multi-select list for a notification scheme, `JIRA-ADMINISTRATORS` will match the `jira-administrators` group. This means you cannot have both a `JIRA-ADMINISTRATORS` and a `jira-administrators` option, as they have the same name.

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Configure**.
2. Locate the relevant context (there will usually only be one, named 'Default Configuration Scheme for ...'), and click the **Options** link in the right-hand column. The 'Edit Custom Field Options' page will be displayed (see below). Here you can:
 - Select from the **Edit parent select list** drop-down to choose which list to edit. (*For a cascading select list only*)
 - Click **Sort alphabetically** to automatically re-order the options alphabetically.
 - Click the arrows in the **Order** column, or specify a number and click the **Move** button, to re-order the options manually.
 - Click **Edit** to change the text of an option.
 - Click **Disable** to hide an option so that it is no longer available for selection. Options that have been used cannot be removed (to preserve data integrity), but due to changing business requirements, they may become invalid over time and so you may wish to make them unavailable for new issues.
 - Click **Delete** to remove an option. (This will only be possible for options that have not been used.)

User filtering

You can limit the set of users available in your user picker field. The users can be limited to users in specific groups and/or project roles.

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Configure**.
2. Click **Edit User Filtering**.
3. Click **Enable group or project role filtering**, then specify the groups and/or roles that you want to limit the user picker to.
The user picker will only show users that are in the groups and roles selected.
4. Click **Save**.

Choosing screens

1. Navigate to the **Custom Fields** page, locate the desired custom field and choose **cog icon > Screens**.
2. Select the checkboxes of the screens on which you wish to display this custom field.

i Note that field visibility depends on the *field configuration* (which is **not** related to the *custom field configuration scheme* described above). See [Specifying field behavior](#) for more information.

Translating a custom field

You can translate the name and description of any custom field that you create into another language. You can only select from the language packs that are installed in JIRA.

1. Navigate to the **Custom Fields** page, locate the desired custom field, and choose **cog icon > Translate**.

te.

2. Choose the language pack that this custom field translation will belong to (e.g. French), and enter the translated strings for the **Field Name** and **Description**.

Tips for custom fields

- **Limit the number of custom fields** — Be careful how many custom fields you define in JIRA. More than a thousand is a large number and may affect JIRA's performance. For more information, see [Scaling Jira 6.4](#).
- **Use fields for reporting** — Consider what reports are needed from JIRA and only create fields that support those fields. Custom field types, such as select and multi-select are great for reporting. On the other hand, text fields are not as useful, since people don't always enter data as expected by the report's query.
- **Combine field content** — If just want to make sure that someone remembers to enter some information, then consider a multi-line custom text field with a text template as a default value. You may also want to try using a "table grid" custom field which lets you enter data in a searchable table. The [Atlassian Marketplace](#) has add-ons that provide this kind of functionality. Note, the standard JIRA fields such as Description do not currently support default values (see [JRA-4812](#)).
- **Don't duplicate names** — Don't create new custom fields with the same name as other existing custom fields. Always check to see whether a custom field with the same name already exists before you create it. If you do, then choosing the correct field in JQL searches can become confusing for users. Also, don't create custom fields with the same name as the standard JIRA fields. For example, having two "Status" fields is particularly confusing.
- **Make names as generic as possible** — Give custom fields non-specific names that can be reused in other places later on. For example, instead of naming a field "Marketing Objective", name the field "Objective", and provide a description in the field configuration that states the JIRA projects where that field is used.

Troubleshooting custom fields

Using the JIRA admin helper

The JIRA admin helper can help you diagnose why a custom field is not showing on your screens. This tool is only available to JIRA administrators.

1. Navigate to the View Issue, Edit Issue or Create Issue screen where the field is not showing.
2. If you are viewing an issue, click **More Actions > Where is my field?** If you are creating or editing an issue, click **Configure Fields > Where is my field?**
3. Enter the name of the field.
4. Click **Submit**.

Tip: You can also access the 'Where is my field?' dialog via the cog menu for each issue in the issue navigator.

Changing the description of a custom field

Not changing the description in a field configuration means that any changes you make to a custom field's description are not seen.

JIRA allows you to define a description of a custom field, and if the [field configuration descriptions](#) are left empty then the original description text will appear when you create or edit an issue, and as help text in the Issue Navigator. However you can also define different description texts in each field configuration and this will override the original field description text.

For example if a custom field "My Field" is defined with a description of "This is my field" and no field configuration changes are made, then the displayed text will be "This is my field" as expected. If field configurations are used and a description "This is my excellent field" is set for the custom field in the field description, then the displayed text will be "This is my excellent field".

Specifying field behavior

A **field configuration** defines the behavior of *all fields* available in your JIRA installation, including JIRA's own 'fixed'/'built in' fields (known as 'system' fields) and [custom fields](#).

For each field, a field configuration specifies:

- the **description** that appears under the field when an issue is edited
- whether the field is **hidden** or **visible**
- whether the 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

On this page:

- [Managing multiple field configurations](#)
- [Modifying field behavior](#)

When defining field behavior for one or more JIRA projects and the fields used by the issue types in these projects, you typically start by adding one or more new field configurations (see [below](#)). You then begin [modifying the behavior](#) of individual fields in these new field configurations.

A new field configuration should be added for each project and issue type combination which requires specific fields to be present and/or fields that express unique behavior. You can then associate each new field configuration with a different issue type through a 'field configuration scheme'. A field configuration scheme can then be associated with one or more projects. For more information, please see the [Overview Diagram](#).

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

Managing multiple field configurations

You can create multiple field configurations for use on separate projects and issue types.

- Multiple field configurations are organized into [field configuration schemes](#), which associate field configurations with issue types.
- A scheme can then be associated with one or more projects, allowing you to control fields on a per project, per issue type basis. See [Associating field behavior with issue types](#) for more information.

About the 'Default Field Configuration'

When JIRA is installed, the **Default Field Configuration** is created automatically. All new projects are associated with this configuration. This configuration is also used for projects that are not associated with a field configuration scheme.

i It is not possible to delete the **Default Field Configuration**.

Adding a field configuration

1. Choose



> **Issues**.

2. Select **Fields > Field Configurations** to view all your field configurations.

3. Click the **Add New Field Configuration** button to open the **Add Field Configuration** dialog box.

4. Complete the **Add Field Configuration** dialog box:

- **Name** — enter a short phrase that best describes your new field configuration.
- **Description** (*optional but recommended*) — enter a sentence or two to describe when this field configuration should be used.

5. Click the **Add** button to add your new field configuration to JIRA. Once you have added your new field configuration, you can then begin modifying the behavior of its fields ([below](#)).

i You will be taken directly to the **View Field Configuration** page, where you can modify the behavior of fields in your new field configuration. See [Modifying field behavior \(from step 4\)](#) below for details.

Editing a field configuration

1. Choose



> **Issues.**

2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Edit** link next to the field configuration you wish to edit.
4. On the **Edit Field Configuration** page, edit the field configuration's **Name** and **Description**.

i Please note: The **Default Field Configuration** cannot be edited.

Deleting a field configuration

1. Choose



> **Issues.**

2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Delete** link next to the field configuration you wish to delete.
 - i** You will be prompted to confirm this operation.

i Please note:

- The **Default Field Configuration** cannot be deleted.
- You can only delete a field configuration that is not associated with a [field configuration scheme](#). The **Delete** link will not be available for field configurations which are associated with one or more field configuration schemes.

Copying a field configuration

1. Choose



> **Issues.**

2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Copy** link next to the field configuration you wish to copy.
4. On the **Copy Field Configuration** page, specify the **Name** and **Description** for the field configuration to be copied.
 - i** The (initial) field settings between the original and copied field configurations will be identical.

i Please note: a newly created field configuration will not take effect until you:

1. [Associate your new field configuration to one or more issue types.](#)
2. [Associate that field configuration with one or more projects.](#)

See [Associating field behavior with issue types](#) for more information.

Modifying field behavior

To modify the behavior of fields in JIRA, you need to modify the field configurations that those fields have been defined in.

1. Choose



> **Issues.**

2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Locate the field configuration of interest and click the **Configure** link to open the **View Field Configuration** page, which lists all system and custom fields in your JIRA installation for that field configuration.

i Please note:

- The **Edit** link only allows you to change the **Name** and **Description** of the field configuration, not of individual fields.
- Note that the **Edit** link is not available for the **Default Field Configuration** on the **View Field Configuration** page (listing all field configurations defined in your JIRA installation).

4. In the **Operations** column, you can perform the following actions for any field:
 - **Edit** — change the field's description (i.e. help text).
 - **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).

i Please note: a newly created field configuration will not take effect until you:

1. Associate your new field configuration to one or more issue types, and then
2. Associate that field configuration with one or more projects.

See [Associating field behavior with issue types](#) for more information.

Editing a field's description

Fields can be given descriptions to better identify the meaning of the field. These descriptions are typically displayed under the fields they are associated with when creating or editing an issue.

1. Follow the first three steps above (in [Modifying field behavior](#)) to access the field configuration whose field's description you wish to edit.
2. Click the **Edit** link next to the field you want to change and update the field's description.
3. Click the **Update** button to save your changes.

Hiding or showing a field

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

i Please note:

- 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.
- 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](#), [security level](#), and [custom fields](#).)
- The fields **Summary** and **Issue Type** cannot be hidden and as such there is no **Hide** option available for these fields.
- If you hide the **Fix Version/s** field, the Change Log report will not work.

1. Follow the first three steps above (in [Modifying field behavior](#)) to access the field configuration whose fields you wish to hide or show.
2. Do either of the following:
 - If you no longer want to expose a field through JIRA's user interface, click the **Hide** link associated with that field.
 - i** You can make this field visible again at any time by clicking the **Show** link.
 - If you want to show a field (which is currently hidden) through JIRA's user interface, click the **Show** link associated with that field.
 - i** You can hide this field again at any time by clicking the **Hide** link.

Making a field required or optional

Certain fields within your organization 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.

1. Follow the first three steps above (in [Modifying field behavior](#)) to access the field configuration whose fields you wish to hide or show.
 - i** When viewing a field configuration (see [above](#)), fields which are already required have that indication next to their name.
2. Do either of the following:

- To make a field mandatory when used through JIRA's user interface, click the **Required** link associated with that field.
 -  The text **Required** will appear next to the field's name.
- To make a field optional, click the **Optional** link associated with that field.
 -  The **Required** text next to the field's name will disappear.

 **Please note:**

- Fields that are hidden cannot be set to required.
- If you make a field **Required**, ensure that the field is present on your **Create Issue** screen(s).
 - Note that you can have different field configurations for different projects and issue types (see [Associating field behavior 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 screen and issue operation mappings with an issue type](#)).
 - Be aware that there is a feature request ([JIRA-5783](#)) to make a field required for only one transition. If you are interested, please watch that issue for status updates.

Changing a field's renderer

Before you change the renderer for a specific field, please read [Configuring renderers](#), paying particular attention to the [Implications for JIRA operations](#) section.

1. Follow the first three steps above (in [Modifying field behavior](#)) to access the field configuration whose field's renderer you wish to change.
 -  When viewing a field configuration (see [above](#)), the **Name** column indicates which renderers are currently enabled for all renderable fields, with the current renderer shown in brackets immediately below its field name.
2. Click the **Renderers** link 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.
3. 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**.

 Changing the renderer only affects how a JIRA field's content is *displayed* or how a user *interacts* with a multi-select field — it does not affect the issue data that exists in the system. Hence, you can therefore toggle between renderer types safely.

Associating field behavior with issue types

A **field configuration scheme** associates (or "maps") [field configurations](#) to issue types in a project. In turn, a field configuration scheme can be [associated](#) with one or more projects.

This means that you can define different [field configurations](#) for each issue type that is available in a given project. For example, it is possible to have separate field configurations for the **Bug** the **Improvement** issue types (whose associations are defined in a field configuration scheme) for a project called 'Test'. Refer to the [Overview Diagram](#) for more information.

Because a field configuration scheme can be associated with more than one project (and associations between field configurations and issue types in a field configuration scheme are flexible), you can minimize your administrative workload as you can reuse the same field configuration for the same (or different) issue types across multiple projects.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

Adding a field configuration scheme

1. Choose  **> Issues**.
2. Select **Fields > Field Configurations** to view all your field configurations.
3. Click the **Add New Field Configuration Scheme** button to open the **Add New Field Configuration**

- [Adding a field configuration scheme](#)
- [Editing a field configuration scheme](#)
- [Deleting a field configuration scheme](#)
- [Copying a field configuration scheme](#)
- [Associating a field configuration scheme with a project](#)

Scheme dialog box.

4. Complete the **Add New Field Configuration Scheme** dialog box:
 - **Name** — enter a short phrase that best describes your new field configuration scheme.
 - **Description** (*optional but recommended*) — enter a sentence or two to describe when this field configuration scheme should be used.
5. Click the **Add** button to add your new field configuration to JIRA.

i You will be taken directly to the **Configure Field Configuration Scheme** page, where you can start associating issue types with field configurations in your new field configuration scheme. See [Modifying field behavior \(from step 4\)](#) for details.

Associating an issue type with a field configuration

1. Choose  **> Issues.**
2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Configure** link for the [field configuration scheme](#) in which to create an association between an a field configuration and an issue type. The **Configure Field Configuration Scheme** page will appear, showing the scheme's current mappings of field configurations to issue types.

i If you have not added any new field configurations since installing JIRA, you will only have JIRA's **Default Field Configuration** to work with.
4. Click **Associate an Issue Type with a Field Configuration**.
5. Select the desired issue type and field configuration and click the **Add** button.

i Please note:

- An issue type can only have one association within a given configuration scheme.
- 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 an issue type and a field configuration

1. Choose  **> Issues.**
2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Configure** link for the [field configuration scheme](#) that contains the association between a field configuration and issue type you want to remove. The **Configure Field Configuration Scheme** page will appear, showing the scheme's current mappings of field configurations to issue types.

i If you have not added any field configurations since installing JIRA, you will only have JIRA's **Default Field Configuration** to work with.
4. Click the **Remove** link next to the issue type you wish to remove from the scheme.

i Please note: The **Default** entry cannot be removed from the scheme.

Associating an issue type with a different field configuration

1. Choose  **> Issues.**
2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Configure** link for the field configuration scheme contains an association between a field configuration and issue type you want to change. The **Configure Field Configuration Scheme** page will appear, showing the scheme's current mappings of field configurations to issue types.

i If you have not added any field configurations since installing JIRA, you will only have JIRA's **Default Field Configuration** to work with.
4. Click the **Edit** link next to the issue type whose field configuration you wish to change.
5. Select the new **Field Configuration** you would like to associate with this issue type.
6. Click the **Update** button.

Editing a field configuration scheme

1. Choose  **> Issues.**
2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Edit** link next to the [field configuration scheme](#) whose name and description you wish to modify.
4. On the **Edit Field Configuration Scheme** page, edit the **Name** and **Description** of the field configuration scheme.
5. Click the **Update** button.

Deleting a field configuration scheme

1. Choose  **> Issues.**
2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Click the **Delete** link next to the [field configuration scheme](#) you wish to delete. You will be prompted to confirm your deletion.

i You can only delete a field configuration scheme that is not associated with a [project](#). The **Delete** link will not be available for field configuration schemes which are associated with one or more projects.

Copying a field configuration scheme

1. Choose  **> Issues.**
2. Select **Fields > Field Configurations** to open the View Field Configurations page, which lists all your field configurations.
3. Select **Administration > Issues > Fields > Field Configuration Schemes** (tab) to open the **View Field Configuration Schemes** (above), which lists all your field configuration schemes (if any exist).
4. Click the **Copy** link next to the [field configuration scheme](#) you wish to copy.
5. On the subsequent page, specify the **Name** and **Description** of the field configuration scheme to be copied.
6. Click the **Copy** button.

i The (initial) associations between field configurations and issue types in both the original and copied field configuration schemes will be identical.

Associating a field configuration scheme with a project

To make your JIRA projects use your field configurations, you need to associate these field configuration(s) with issue types in a field configuration scheme (above) and then associate this field configuration scheme with a project. (This association means that the field configuration scheme will be applied to the project.) Once this is done:

- The issues in this project will use the field configuration(s) 'mapped' to their issue type (defined by the field configuration scheme associated with the project)
but also:
- The issue types available to this project are defined by the [issue type scheme](#) associated with the project.

Therefore, even though a project's field configuration scheme may associate various different field configurations with a large set of issue types, only a subset of these issue types (as defined by the project's issue type scheme) and hence, field configurations themselves, may be available in that project. In other words, the issue types available to a project are restricted by the project's issue type scheme.

i Note that newly created projects are not associated with any field configuration schemes and hence, use the **Default Field Configuration** for all issues.

To associate a field configuration scheme with a project:

1. Access the **Project Summary** administration page for your project (see [Configuring a project](#)).
2. In the **Fields** section of this page, click the name of the current [field configuration scheme](#).
3. Click the **Actions** dropdown menu and choose **Use a different scheme**.
4. In the resulting page, select the scheme you want to associate with this project.
 - i** Selecting *None* will result in all issue types available to your project using JIRA's **Default Field Configuration**.
5. Click the **Associate** button. You will be returned to the **Project Summary** administration page, with the project now associated with the selected field configuration scheme.

Configuring renderers

Renderers are configured on a per field basis. To configure a renderer for a particular field, see [Specifying field behavior](#). Note that you can configure the same field differently for different projects and issue types — see [Associating field behavior with issue types](#).

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.

Please read [Implications for JIRA operations](#) below before configuring renderers.

On this page:

- [Renderable fields](#)
- [Renderer types](#)
- [Implications for JIRA operations](#)
- [Configuring renderers](#)

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.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Renderable fields

Potentially any field within JIRA applications can be a renderable field, but this only really makes sense in the case of text-based fields (for the default text renderer and the wiki style renderer) and multi-select fields (for the autocomplete renderer and the select list renderer). The following table shows the JIRA fields that are renderable out-of-the-box:

Field	Available Renderers
Description	Wiki style renderer (default), Default text renderer
Comment	Wiki style renderer (default), Default text renderer
Environment	Wiki style renderer (default), Default text renderer
Component	Autocomplete renderer (default), Select list renderer
Affects version	Autocomplete renderer (default), Select list renderer
Fix version	Autocomplete renderer (default), Select list renderer
Custom field of type "Free Text Field (unlimited text)"	Wiki style renderer (default), Default text renderer
Custom field of type "Text Field"	Wiki style renderer (default), Default text renderer
Custom field of type "Multi Select"	Select list renderer

Custom field of type "Version Picker"	Autocomplete renderer (default), Select list renderer
---------------------------------------	---

Renderer types

JIRA ships with the following renderers:

- for text fields: [wiki style renderer](#) and [default text renderer](#)
- for multi-select fields: [autocomplete renderer](#) and [select list renderer](#)

Default text renderer

The default text renderer renders a field's content as plain text, with the following additional auto-linking features:

Content	Description	Sample
JIRA issues	If the text contains text that resolves to a JIRA issue key, an HTML link will be generated, pointing to that issue.	<div style="border: 1px solid black; padding: 5px;"> <p>Description _____</p> <p>This relates to ANGRY-304</p> </div>
Web links	If the text contains text that resolves to a web page on a website, an HTML link will be generated, pointing to that web link.	<div style="border: 1px solid black; padding: 5px;"> <p>Description _____</p> <p>More details are found in https://answers.atlassian.com.</p> </div>

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

Wiki style renderer

The wiki style renderer allows a user to enter wiki markup to produce HTML content.

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 organize your document
 - Bullets, numbering, tables and quotations
 - Images, screenshots, and emoticons
 - Powerful mini-applications using macros
- A full notation guide can be found [here](#).

The wiki style renderer can only be used with JDK 1.4 and up. The renderer will not run on JDK 1.3.

Please note that some fields may require further field behavior configurations to be enabled — see [Specifying field behavior](#).

Wiki style renderer macro support

The Wiki style 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.

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.

Autocomplete and select list renderers

The autocomplete and select list renderers let you start typing text, which is then autocompleted, or to select from a drop-down list of options:

The image shows a screenshot of a JIRA form field. The field is labeled "Fix Version/s:". The input field contains the text "1.". A dropdown menu is open, displaying a list of options under the heading "Released Versions". The options are "1.3", "1.2", and "1.1". The option "1.3" is currently selected and highlighted in blue.

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 wiki style 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 are the same

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 you with any warnings.

If the renderer types are different

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 you with a warning so that you are 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.

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 wiki style renderer for both of the fields. Let's say you also have a project 'B' which is configured to use the default text renderer for the 'Custom text area' field and the wiki style 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 following:

- Choose Issues
Selected 2 issues from 1 project(s)
- Choose Operation
- **Operation Details**
- Confirmation

Step 3 of 4: Operation Details

Choose the bulk action(s) you wish to perform on the selected 2 issue(s).

- Change Issue Type

?

Some issue types are unavailable due to incompatible field configuration and/or workflow associations.
- Change Priority

?
- Change Fix Version/s

Start typing to get a list of possible matches or press down to select.
- Change Component/s

Start typing to get a list of possible matches or press down to select.
- Change Assignee

? Assign to me
- Change Reporter

Start typing to get a list of possible matches.
- Change Environment

For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).
- Change Due Date

📅
- Change Comment

🔒 ? 👤 Viewable by All Users

Email notifications

JIRA allows for extensive configuration in relation to email notifications, and can send out two types of emails, HTML, and text. See [Creating a notification scheme](#) and [Configuring email notifications](#) for more information.

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 wiki style 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 wiki style 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 wiki style 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.

The unrendered wiki markup will be shown in Excel cells for fields that use the wiki style 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 wiki style 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=raw
```

Editing a renderable custom field's default value

When editing a renderable custom field's default value, even if it is only ever configured to use the wiki style renderer you will not be presented with the edit and preview options. Unfortunately, in this context, it is not possible to tell which renderer should be used for editing. However, 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 wiki style renderer.

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 [Specifying field behavior](#).

Enabling a renderer plugin

Renderers within JIRA are implemented as JIRA plugins. The macros that the wiki style renderer uses are also implemented as JIRA plugins. For general information on plugins, see the [JIRA Plugin Guide](#).

Note that plugins are configured at an instance-wide level — it is not possible to configure plugins at a project/issue type level.

Configuring a renderer plugin

Renderers and their dependant components, except for the default text renderer, can be enabled or disabled as follows.

1. Choose



> **Add-ons.**

2. The 'Find add-ons' screen shows add-ons available via the [Atlassian Marketplace](#). Choose **Manage Add-ons** to view the plugins currently installed on your JIRA instance.
3. Select **Manage Add-ons**, and then search for 'renderer', filtering for system add-ons, as shown here:

This screen displays all the configured renderers within JIRA.

- Click the **Disable** button to deactivate the renderer for the entire instance of JIRA.

Any fields still set up to use a disabled renderer will fall back to the default text renderer. When you attempt to edit the field, a warning message alerts you to the fact that you are configured to use a renderer that is not available.

When a renderer is disabled it will not be available for selection when changing a field's renderer. To enable the renderer, click the **Enable** button. Enabling or 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 affecting any data.

Configuring macro plugins for the wiki style renderer

The macros used by the wiki style renderer can be enabled or disabled as follows.

1. Choose



> Add-ons.

- The 'Find add-ons' screen shows add-ons available via the [Atlassian Marketplace](#). Choose **Manage Add-ons** to view the plugins currently installed on your JIRA instance.
- Select **Manage Add-ons**, and then search for 'renderer', filtering for system add-ons.
- Expand the **Wiki Renderer Macros Plugin** to display the following:

System Add-ons

⚠ These add-ons are integral parts of your JIRA system. They cannot be uninstalled. Disabling or removing them will have serious effects, and may render JIRA inoperable. Do not make changes here unless instructed by Atlassian Support.

Wiki Renderer Macros Plugin

JIRA's base system macros.

No Screenshots Available	Version: 1.0 Developer: Atlassian Add-on key: com.atlassian.jira.plugin.system.renderers.wiki.macros	7 of 8 modules enabled
--------------------------	---	------------------------

- anchor**^(anchor)
Create an anchor that allows people to link to a specific point in a page
- code**^(code)
Format blocks of source-code or XML
- quote**^(quote)
Generate blockquotes that may contain multiple paragraphs or complex markup
- noformat**^(noformat)
Create blocks of text where other wiki formatting is not applied
- panel**^(panel)
Draw a panel with an optional title and border
- color**^(color)
Change the color of the contained text
- loremipsum**^(loremipsum)
Insert paragraphs of "lorem ipsum" space-filler text

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. 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, see the documentation on [Writing Macros](#).

Defining a screen

Screens group all available fields (or a subset of all available fields) defined in JIRA applications, and organize them for presentation to a user. Through screens, you can control what fields are displayed to the user during [issue operations](#) (e.g. **Create Issue** and **Edit Issue** dialog boxes) or [workflow transitions](#) (e.g. **Resolve Issue** dialog box), as well as define the order in which these fields are shown to them. A screen also allows you to split subsets of fields across multiple tabs.

On this page:

- [Adding a screen](#)
- [Editing a screen's details](#)
- [Copying a screen](#)
- [Deleting a screen](#)
- [Configuring a screen's tabs and fields](#)
- [Activating a screen](#)

When it comes to field visibility, screens functionally overlap slightly with [field configurations](#). For example, on the **Create Issue** dialog box, users will only see issue fields that:

- are present on the screen associated with the issue's **Create Issue** issue operation
- are also *not hidden* in the field configuration applicable to the issue (as defined by the project's [field configuration scheme](#))
- the user has [permission](#) to edit (e.g. the **Due Date** field can only be edited by users with the **Schedule Issues project permission**)

Hence, a field may be present on a screen used by a project, but if that field is hidden in the field configuration used by the project, that field will not be visible to the user when that screen in the project is displayed.

✔ If a particular field needs to be hidden at all times, it is easier to hide the field in the relevant field configuration than remove it from all screens.

⚠ Be aware that any newly created screen in JIRA is not usable by a JIRA project until it has been associated with either:

- An issue operation and issue type (via a [screen scheme](#) and then [issue type screen scheme](#))
OR
- A workflow transition.

See [Activating a screen](#) (below) for details.

JIRA applications ship with the **Default Screen**, **Resolve Issue Screen** and **Workflow Screen**, which are used as described below:

- **Default Screen** — used for the default issue operations for creating, editing or viewing an issue.
- **Resolve Issue Screen** — used for the transition view for the default **Close Issue** and **Resolve Issue** transitions, originating from the **Open**, **In Progress** and **Reopened** steps in JIRA's default workflow.
- **Workflow Screen** — used for the transition view for the default **Reopen Issue** transitions, originating from the **Resolved** and **Closed** steps and **Close Issue** transition, originating from the **Resolved** step in JIRA's default workflow. The **Workflow Screen** defines a smaller set of fields than the **Resolve Issue Screen**.

Adding a screen

To add a new screen to JIRA:

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose 
 - > **Issues**. Select **Screens** to open the View Screens page, which lists all screens that have been defined in JIRA.
3. Click the **Add New Screen** button to open the **Add New Screen** dialog box.
4. Complete the **Add New Screen** dialog box:
 - **Name** — enter a short phrase that best describes your new screen.
 - **Description** — enter a sentence or two to describe the situations screen will be used.
5. Click the **Add** button to add your new screen to JIRA.
 - ⓘ You will be taken directly to the **Configure Screen** page, where you can add fields to your new screen. See the [Configuring a screen's fields](#) section below for details.

Editing a screen's details

To change a screen's name and/or description:

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose 
 - > **Issues**. Select **Screens** to open the View Screens page, which lists all screens that have been defined in JIRA.
3. Click the **Edit** link next to the appropriate screen.
4. You will now be directed to the **Edit Screen** page where you can edit the name and/or description of the Screen.

Copying a screen

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose



- > **Issues**. Select **Screens** to open the View Screens page, which lists all screens that have been defined in JIRA.
- Click the **Copy** link next to the Screen you wish to copy. You will be directed to the **Copy Screen** page, where you can enter a name and a description for the new Screen.

Deleting a screen

- Log in as a user with the **JIRA Administrators** [global permission](#).
- Choose



- > **Issues**. Select **Screens** to open the View Screens page, which lists all screens that have been defined in JIRA.
- Click the **Delete** link next to the screen you wish to delete. You will be prompted to confirm your deletion

i Screens that are associated with one or more screen schemes, or one or more workflow transitions, cannot be deleted.

Configuring a screen's tabs and fields

You can configure the fields that display on a particular screen by adding/removing fields, as well as reordering them. Tabs can also be used to help group related fields. Tabs are useful for organizing complex screens, as you can place less used fields onto separate tabs. You can also add, remove and reorder tabs, as well as rename them.

To configure a screen's tabs and fields:

- Log in as a user with the **JIRA Administrators** [global permission](#).
- Choose



- > **Issues**. Select **Screens** to open the View Screens page, which lists all screens that have been defined in JIRA.
- Click the **Configure** link next to the screen you want to add a field to. You can perform the following operations:

Operation	Instructions
Add a tab	Click Add Tab . Enter the name of the new tab in the dialog that appears and click Add .
Move a tab	Hover over the dotted part of the tab (next to the tab name) and drag the tab to the desired position.
Rename a tab	<ol style="list-style-type: none"> Hover over the tab name and click the pencil icon. Enter the new name and click OK.
Delete a tab	Hover over the tab name and click the X .
Add a field	<ol style="list-style-type: none"> Click the tab that you want to add the field to. Type the name of the field in the drop-down displayed at the bottom of the current fields. Field suggestions will appear as you type. Click Add Field to add it to the current tab.
Move a field	<p>Hover over the dotted part of the field (next to the field name) and drag the field to the desired position.</p> <p>Move a field to a different tab by dragging it to the name of the tab and dropping it.</p>

Delete a field	Hover over the field and click the Delete button that appears.
----------------	---

Tips on configuring screens

- **Date fields on View Issue screen** — Fields of type 'Date' will always be displayed in the 'Dates' area of the default 'View Issue' screen, regardless of how you reorder them. This applies even if the dates are custom fields.
- **System fields on View Issue screen** — System fields on the default 'View Issue' screen (e.g. Summary, Security Level, Issue Type, etc.) are fixed. This means that they will always appear in the same place on the 'View Issue' screen, even if you configure the Screen to move them onto a separate tab. Custom fields of related to Dates and People will also appear in their fixed section of the view issue screen. If none of the fields on a tab contain data then the tab is not shown. To make a tab show up, make sure it has a custom field with a type such as Text or Select and that the field has a value.
Note, this information only applies to the screen associated with the 'View Issue' operation in a screen scheme.
- **Timetracking** — You can add the ability to log work and/or specify/modify time estimates to a screen by adding the special **Log Work** and/or **Time Tracking** fields respectively.
 - If these fields cannot be found in the **Add Field** selection box and they have not already been added to the screen, check whether JIRA's [Time Tracking feature](#) has been enabled. These fields will not be available to add to any screen if Time Tracking is disabled.
 - If any screens have the **Log Work** or **Time Tracking** fields and JIRA's Time Tracking feature is subsequently deactivated, those screens will retain these fields until you specifically remove them. However, the fields will not be visible to the user until Time Tracking is reactivated.
- **Renaming standard JIRA fields** — You cannot rename the standard JIRA fields (e.g. Priority, Summary, etc) via the JIRA administration console. If you want to rename the standard JIRA fields, you will need to modify files in your JIRA installation. Please see [this knowledge base article](#) for instructions. Note, renaming the standard JIRA fields is not supported.

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](#); **or**
- Associate the Screen with a **Workflow Transition** (e.g. 'Resolve Issue') — see [Configuring Workflow](#).

Associating a screen with an issue operation

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 three issue operations for which you can choose a screen:

- **Create issue** — the screen that is shown when an issue is being created.
- **Edit issue** — the screen that is shown when an issue is edited.
- **View issue** — the screen that is shown when a user views an issue.

In a screen scheme, you can specify the same screen (or choose different screens) for these issue operations. Once you have created your screen scheme, you will need to activate it by associating the screen scheme with issue types via an ['issue type screen scheme'](#). (In turn, issue type screen schemes are associated with JIRA projects.)

i Please be aware that although it is possible to associate any screen defined in your JIRA installation with either a screen scheme or a [workflow transition view](#), screen schemes and workflow transition views are distinct and unrelated.

On this page:

- What is a 'screen scheme'?
- Adding a screen scheme
- Editing a screen scheme's details
- Deleting a screen scheme
- Copying a screen scheme
- Configuring a screen scheme
- Activating a screen scheme

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Adding a screen scheme

Depending on your requirements, you may want to create multiple screen schemes, and associate them with different projects and issue types.

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page.
3. Click the **Add New Screen Scheme** button.
4. Fill out the details for the new screen scheme on the form that is displayed.

Note: The default screen is used for issue operations that do not have a screen associated with them.

Editing a screen scheme's details

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page.
3. Click **Edit** next to the desired screen scheme.
4. 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.

Deleting a screen scheme

Note that screen schemes that are associated with an issue type screen scheme cannot be deleted. You will first need to edit the issue type screen scheme and remove the screen scheme.

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page.
3. Click the **Delete** link next to the desired screen scheme. You will be prompted to confirm your deletion.

Copying a screen scheme

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page.
3. Click **Copy** next to the screen scheme you wish to copy.
4. You will now be directed to the Copy Screen Scheme page. Enter the name and description of the new screen scheme and click the **Copy** button.

Configuring a screen scheme

Associating a screen with an issue operation

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page:

Name	Issue Type Screen Schemes	Operations
Angry Nerds Feature Screen Scheme	• Angry Nerds Issue Type Screen Scheme	Configure · Edit · Copy
Angry Nerds Screen Scheme <small>Screens for Nerd Transitions</small>	• Angry Nerds Issue Type Screen Scheme	Configure · Edit · Copy
Customer Screen Scheme <small>for USER project</small>	• User Testing Issue Type Screen Scheme	Configure · Edit · Copy
Default Screen Scheme <small>Default Screen Scheme</small>	• Default Issue Type Screen Scheme • GTM Issue Type Screen Scheme	Configure · Edit · Copy
Experiment Screen Scheme <small>Screens used for experiment issues</small>	• Default Issue Type Screen Scheme	Configure · Edit · Copy

3. Locate the screen scheme in which you are interested, and click the **Configure** link next to it.
4. Click **Associate an Issue Operation with a Screen**, and select the following options:
 - a. Select the Issue Operation with which you wish to associate a screen.
 - b. Select the desired screen.

Important notes

1. 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.
2. 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. Click **Edit** next to the *Default* entry to change the screen that is associated with it.
3. The View Issue operation only allows you to control the layout of **custom fields** in the middle of the View Issue page. It ignores all the non-custom fields on the screen.

Editing an association

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page.
3. Locate the screen scheme in which you are interested, and click the **Configure** link next to it. The Configure Screen Scheme page is displayed.
4. Click **Edit** next to the issue operation you wish to edit. The Edit Screen Scheme Item page is displayed.
5. Select the desired screen and click **Update**.

Deleting an association

1. Choose



> **Issues.**

2. Select **Screens > Screen Schemes** to open the View Screen Schemes page.
3. Locate the screen scheme in which you are interested, and click the **Configure** link next to it. The

Configure Screen Scheme page is displayed.

4. Click the **Delete** link next to the issue operation you wish to remove.

Activating a screen scheme

To activate a screen scheme, you need to associate it with one or more projects and issue types, using issue type screen schemes.

1. Configure an issue type screen scheme to use the screen scheme.
2. Associate the issue type screen scheme with a project.

For details of both procedures, see [Associating screen and issue operation mappings with an issue type](#).

Associating screen and issue operation mappings with an issue type

What is an 'issue type screen scheme'?

An 'issue type screen scheme' associates a [screen scheme](#) (which defines mappings between screens and issue operations) with [issue types](#). Hence, an issue type screen scheme allows you to specify different [screens](#) for different issues types when used *for the same* issue operation (e.g. 'Create Issue') in a given JIRA project. For more information, please see the [overview diagram](#).

By default, your JIRA system contains an issue type screen scheme that's called a **default issue type screen scheme**. You may want to edit this scheme or copy it to make a new one.

Configuring an issue type screen scheme

The configuration of an issue type screen scheme involves associating an issue type(s) with a particular screen scheme. For example, associating the 'Bug' issue type with the 'Default Screen Scheme', and then associating the 'Improvement' issue type with the 'Improvement Screen Scheme'.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- [What is an 'issue type screen scheme'?](#)
- [Configuring an issue type screen scheme](#)
- [Adding an issue type screen scheme](#)
- [Editing an issue type screen scheme](#)
- [Deleting an issue type screen scheme](#)
- [Copying an issue type screen scheme](#)
- [Associating an issue type screen scheme with a project](#)

Associating an issue type with a screen scheme

1. Choose



> **Issues.**

2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Configure** link next to the desired issue type screen scheme.
4. Click **Associate an issue Type with a Screen Scheme** and select the following options:
 - a. Select an **Issue Type** you wish to associate a screen scheme with.
 - b. Select the desired **Screen Scheme**.
5. Click the **Add** button and the new association will be added to the association list above.

Please 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.
- If there is no specific entry for an issue type, the screen scheme associated with the *Default* entry will be used.

Editing an association

1. Choose  **> Issues.**
2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Configure** link next to the desired issue type screen scheme, which opens the **Configure Issue Type Screen Scheme** page (see [above](#)).
4. Click the **Edit** link next to the issue type you wish to edit, which displays the **Edit Issue Type Screen Scheme Entry** page.
5. Select the screen whose association you wish to change, and click the **Update** button.

Deleting an association

1. Choose  **> Issues.**
2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Configure** link next to the desired issue type screen scheme, which opens the **Configure Issue Type Screen Scheme** page (see [above](#)).
4. Click the **Delete** link next to the issue operation you wish to remove.

The *Default* entry is used for all issue types that do not have a specific entry in the scheme. It cannot be deleted.

Adding an issue type screen scheme

1. Choose  **> Issues.**
2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Add Issue Type Screen Scheme** button.
4. Enter the name for the new scheme. You can optionally add a description.
5. Select a screen scheme for the *Default* entry in the new scheme. The *Default* entry will be used for issue types that do not have a specific mapping in the scheme.
6. Click the **Add** button. The screen will automatically update the Issue Type Screen Schemes list with the new issue type screen scheme.

Editing an issue type screen scheme

1. Choose  **> Issues.**
2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Edit** link next to the desired issue type screen scheme to open 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.
4. Click the **Update** button, which returns you to the View Issue Type Screen Schemes page, with your updates now applied to the Issue Type Screen Schemes list.

Deleting an issue type screen scheme

1. Choose



> Issues.

2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Delete** link next to the issue type screen scheme you wish to delete.

Issue type screen schemes that are associated with a project cannot be deleted.

Copying an issue type screen scheme

1. Choose



> Issues.

2. Select **Screens > Issue Type Screen Schemes** to open the View Issue Type Screen Schemes page.
3. Click the **Copy** link next to the field screen you wish to copy, which opens the Copy Issue Type Screen Scheme page.
4. Enter the name and description of the new issue type screen scheme, and click the **Copy** button.

Associating an issue type screen scheme with a project

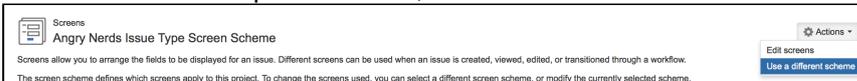
Once you have created and configured an issue type screen scheme to your desired settings, you can now associate the scheme with a project. This will apply your chosen screen scheme to each issue type within the selected project.

1. Choose



> Projects, and select the relevant project.

2. Select the project you wish to configure by clicking on its name.
3. Select **Screens**.
4. Click the **Actions** drop-down menu, and choose **Use a different scheme**:



5. Select the screen scheme you wish to associate with this project.
6. Click the **Associate** button.

To control which issue types apply to a project, please see '[Associating issue types with projects](#)'.

Creating a notification scheme

JIRA applications can generate **email notifications** for various *events* that happen throughout the lifecycle of an issue, including *custom events*. 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](#).

i You can use the same notification scheme for more than one project.

JIRA applications are pre-packaged with a notification scheme called **Default Notification Scheme**. *This scheme is associated with all new projects by default.* This means that if you have an [outgoing \(SMTP\) mail server](#) set up, that email notifications will be sent as soon as there is any activity (e.g. issues created) in the new project. However, you can disassociate this notification scheme from the project via the **Project Summary** page, as described [below](#). You can also modify this scheme or if you prefer, create other notification schemes for particular projects.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Creating a notification scheme

On this page:

- [Creating a notification scheme](#)
- [Adding an event recipient to a notification scheme](#)
- [Associating a notification scheme with a project](#)

1. Choose  **> Issues.**
2. Select **Notification Schemes** to open the Notification Schemes page, which lists all the current notification schemes in your JIRA installation.
3. Start creating the new notification scheme, by doing either of the following:
 - Click the **Copy** link to copy an existing notification scheme. If you have a notification scheme whose event recipients are reasonably similar to what you require, creating a copy is the quickest way to add a new scheme.
 - OR
 - Click the **Add Notification Scheme** button. On the **Add Notification Scheme** page, enter a name for the notification scheme and a short description of the scheme
4. If you added a new notification scheme or you copied an existing one but have clicked the **Edit** link to modify the automatically generated name and/or description of the copied notification scheme:
 - a. Enter a name (or modify the existing one) for the notification scheme (e.g. 'Angry Nerds Notification scheme').
 - b. *(Optional)* Enter a description (or modify the existing one) for the notification scheme.
 - c. Click the **Add** button to create the notification scheme.
5. Add notifications/recipients as described [below](#).
6. Associate your new notification scheme with a project as described [below](#).

Adding an event recipient to a notification scheme

To add a new recipient for a particular event to a notification scheme, you need to:

1. Identify the notification scheme used by the relevant project.
2. Add that recipient to the appropriate event in this notification scheme.

To add a new recipient for a particular event:

1. Choose  **> Issues.**
2. Select **Notification Schemes** to open the Notification Schemes page, which lists all the current notification schemes in your JIRA installation.
3. Locate the notification scheme of interest and click its linked name to open the **Edit Notifications** page for that notification scheme.
The **Edit Notifications** page lists all of the [events](#), along with the recipients who will receive notifications when each event occurs.
4. Click the **Add** link in the appropriate event row (see the list of [events](#) below), which opens the **Add Notification** page, where you can choose who to notify (about the event) from the list of available [recipients](#) (see below).
5. Select the appropriate recipient (filling in any required information for your particular choice of recipient).
6. Click the **Add** button. You are taken back to the **Edit Notifications** page (see [above](#)), with the notification you just specified now listed against the appropriate issue event.
7. If you make a mistake, or you would like to remove who is being notified, simply click the **Delete** link beside the person/group/role.

Associating a notification scheme with a project

1. Choose  **> Projects**, and select the relevant project.
2. At the lower-right of the **Project Summary** page, locate the **Notifications** section, click the name of the current scheme (e.g. **Default Notification Scheme**) or **None** (if the project is not yet associated with a scheme) to display details of the project's current notification scheme.
3. Click the **Actions** dropdown menu and choose **Use a different scheme** (or **Select a scheme**).
4. On the subsequent **Associate Notification Scheme to Project** page, which lists all available notification schemes, select the notification scheme you want to associate with the project and click the **Associate** button.

Events

JIRA applications support the following events, which can generate email notifications (as defined in a notification scheme).

Event	Description
Issue created	An issue has been entered into the system.
Issue updated	An issue has had its details changed. This includes the deletion of an issue comment.
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).
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 or out of 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.

i JIRA applications do not have a specific notification event for the deletion of issue comments. When an issue's comment is deleted, JIRA sends out an email notification as an 'Issue Updated' event.

Recipients

The following types of recipients can receive email notifications.

Recipient	Description
Current assignee	The user to whom the issue is currently assigned.
Reporter	The user who originally created the issue.

Current user	The user who performed the action that has triggered this event.
Project lead	The user who is managing the project to which the issue belongs.
Component lead	The user who is managing the component to which the issue belongs.
Single user	A particular user in your JIRA system.
Group	A particular group in your JIRA system.
Project role	The members of a particular project role for this project. Note that it is recommended to use project roles (rather than groups) in your notifications as this can help minimize the number of notification schemes in your system.
Single email address	Any email address that you wish to alert. A Single Email Address notification will only be sent if the issue is publicly viewable (as the email address of a non-JIRA user could be specified, in which case a security check is not possible). Publicly viewable issues are issues which have a Permission scheme that gives the 'Browse Projects' permission to 'Anyone' (any non-logged-in users). The text template is used for notifications to a single email address.
All watchers	All users who are watching the issue.
User custom field value	The value of a custom field of type <i>User Picker</i> or <i>Multi User Picker</i> that may have been associated with issues. An example of where this can be useful: if you have a custom User field called Tester, you can have the tester notified when an issue is resolved.
Group custom field value	The value of a custom field of type <i>Group Picker</i> or <i>Multi Group Picker</i> that may have been associated with issues..

i Please note:

- Email notifications will only be sent to people who have permission to view the relevant issue — that is, people who:
 - have the **Browse Projects** 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.
- JIRA can only send email notifications if SMTP email has been enabled (see [Configuring email notifications](#)).
- JIRA's default setting is to not notify users of their own changes. This can be changed on a per user basis via their profile preferences.

! Please also note:

JIRA will send notification emails to both the **previous assignee and the current assignee**, whenever the assignee field changes.

However, earlier versions of JIRA only sent a notification email to the previous assignee *if* the operation that changed the event was the **Assign Issue** operation. It did not send a notification if the issue was edited in some other way.

The `jira.assignee.change.is.sent.to.both.parties` advanced JIRA option allows this legacy behavior to be re-instated, for those customers who prefer this behavior.

See [JRA-6344](#) for more details.

Using the issue collector

What is an 'issue collector'?

The issue collector allows you to easily embed a JIRA feedback form into your own web site. This form is typically accessed by clicking a 'trigger' tab exposed along the edge of pages in your web site.

When used by people visiting your web site click this trigger tab and submit the resulting JIRA feedback form, an issue is conveniently created in JIRA.

Visitors to your web site do not require a user account in JIRA to use the JIRA feedback form.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Accessing JIRA's issue collectors

In JIRA applications, issue collectors are configured (and hence organized) on a per-project basis.

To access all issue collectors configured in JIRA:

1. Choose



> **System**.

2. Select **Issue Collectors** to open the Issue Collectors page, which shows a list of all existing issue collectors in your JIRA system.
3. Click the name of a project to access a more detailed list of issue collectors belonging to that project or click the name of an issue collector to access detailed information about it. On the issue collector page (containing detailed information), you can access:
 - An activity graph, showing the number of issues created via this issue collector (Y-axis) on a daily basis (X-axis).
 - A list of recent issues in reverse chronological order, which have been created via this issue collector.

On this page:

- What is an 'issue collector'?
- Accessing JIRA's issue collectors
- Adding an issue collector
- Embedding an issue collector into your web site
- Editing an issue collector
- Copying an issue collector
- Disabling or deleting an issue collector
- Known limitations

Related pages:

- Advanced use of the JIRA issue collector

To access issue collectors belonging to a specific project:

1. Choose



> **Projects**, and select the relevant project.

2. Select a project.
3. From the project administration page, click the **Issue Collectors** tab. The Issue Collectors page is displayed, listing any issue collectors that have already been set up in your project:

Name	Last Updated By	Issue Type	Description	Activity	Operations
Test issue collector	admin admin [Administrator]	Bug			Edit · Copy · Disable · Delete

4. Click the name of an issue collector to access detailed information about it — in particular, its recent activity and details on how to embed the issue collector into your web site.

Adding an issue collector

1. Choose



> **Projects**, and select the relevant project.

2. On the left of the Project Summary page, click the **Issue Collectors** tab. The Issue Collectors page is displayed, listing any issue collectors that have already been set up in your project.

3. Click the **Add Issue Collector** button to open the Add Issue Collector page.
4. In the top section of the Add Issue Collector page, specify the following:

Name	Specify the name of the issue collector, as you want it to appear throughout the JIRA user interface.
Description	Specify a description for the issue collector. This description will appear adjacent to the name of your issue collector, throughout the JIRA user interface.
Issue type	Select the type of issue that you want created in JIRA when visitors to your web site submit your issue collector's JIRA feedback form.
Issue reporter	Specify the username that will be the default reporter of JIRA issues created when visitors to your web site submit your issue collector's JIRA feedback form.
Match reporter?	<p>Select either of the following:</p> <ul style="list-style-type: none"> • Always use Issue Reporter — select this option to ensure that the default issue reporter you specify above, will always be the reporter of issues created by submission of the JIRA feedback form on your web site. • Attempt to match user session of submitter or submitter email address — select this option if you want the reporter of an issue created by submission of the JIRA feedback form on your web site, to be a JIRA user: <ul style="list-style-type: none"> • Who is logged in to JIRA when they submit a JIRA feedback form on your web site (in the same browser session). • Who's email address matches the email address specified in the email field of the JIRA feedback form. <p>Please note that if the JIRA user does not have the Create Issues project permission in your JIRA project, the default issue reporter you specify above will be used as the issue's reporter.</p>
Collect browser info	Select this option to collect meta-information about your browser's statistics, which will be incorporated into issues created by submission of the JIRA feedback form on your web site.

5. In the middle section of the Add Issue Collector page (entitled 'Trigger'), specify the following:

Trigger text	Specify a short, brief phrase that will appear on the trigger tab on your web site.
Trigger style	Choose the style in which the trigger tab will appear on your web site. 'Custom' will not display a trigger, but will add additional javascript to the generated script, so you can create a custom trigger on your web page.

6. In the lower section of the Add Issue Collector page (entitled 'Issue Collector Form'), specify the following:

Template	<p>Choose from the options provided. Typically, your choice would reflect the type of issue being created (i.e. chosen above). You can choose:</p> <ul style="list-style-type: none"> • A predefined template for your JIRA feedback form — either 'Got Feedback?' or 'Raise a Bug'. • Custom to create a custom JIRA feedback form, which allows you to specify your own wording on the dialog box, as well as add or remove other fields on the form, and change their positions on the form. <ul style="list-style-type: none"> • Please note that if a field is <i>required</i> for the chosen issue type but that field has: <ul style="list-style-type: none"> • No specified a default value, the field will automatically appear on the form. This field's position can be changed on the form, although it cannot be removed. • A default value but the field is not added to the form, then the field's default value is used when an issue is created via the issue collector. • Not all fields of types of fields can be added to the form, since some fields cannot be displayed to anonymous users. The fields types that can be displayed are: <ul style="list-style-type: none"> • <i>Standard Fields</i>: Summary, Description, Components, AffectsVersion, Environment, Priority, Attachment • <i>Custom Field Types</i>: Date Time, Radio Buttons, Multi-Checkbox, Multi-Select, Number, Select List, URL field, Version Picker, Cascading Select, Project Picker, Single Version Picker, Text Field, Free Text Field
Message	<p>Type a message, which appears in the blue 'information' panel along the top of the dialog box.</p>

7. Click the **Submit** button to save your changes.

Embedding an issue collector into your web site

After clicking the **Submit** button to save your new issue collector, a page containing code snippets is displayed. Use the code and information provided to embed your new issue collector into your web site.

✔ If you accidentally click away from this page, you can easily retrieve the information that was on it by accessing your issue collector's details ([above](#)) and scrolling to the end of the page.

Editing an issue collector

Editing an issue collector should not require any changes to web pages that include the issue collector, unless you change the trigger style to or from a custom trigger. Changing the trigger style to or from a custom trigger will change the generated javascript, so you may need to change what you embed in any web page that includes the issue collector.

1. Log in to JIRA as a [project administrator](#) or a user with the **JIRA Administrators global permission**.
2. Access the relevant project's list of issue collectors ([above](#)).
3. In the Operation drop-down for the issue collector you would like to edit, select **Edit** to open the Edit Issue Collector page.
4. Update the issue collector, as desired.
5. Click **Update** to save your changes.

Copying an issue collector

Copying an issue collector will create an entirely new issue collector and will not affect any existing issue collectors. You will need to embed it in whatever web pages you would like, just as if you had created a new issue collector.

1. Log in to JIRA as a [project administrator](#) or a user with the **JIRA Administrators global permission**.
2. Access the relevant project's list of issue collectors ([above](#)).
3. In the Operation drop-down for the issue collector you would like to copy, select **Copy** to open the Add Issue Collector page.

4. All the information from the copied issue collector will be the same as the copied issue collector, with the exception of the name (which will be "Copy of " + *the original name of the copied issue collector*.)
5. Update the issue collector, as desired.
6. Click **Submit** to save your changes

Disabling or deleting an issue collector

1. Access the relevant project's list of issue collectors ([above](#)).
2. On the list of the project's issue collectors, click **Disable** or **Delete** to respectively disable or delete the associated issue collector.
 -  While an issue collector is disabled, its trigger tabs will still be visible on pages of your web site(s) to which the issue collector code has been added until a user refreshes the page. However, clicking these triggers results in a message indicating that the issue collector is currently out of action.

Known limitations

- Placing the Issue Collector plugin within an iframe will not close the prompt window automatically. This is a known limitation for the Issue Collector plugin, and has been tracked at

JRASERVER-41400 - Issue Collector Cannot Be Closed When Placed Inside an iframe
 GATHERING IMPACT

- If an anonymous user tries to create an issue collector in a project and there are required fields present in this project, these fields are not shown to the anonymous user. As a result, the user cannot create the collector. The workaround is to make the fields optional.

This is a known limitation for the Issue Collector plugin, and has been tracked at

JRASERVER-67094 - Unable to create Issue Collectors, because of required fields not visible to anonymous users
 GATHERING IMPACT

Advanced use of the JIRA issue collector

Customizing the JIRA issue collector

The JIRA issue collector can be used without any additional JavaScript beyond the single line generated in the issue collector administration screens in JIRA. However, you can also customize the JIRA issue collector in a number of different ways:

- Set up a custom trigger, so the feedback form launches from a different link or button than the packaged triggers provided.
- Set the default values of fields for your users, using JavaScript.
- Specify the values of fields on the issue, which are not shown in the feedback form.

This page assumes you are already familiar with [using the issue collector](#).

 **Warning:** The JavaScript exposed by the issue collector is not considered a stable API and may change with new JIRA releases.

On this page:

- [Customizing the JIRA issue collector](#)
- [Setting up a custom trigger](#)
- [Adding the custom trigger function manually](#)
- [Setting field values from JavaScript](#)
- [Embedding multiple issue collectors](#)
- [Embedding the issue collector](#)

Setting up a custom trigger

Configuring your collector to use a custom trigger

If you want to use a different trigger, or button, to launch the issue collector on your website, configure your issue collector as described below:

1. Add a new issue collector, or edit an existing issue collector.
2. Scroll down to section **Trigger** and select the option 'Custom'.
3. You don't need to set any **Trigger Text** as this will be overridden by your custom trigger.

Adding the issue collector script for a custom trigger

Creating and debugging custom scripts are outside of the scope of Atlassian Support. For assistance, please post any questions at <https://answers.atlassian.com>

The issue collector script generated by JIRA for adding a custom trigger is slightly different to the script generated for the standard triggers, because it includes the JavaScript function for the custom trigger.

Customization of the issue collector is done by creating/extending the global object **ATL_JQ_PAGE_PROPS**. This allows you to add a custom trigger, set default values for fields and more.

Note: In JIRA 5.1 (and version 1.1 of the Issue Collector plugin), the issue collector administrative interface let you define the custom trigger function UI, and you did not need to include it in the JavaScript on the page. In version 1.2 of the Issue Collector, the custom trigger JavaScript is a part of the generated JavaScript that you should copy and paste into your web page.

The code snippet below shows a sample HTML page with the generated issue collector JavaScript.

In the example below, we've added a simple button in HTML, and made that button launch the issue collector. This is done simply by replacing 'myCustomTrigger' in the generated JavaScript with the HTML id of the button ('feedback-button')

```
<head>
  <!-- We pasted the generated
code from the Issue Collector
here, after choosing a custom
trigger -->

  <!-- This is the script for
the issue collector feedback
form -->

  <script
type="text/javascript"
src="<JIRA
URL>/s/en_US-ydn9lh-418945332
/803/1088/1.2/_/download/batc
h/com.atlassian.jira.collecto
r.plugin.jira-issue-collecto
r-plugin:issuecollector/com.at
lassian.jira.collector.plugin
.jira-issue-collector-plugin:
issuecollector.js?collectorId
=d03d7bd1"></script>

  <!-- This is the script for
specifying the custom
trigger. We've replaced
'myCustomTrigger' with
'feedback-button' -->

  <script
type="text/javascript">
```

```
    window.ATL_JQ_PAGE_PROPS =
    {
      "triggerFunction":
function(showCollectorDialog)
    {
      //Requires that jQuery is
available!

jQuery("#feedback-button").cl
ick(function(e) {
    e.preventDefault();
    showCollectorDialog();
  });
  }
  };
</script>

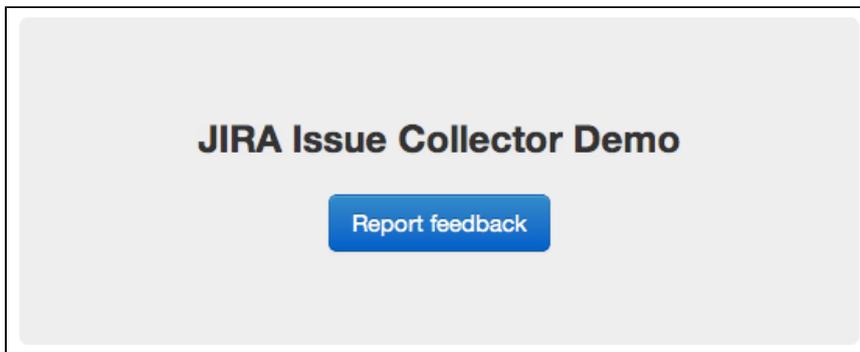
</head>

<body>

  <h2>JIRA Issue Collector
Demo</h2>
  <a href="#"
id="feedback-button"
class='btn btn-primary
btn-large'>Report
```

```
feedback</a>

</body>
```



Adding the custom trigger function manually

The custom trigger JavaScript will be included in the JavaScript generated by the issue collector. However, this section provides details on how you could do it without pasting in the additional lines of generated JavaScript.

To add a custom trigger, add the property **triggerFunction** in the global object **ATL_JQ_PAGE_PROPS**. **triggerFunction** needs to be defined as a function and takes one argument which is the function for displaying the issue collector.

You can invoke the issue collector from any element on your page by adding a click handler in **triggerFunction** as shown below. In this example, we will be calling the issue collector from our **#feedback-button** anchor tag defined in the above HTML markup. You can assign multiple triggers for the same issue collector by adding more click handlers.

```
window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {

    // ==== custom trigger function ====
    triggerFunction : function( showCollectorDialog ) {
        $('#feedback-button').on( 'click', function(e) {
            e.preventDefault();
            showCollectorDialog();
        });

        // add any other custom triggers for the issue collector here
    }

});
```

The **triggerFunction** will be invoked by the issue collector after the `$(document).ready()` phase.

Setting field values from JavaScript

Setting field values

The issue collector gives you the option to set field values for any of the fields on the issue type. This is done by adding the property **fieldValues** in the global object **ATL_JQ_PAGE_PROPS**. There are different methods for setting default values for different field types. The code samples below show a visual

representation of a field in JIRA and its relevant markup, and how to set a default value for that field type. Use a DOM inspection tool such as Firebug in the JIRA Issue Create Screen to extract the field names and values relevant to your issue collector. Please note that the Issue Collector is not supposed to be a replacement for the [JIRA REST API](#). If you require a more customized solution, make use of the JIRA REST API to create JIRA issues from external websites. The [JIRA Travel App](#) is a good example of how you can build a front end interface with JIRA as the back end.

Visible fields (setting default field values)

If you set the value of a field that is visible on the issue collector feedback form, the fields will already be filled in with that value when the form opens.

Hidden fields

There might be cases where you might want to set a field value without actually displaying the field on the issue collector. In this case, simply use the same method as above to set the field values via JavaScript. The fields will not be shown as they were not added in the form template but their values will still be present in issues created with the issue collector.

JavaScript for setting field values

Setting field values is done by specifying field name / value pairs within the "fieldValues" block of window.ATL_JQ_PAGE_PROPS. If you already have a custom trigger defined, you can simply add to the definition of window.ATL_JQ_PAGE_PROPS like the example below.

Note the names of the fields are always the names of the field in the JIRA Create Issue Screen, not any overridden names you may have provided in the issue collector form.

```

window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {

  // ==== custom trigger function ====
  triggerFunction : function( showCollectorDialog ) {
    $('#feedback-button').on( 'click', function(e) {
      e.preventDefault();
      showCollectorDialog();
    });
  },
  // ==== we add the code below to set the field values ====
  fieldValues: {
    summary : 'Feedback for new website designs',
    description : 'The font doesn\'t quite look right',
    priority : '2'
  }
});

```

Examples of how to set specific field types

Text field example

Setting the value for a text field, like the issue summary, is straightforward. Here's the markup for a text field like Summary in the issue collector (you do not need to add this, this is simply to show the representation that the issue collector contains):

```

<div class="field-group">
  ...
  <input class="text long-field" id="summary" name="summary"
  type="text" value="">
  ...
</div>

```

And here's how you set the value of the field in JavaScript:

```

fieldValues : {
  summary : 'This is the default summary value'
}

```

Select list example with issue priority

Setting the value for a select list field, such as the issue priority, requires a little more effort, because you need to know the HTML element id for the choice you want to select. Here's the markup for the Priority field in the issue collector (you do not need to add this, this is simply to show the representation that the issue collector contains):

```

<div class="field-group">
  ...
  <input id="priority-field" class="text aui-ss-field
  ajs-dirty-warning-exempt" autocomplete="off">
  ...
  <select class="select" id="priority" name="priority" style="display:
  none; " multiple="multiple">
    <option class="imagebacked"
  data-icon="/images/icons/priority_blocker.gif"
  value="1">Blocker</option>
    <option class="imagebacked"
  data-icon="/images/icons/priority_critical.gif"
  value="2">Critical</option>
    ...
  </select>
  ...
</div>

```

And here's how you set the value of the field in JavaScript:

```

fieldValues : {
  'priority' : '2'
}

```

Multi-select or checkboxes example

Setting the value for a multi-select (like the Browser field) or checkbox requires that you provide an array of values. Like the select list, you need to know the values to set, by looking at the markup on the Create Issue Screen.

```

<div class="field-group">
  ...
  <select class="select" id="customfield_10110" multiple="multiple"
name="customfield_10110" size="5">
    <option value="-1" selected="selected">None</option>
      <option value="10039">All Browsers</option>
    <option value="10037">Chrome</option>
    ...
  </select>
  ...
</div>

```

And here's how you set the value of the field in JavaScript: the field values must be set as an array of values, even if there is only one value.

```

fieldValues : {
  'customfield_10110' : [ '10039', '10037' ]
}

```

Custom fields

Setting a value for a custom field is exactly the same as any other field in JIRA. Since multiple custom fields can share the same name, custom fields will be referenced by "customfield_" + the Id of the custom field in JIRA. This ID can be seen in the HTML markup for the Create Issue Screen in JIRA, but can also be determined by looking at the URLs on the custom fields screen in JIRA administration. Here's what the JavaScript would look like for setting a custom field whose id in JIRA was 11111:

```

fieldValues : {
  'customfield_11111' : 'San Francisco'
}

```

Cascading selects

Setting a value for a cascading select is done in two steps - one for the parent value and one for the child. Below is an example of setting the value of a cascading select field.

```

fieldValues : {
  'customfield_12345' : 'Australia',
  'customfield_12345:1' : 'Sydney'
}

```

Special case fields

Environment field

By default, the issue collector puts user context such as the URL, User-Agent and screen resolution in the environment field. There might be cases where you wish to include more information in the environment field. In this case, you can add the property **environment** in the global object **ATL_JQ_PAGE_PROPS**. This allows you to add key value pairs that will appear on the environment field in the JIRA issue.

```
window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {
  // ==== custom trigger function ====
  triggerFunction : function( showIssueCollector ) {
    ...
  },
  // ==== default field values ====
  fieldValues : {
    ...
  },
  // ==== Special field config for environment ====
  environment : {
    'Custom env variable' : $('#build-no').text(),
    'Another env variable' : '#007'
  }
});
```

Restricted fields

Some fields that require a user to be logged into JIRA cannot be set through JavaScript. Assignee is an example of a field that cannot be set via JavaScript.

Dynamic functions

Environment and **fieldValues** properties can also be a function returning a JSON object that will be executed immediately when the collector trigger is shown (**not** just before opening the collector form). This might come in handy when you might wish to capture contextual information relevant to the user.

```

window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {
  // ==== custom trigger function ====
  triggerFunction : function( showIssueCollector ) {
    ...
  }
  // ==== Special field config for environment ====
  , environment : function() {

    var env_info = {};

    if ( window.ADDITIONAL_CUSTOM_CONTEXT ) {
      env_info[ 'Additional Context Information' ] =
window.ADDITIONAL_CUSTOM_CONTEXT;
    }

    return env_info;
  }
  // ==== default field values ====
  , fieldValues : function() {

    var values = {};

    var error_message = $(' .error_message ');
    if ( error_message.length !== 0 ) {
      // record error message from the page context rather than asking
the user to enter it
      values[ 'summary' ] = error_message.children( '.summary' ).text();
      values[ 'description' ] =
error_message.children( '.description' ).text();
    }

    return values;

  }
});

```

Embedding multiple issue collectors

If you want to have two different forms appear on the same web page, you will need to create two different issue collectors in JIRA. To set custom triggers, or set field values on those issue collectors requires a few changes to your page:

1. Include the generated JavaScript for both of your issue collectors in the page.
2. Find the id of each collector. This can be done one of two ways:
 - a. The parameter of the script is "collectorId=<8 character id>". That's the ID you want.
 - b. Go to the Issue Collector page in the Admin section and click on the Issue Collector you wish to embed. Copy the collectorId from the URL.

```

https://<JIRA_URL>/secure/ViewCollector!default.jsps?projectKey=<PROJ
ECT_KEY>&collectorId=<copy this part here>

```

Then, create separate namespaces for each of the issue collectors in the **ATL_JQ_PAGE_PROPS** object.

```
window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {
  '<collectorId_1>' : {
    triggerFunction:
      // define trigger function

    fieldValues: {
      // define field values here
    }
  },
  '<collectorId_2>' : {
    triggerFunction:
      // define trigger function
    fieldValues: {
      //define field values here
    }
  }
});
```

Embedding the issue collector

Embedding the issue collector in your Confluence Site

The issue collector can be embedded into Confluence using the [HTML Macro](#). Note that using the HTML Macro would require you to embed the issue collector code separately on each page.

The issue collector was previously embeddable in Confluence via a [User Macro](#), allowing you to create a re-usable issue collector macro that other Confluence users can embed into their pages. This option is currently unavailable due to a known bug:

CONFSERVER-26104 - Some JavaScripts are not executed if included in User Macro
GATHERING IMPACT

Embedding the issue collector is not currently supported in Confluence Cloud.

JIRA

The issue collector can be embedded in the announcement banner on a JIRA page by embedding the above script and HTML markup for your custom trigger in the [announcement banner configuration screen](#). If you wish to change the location of your custom trigger, this can be easily done via jQuery. The following snippet shows how you can add the custom trigger onto the footer of all JIRA pages.

You cannot embed an issue collector in your JIRA Cloud site since HTML markup is disabled for the announcement banner.

```

window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {
  // ==== custom trigger function ====
  triggerFunction : function( showIssueCollector ) {
    // button markup - relevant css can be added via the style
    attribute
    var feedbackButton = "<a id='feedback-button'>Got Feedback?</a>";
    // embed the button in the footer
    $('<code>.footer-link</code>').append(feedbackButton);

    $('<code>#feedback-button</code>').click(function(e) {
      ...
    });
  }
});

```

Please note that embedding the issue collector requires you to enable HTML markup for the announcement banner.

Full source code

This source code shows how to embed two different issue collectors on the same page with custom triggers.

```

<body>

  <h2>JIRA Issue Collector Demo</h2>
  <a href="#" id="feedback_button" class='btn btn-primary
  btn-large'>Report feedback</a><br />
  <a href="#" id="bug_button" class='btn btn-primary btn-large'>Report
  bug</a>

  <!-- JIRA Issue Collector - append this at the bottom of <code>body</code>
  -->
  <script type="text/javascript" src="https://<JIRA
  URL>/s/en_US-ydn9lh-418945332/803/1088/1.2/_/download/batch/com.atlas
  sian.jira.collector.plugin.jira-issue-collector-plugin:issuecollector
  /com.atlassian.jira.collector.plugin.jira-issue-collector-plugin:issu
  ecollector.js?collectorId=<collectorId_1>"></script>
  <script type="text/javascript" src="https://<JIRA
  URL>/s/en_US-ydn9lh-418945332/803/1088/1.2/_/download/batch/com.atlas
  sian.jira.collector.plugin.jira-issue-collector-plugin:issuecollector
  /com.atlassian.jira.collector.plugin.jira-issue-collector-plugin:issu
  ecollector.js?collectorId=<collectorId_2>"></script>

  <!-- We will customize JIRA in the following script tag -->

  <script type="text/javascript">
    // safely use jquery here since the issue collector will load it
    for you
    $(document).ready(function() {

      window.ATL_JQ_PAGE_PROPS = $.extend(window.ATL_JQ_PAGE_PROPS, {

        // ==== feedback collector ====
        '<code>collectorId_1</code>' : {

```

```

// === custom trigger function ===
  triggerFunction : function( showCollectorDialog ) {
    $('#feedback_button').click( function(e) {
      e.preventDefault();
      showCollectorDialog();
    });
  }

// === default and hidden field values ===
, fieldValues : {

  // default values
  summary : 'Feedback for new website designs'
  , description : 'The font doesn\'t quite look right'

  // hidden field value
  , priority : '2'

}

}

// ===== bug collector =====
, '<collectorId_2>' : {
  // === custom trigger function ===

  triggerFunction : function( showCollectorDialog ) {
    $('#bug_button').click( function(e) {
      e.preventDefault();
      showCollectorDialog();
    });
  }

  // === additional environment details ===
  , environment : function() {

    var env_info = {};

    if ( window.ADDITIONAL_CUSTOM_CONTEXT ) {
      env_info[ 'Additional Context Information' ] =
window.ADDITIONAL_CUSTOM_CONTEXT;
    }

    return env_info;
  }
  // === default field values ===
  , fieldValues : function() {

    var values = {};

    var error_message = $('#.error_message');
    if ( error_message.length !== 0 ) {

      // record error message from the page context rather than
      asking the user to enter it

```

```
        values[ 'summary' ] =
error_message.children('.summary').text();
        values[ 'description' ] =
error_message.children('.description').text();

    }

    return values;

}

}

});

});
```

```

</script>

</body>

```

Is localization of an issue collector possible?

You can create an issue collector 100% localized to the default language of your JIRA instance. Beyond that, complete localization of the issue collector is not possible.

The strings and text in the issue collector feedback form of the issue collector is a combination of:

1. The issue collector strings set by the JIRA Administrator
2. Either the default language setting for JIRA, or the language preference of the user if they are logged in to JIRA.
 - All users will see the names of the fields as they are set by the JIRA Administrator. These are not affected by the default language of JIRA, and are not affected by the default language of logged in JIRA users.
 - All users will see the field descriptions as they are set in the JIRA Administration UI.
 - For everything else:
 - **Anonymous** users will see everything else in the default JIRA language.
 - Logged in users will see everything else in the feedback form in the language specified by their JIRA profile.

Because of the above, you cannot create a single issue collector that will present itself entirely in the language of the end user.

However, if you want to create an issue collector that will present itself to anonymous users in the default language of your JIRA instance, you should:

1. Use the custom feedback template for the issue collector
2. Change the field labels in JIRA, and the labels for name and email, to the words you want to use in the default JIRA language.

The language setting of the browser will not impact the text in the feedback form.

Working with workflows

A JIRA workflow is a set of *statuses* and *transitions* that an issue moves through during its lifecycle and typically represents processes within your organization. There are default built-in workflows that cannot be edited; however, you can copy and use these workflows to create your own.

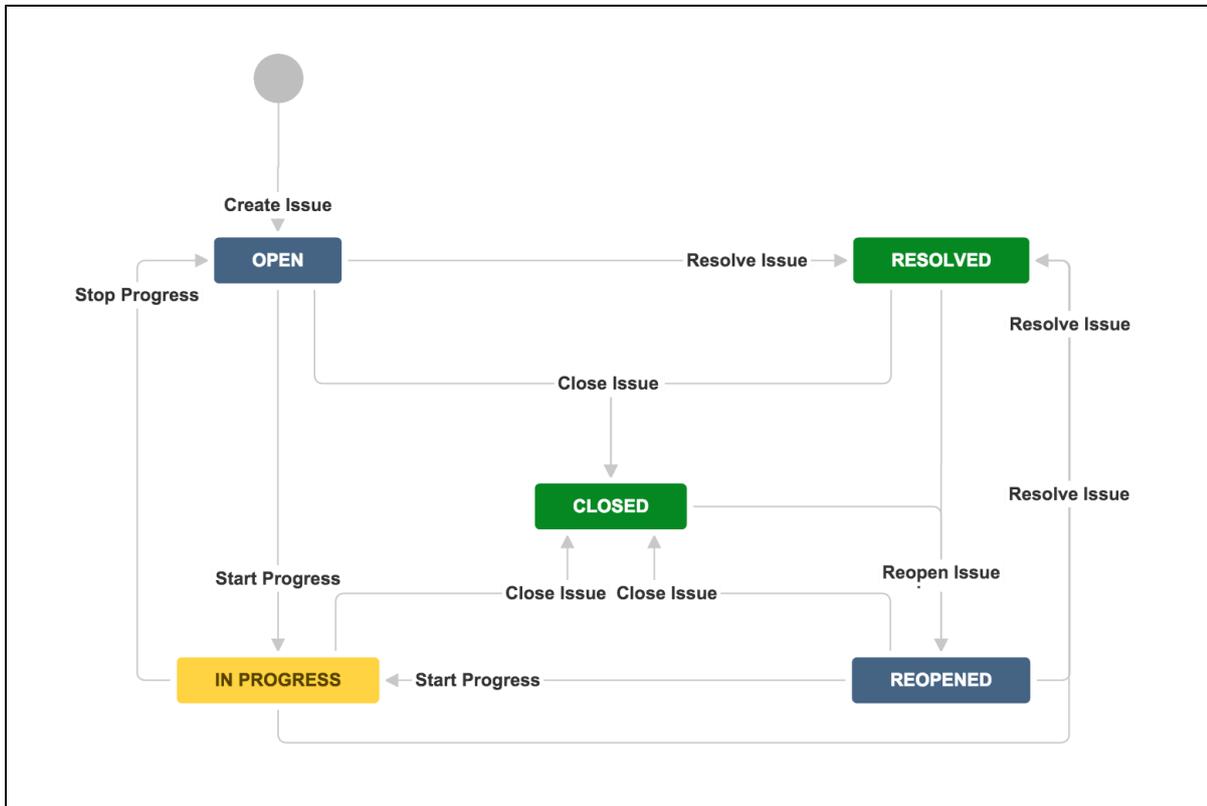
You can also create your own workflows from scratch, or import workflows from Atlassian Marketplace. Workflows can be associated with particular projects and, optionally, specific [issue types](#), by using a [workflow scheme](#).

You will need to log in as a user with the 'JIRA Administrators' [global permission](#) to access and manage workflows. If the workflow isn't the default workflow, or shared with another project, project administrators also have limited editing permission to the workflow.

On this page:

- [Statuses and transitions](#)
- [Active and inactive workflows](#)
- [Workflow designer](#)
- [Creating workflows](#)
- [Configuring a workflow](#)
- [Advanced workflow configuration](#)

Here's an example of a default workflow:



Statuses and transitions

A status represents the state of an issue at a specific point in your workflow. An issue can be in only one status at a given point in time. When defining a status, you can optionally specify *properties*.

A transition is a link between two statuses that enables an issue to move from one status to another. In order for an issue to move between two statuses, a transition must exist.

A transition is a *one-way* link, so if an issue needs to move back and forth between two statuses, two transitions need to be created. The available workflow transitions for an issue are listed on the View issue screen.

Active and inactive workflows

There are slight differences between editing an inactive and an active workflow. We place restrictions 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 use this workflow.

Workflow status	Description
Inactive workflow	An <i>inactive workflow</i> is a workflow that is not currently being used by any projects. Because there are no issues currently transitioning through an inactive workflow, you can edit the workflow's steps and transitions directly. For details on this, see Working in text mode .

Active workflow	<p>An <i>active workflow</i> is a workflow that is currently being used by one or more projects. When you edit an active workflow, JIRA first creates a draft of it, that you can then modify as you see fit. When you've finished, you can publish your draft and, optionally, save your original workflow as an inactive backup.</p> <p>The following limitations apply when editing the draft for an active workflow:</p> <ul style="list-style-type: none"> ▼ Editing limitations... <ul style="list-style-type: none"> • It is not possible to edit the workflow name (only the description) if a workflow is active. • Workflow statuses cannot be deleted. • The step ID cannot be changed. See Cannot Add Transitions or Delete Steps in Draft Workflows. <p>To make any of the modifications listed above, you need to copy the workflow (see Creating a workflow), modify the copy, and then activate it.</p>
-----------------	--

Workflow designer

The workflow designer is a graphical tool that allows you to see the layout of your workflow and to create and edit a workflow's steps and transitions. You will need to log in as a user with the 'JIRA System Administrators' [global permission](#) to access the functionality described below.

With the workflow designer, you can:

- Manage status and transitions: add, click and drag, or select to edit properties ([Workflow properties](#)) to rename, or delete (from the workflow but not JIRA).
- Add a global transition that allows every other status in the workflow to transition to the selected status. Select **Allow all statuses to transition to this one** in the properties panel for the transition.
- Change the screen that a transition uses. See [Working in text mode](#) for details.
- Configure advanced transition options, such as triggers, conditions, validators, and post functions. See the [Advanced workflow configuration](#) page.
- ▼ [Expand for workflow designer tips...](#)
 - Statuses are *global objects*. Changing the name of a status on one workflow also changes it in *all workflows that use that status*.
 - Hover over a transition or a status to see the relevant transition labels.
 - Zoom the diagram with your mouse wheel. Pan the diagram by clicking and holding the mouse while on white space, then moving your mouse across the diagram.
 - You cannot clone transitions in the workflow designer.
 - You cannot create annotations in the workflow designer.
 - You cannot directly set the `issue.editable` property. To do this, simply add the `issue.editable` property to the [status properties](#).
 - The workflow designer will automatically validate your workflow and highlight any statuses that have no incoming or outgoing transitions. The workflow validator will also highlight all transitions that have an invalid permission condition that you don't have available in JIRA. The validator is particularly useful if you import workflows, or deal with complex workflows.

Creating workflows

There are a few ways you can start a new workflow. These include cloning an existing workflow, creating a new workflow, and importing a workflow.

Copy an existing workflow

1. Choose  **> Issues**.
2. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.

Name	Last modified	Assigned Schemes	Steps	Operations
jira (Read-only System Workflow) DEFAULT The default JIRA workflow.		<ul style="list-style-type: none"> Product Development Workflow Scheme 	5	View · Copy
JIRA Service Desk IT Support Workflow generated for Project TD This JIRA Service Desk IT Support Workflow was generated for Project TD	09/Jan/15 System Administrator	<ul style="list-style-type: none"> JIRA Service Desk IT Support Workflow Scheme generated for Project TD 	3	View · Edit · Copy

- Copy an existing workflow using the **Copy** link in the Operations column (shown above). Enter a name and description and select the **Copy** button.
- Customize it by adding or editing steps and transitions.

When you have finished customizing your workflow, see [Managing your workflows](#) for details on how to use it with a JIRA project.

Create a new workflow

For advanced administrators

- Click **Workflows** in the left-hand nav panel, then **Add Workflow** at the top of the screen.
- Enter a name and description for your workflow. Click **Add**.
The workflow opens in edit mode, and contains a step called **Open** and an incoming transition called **Create**.
- Continue with your workflow customizations, by adding and editing steps and transitions.

Import a workflow

Please see the documentation on [Importing workflows](#).

Configuring a workflow

Editing a project's workflow

Whenever you create a new JIRA project, your project automatically uses the default workflow scheme. The scheme associates all available issue types in the project with the JIRA system workflow. Since neither the JIRA system workflow nor the default workflow scheme are editable, JIRA creates an editable copy of the system workflow and workflow scheme for your project.

- Choose



> **Projects**, and select the relevant project.

- On the Administration page for the project, click **Workflows**.
- Click the 'edit' icon at the top-right of the box, and JIRA automatically does the following:
 - Creates a draft copy of the system workflow named '*Your Project Name Workflow (Draft)*'.
 - Creates a new workflow scheme for the workflow named '*Your Project Name Workflow Scheme*'.
 - Associates any existing issues in your project with the new workflow.
- You can now edit your draft workflow. Click on a status or transition to see editing options in the panel that appears.
- When you are finished, click **Publish**. The dialog allows you to publish your draft and, optionally, save your original workflow as an inactive backup.

▼ [Expand for performance notes about modifying workflows...](#)

- The number of issues impacts the speed when configuring a workflow - for small numbers of issues, this process is relatively quick, however if you have many (e.g. thousands of) existing issues in your JIRA project, this process may take some time.
- Once this process begins, *it cannot be paused or canceled*. Please avoid editing or transitioning any issues within your project while this process is taking place.

Setting the resolution field

In JIRA, an issue is either open or closed, based on the value of its 'Resolution' field — not its 'Status' field.

- An issue is open if its resolution field has not been set.
- An issue is 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). Therefore, 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. See [Working in text mode](#) for details on this.

Renaming workflow transition buttons

If you copied the system workflow and you wish to rename the workflow transition buttons on the View Issue page, you must delete the following properties from all transitions in the copied workflow:

- `jira.i18n.title`
- `jira.i18n.description`

Otherwise, the default names (i.e. values of these properties) will persist. Read more about [transition properties](#).

Working in text mode

Text mode is an advanced way of working with workflows, and it shows the difference between steps and statuses. In text mode, you work directly with steps. For details, see [Working in text mode](#).

Advanced workflow configuration

See the documentation on [Advanced workflow configuration](#).

Managing your workflows

Workflows need to be activated to use them in JIRA. Activating a workflow is the process of mapping the workflow to a workflow scheme, and then associating the workflow scheme with a project. To configure a workflow scheme, see [Configuring workflow schemes](#).

A workflow scheme defines a set of associations – or mappings – between a workflow and an issue type. Workflow schemes are associated with a project and make it possible to use a different workflow for every combination of project and issue type.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- [Activating a workflow](#)
- [Managing workflows for projects](#)
- [Exporting your workflow](#)
- [Importing workflows](#)

Activating a workflow

Active workflows are those that are currently being used, while inactive workflows are those that are not associated with any workflow schemes, or are associated with workflow schemes that are not associated with any projects. Active workflow schemes are also those associated with projects, while inactive workflow schemes are not.

1. Create a workflow scheme or find an existing workflow scheme. See [Configuring workflow schemes](#) for instructions.
2. Configure the workflow scheme to use your workflow. See [Configuring workflow schemes](#) for instructions.

Associate your workflow scheme with a project, as described in the [Associating a workflow scheme with a project](#) section below.

Managing workflows for projects

You can manage your workflows by associating workflow schemes, importing, exporting, uploading, and sharing.

Associating a workflow scheme with a project

You can associate a single workflow scheme with more than one project, although only one workflow scheme can be associated with a given project. The [issue type scheme](#) associated with a project defines the issue types that are available to that project. If an issue type is not defined in the project's issue type scheme, its workflow is not used.

1. Choose



> **Projects**, and select the relevant project. The Project Summary page is displayed.

2. Click **Workflows** on the left of the Project Summary page (you can also click the **More** link in the Workflows section in the middle of the screen). This is the current workflow scheme used by the project.
3. Click the **Switch Scheme** link to display the Associate Workflow Scheme to Project page.
4. Select the relevant workflow scheme from the Scheme list and click the **Associate** button to begin the migration process.
Each issue has to be in a valid status. The valid statuses for an issue are defined by its workflow. This means that when changing a workflow, you may need to tell JIRA the status for specific issues after the change.
5. A screen displays that indicates the progress of migrating all the project's issues to the updated scheme's workflows. **Acknowledge** to finish the process.

Disassociating a workflow scheme from a project

A JIRA project must always be associated with a workflow scheme, since all issues must move through a workflow, even if that workflow only consists of a single *Create Issue* transition. By default all JIRA projects with unmodified workflows use JIRA's system workflow. *Disassociating* a workflow scheme re-associates your project's workflow with JIRA's default workflow scheme.

1. Follow the instructions in [Associating a workflow scheme with a project](#) above.
2. When selecting the workflow scheme from the Scheme list, select the **Default** workflow scheme
3. Click the **Associate** button, and follow the wizard, which guides you through migrating all of the project's issues.

Exporting your workflow

The workflow sharing feature allows you to share your team's workflow with other teams in your organization on different JIRA instances, or external parties in other organizations via the [Atlassian Marketplace](#). This feature allows you to easily share and use workflows that other people have published, or to move a workflow from staging to production in your own organization. If you wish to share your JIRA Workflow with another instance of JIRA or upload it to the Atlassian Marketplace, you first need to download it.

1. Choose



> **Issues**.

2. Find the workflow you wish to share by clicking on the Workflows section in the left-hand panel.
3. Click **View** or **Edit** under the Operations column.
4. Select **Export > As Workflow** and click **Next** to continue.
5. In the Add Notes field, add any special configuration notes; for example, information about plugins that should be installed. JIRA auto-populates these notes for you when it discards parts of your workflow (for example, plugins, post functions, conditions, validators).
6. Click **Export** and select a download location. Ensure the location is publicly accessible.

Uploading to Atlassian Marketplace

To share your workflow with other JIRA users, upload it to the Atlassian Marketplace.

1. Create an account on [Atlassian Marketplace](#), or log in and choose **Manage Add-ons** (more info: [Step-](#)

- by-step Paid-via-Atlassian Listing).
2. Click **Create new add-on**.
 3. Choose **My add-on is not directly installable** (ensure that 'Add-on Type' is listed as 'Not a Plugin'). You will need to host the workflow on your own servers, and add information about where the workflow export can be accessed in the Binary URL textbox. This should be the location you specified in step 6 of the prior instruction set.
 4. Fill out the submission form, be sure to note the following:
 - a. The Summary field contains the information that will be displayed to users searching the Marketplace.
 - b. The Category for your workflow must be Workflow Bundles. Choosing Workflow Bundles ensures other JIRA users will have visibility to your workflow.
 - c. The Add-on Key must be unique, as it uniquely identifies your application; it will become the application URL.

You don't have to complete the form in one session. You can save your form and come back to it later. Once you accept the [Atlassian Marketplace Vendor Agreement](#), the system submits your add-on for review by Atlassian's Developer Relations team.

Importing workflows

Custom fields in workflow imports

If your workflow contains custom fields that are disabled, the workflow importer will not create these fields unless they are enabled before importing. You will receive a warning about this. To fix this, you need to enable the missing custom fields before proceeding with the import.

1. Click on the highlighted **Custom Field Types & Searchers** plugin in the displayed warning. This opens the plugin in a new window and scrolls to the right place to make the necessary changes.
2. Click to expand the list of enabled modules.
3. Find the modules that are disabled and enable them.

After enabling the corresponding modules of the Custom Field Types & Searchers plugin, return to the summary page and proceed. You may need to refresh the page first. For information on installing add-ons, see [Viewing installed apps](#).

Importing from Atlassian Marketplace

This procedure covers importing a workflow from Atlassian Marketplace.

1. Choose  **> Issues**.
2. Click on the **Workflows** section in the left-hand panel.
3. Select **Import > Import Workflow** in the top right of the screen.
4. The **From Atlassian Marketplace** option should be selected by default.
5. Find the workflow you want and click the **Select** button.
6. Follow steps 5 through 8 of the **Importing from a local instance** procedure.

Importing from a local instance

This procedure covers importing a workflow from a local instance.  You must be logged in as System Administrator to perform this function.

1. Click on the **Workflows** section in the left-hand panel.
2. Select **Import > Import Workflow**.
3. Select a workflow from your computer to upload, and then click **Next**.
4. JIRA automatically generates a workflow name, but you can change this if you like. Click **Next**.
5. Next, you are presented with a screen that details your workflow statuses, as shown below. You can map the steps of the workflow to your existing workflow statuses or create new statuses at this point. When you are finished, click **Next** to continue.
6. At the Preview of Import screen, click **Import** at the bottom of this screen to accept the changes and import the workflow.
7. Your workflow is imported and you are presented with a screen with additional configuration details.

Click **Done** to exit this process.

i All custom fields will have brand new custom fields created. This is regardless of a custom field of the same name / type already existing. See:

JRASERVER-37358 - Workflow import creates duplicate custom fields

GATHERING INTEREST

for the request

to improve this.

Configuring workflow schemes

A workflow scheme defines a set of associations – or mappings – between a workflow and an issue type. Workflow schemes are associated with a project and make it possible to use a different workflow for every combination of project and issue type.

By default, projects use JIRA's [system workflow](#). The default workflow scheme:

- Associates JIRA's system workflow *jira* with all issue types (available to the JIRA project).
- Appears as the default workflow scheme for your selected project type.

In addition, you can share an existing project's workflow scheme when you are creating a new project by selecting **Create with shared configuration** in the [Project Creation Wizard](#). This allows you to reuse your existing schemes without having to recreate them for new projects. Keep in mind that changing shared workflow schemes will affect all projects that are using that theme.

On this page:

- [Adding a workflow scheme](#)
- [Configuring workflows for a workflow scheme](#)
- [Editing, copying, and deleting workflow schemes](#)

This page describes how to configure workflows and issue type workflow associations in the scheme.

i To associate a workflow scheme with a project (part of activating a workflow), see [Managing your workflows](#).

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Adding a workflow scheme

1. Choose



> **Issues**. Select **Workflow Schemes** to open the Workflow Schemes page.

2. Click the **Add Workflow Scheme** button.
3. Enter the name and description of the new workflow scheme.
4. Click the **Add** button. The new workflow scheme is created.
5. Follow the instructions in [Configuring workflows for a workflow scheme](#) below.

Configuring workflows for a workflow scheme

If your scheme is associated with a project, follow the instructions in [Configuring a workflow scheme associated with a project](#). Otherwise, follow the instructions in [Configuring a workflow scheme outside of a project](#).

Configuring a workflow scheme associated with a project

JIRA's default workflow scheme cannot be modified. If you attempt to modify it, a copy of the scheme is created with the name of the project you are administering. You cannot configure a workflow scheme shared by multiple projects using this method; follow the instructions in [Configuring a workflow scheme outside of a project](#) instead.

1. Choose



> Projects.

2. Select a project from the displayed list.
3. Click **Workflows** on the left of the Project Summary page. The Workflows page is displayed, indicating the current workflow scheme used by the project. Configure the workflow scheme using the table below.
4. At the Publish Workflows screen, click **Associate** to begin the migration process. Each issue has to be in a valid status. The valid statuses for an issue are defined by its workflow. This means that when changing a workflow, you may need to tell JIRA the status for specific issues after the change.
5. A screen displays that indicates the progress of migrating all the project's issues to the updated scheme's workflows.
6. Click **Acknowledge** to finish the process. A message displays letting you know that 'your workflows have been published.'

Configure the issue types for the workflow scheme as desired. This is not the same as editing the workflow (clicking the **Edit** button in the workflow diagram at the center of your screen). If you do that, you will be asked to publish your *draft workflow scheme*.

What do you want to do?	Instructions
Add a workflow to the scheme	<ol style="list-style-type: none"> 1. Click Add Workflow, and select Import From Bundle or Add Existing. After selecting Import From Bundle, you can select a workflow From Atlassian Marketplace. See Sharing your workflow for more information. 2. Select the desired workflow and issue types.
Edit a workflow	Hover over the desired workflow and click the Edit button. See Working with workflows for further instructions. The Edit button only displays if you have the edit permission.
Remove a workflow from the scheme	Click the cross icon under Operations to remove the workflow from the scheme.
Change the issue types associated with a workflow	<ol style="list-style-type: none"> 1. Click the Assign link under Issue Types for the desired workflow. 2. Select the desired issue types in the dialog that appears. 3. Click Finish.
View the text-based representation of a workflow	Hover over the desired workflow, and click the View as Text link.
Change the workflow scheme associated with the project	Click the Switch Scheme button next to the scheme name. See Managing your workflows for further instructions.

Configuring a workflow scheme outside of a project

You can use this procedure to edit any workflow scheme in the system, including those shared by multiple projects. The workflow scheme can be either active or inactive.

- If your workflow scheme is associated with a project, you may want to follow the [instructions above](#) instead. When a workflow scheme is used by more than one project, you must use this configuration method.
- When a workflow scheme is active, it creates a *draft workflow scheme* when you edit it.

1. Choose



> Issues. Select **Workflow Schemes** to open the Workflow Schemes page.

2. Click the **Edit** link under the Operations column for the desired workflow.
3. Edit your workflow scheme, as described in the table below.
4. If your workflow is active, you need to publish it to make your changes active.

What do you want to do?	Instructions
Add a workflow to the scheme	<ol style="list-style-type: none"> 1. Click Add Workflow, and select Import From Bundle or Add Existing. After selecting Import From Bundle, you can select a workflow From Atlassian Marketplace. See Sharing your workflow for more information. 2. Select the desired workflow and issue types.
Remove a workflow from the scheme	Click the Remove link in the Operations column.
Change the issue types associated with a workflow	<ol style="list-style-type: none"> 1. Click the Assign link under Issue Types for the desired workflow. 2. Select the desired issue types in the dialog that appears. 3. Click Finish.
View a representation of a workflow	Click either the text or diagram link next to the Workflow name.
Remove an issue type from the scheme	Click the x next to the name of the issue type to remove it.

Editing, copying, and deleting workflow schemes

Choose



> **Issues**. Select **Workflow Schemes** to open the Workflow Schemes page.

Operation	Instructions
Edit the name and description of a workflow scheme	Click the Edit link. Use inline edit mode – click in the associated field – to update the name and description.
Copy a workflow scheme	Click the Copy link to create a workflow scheme with the prefix "Copy of (name of current workflow)" and placed in the inactive workflow schemes.
Delete a workflow scheme	Click the Delete link and confirm the deletion. You cannot delete an active workflow scheme. You must first disassociate it from all projects.

Sharing your workflow

The new Workflow Sharing feature allows you to share your team's workflow with other teams in your organization on different JIRA instances, or external parties in other organizations via the [Atlassian Marketplace](#). This feature allows you to easily share and use workflows that other people have published, or to move a workflow from staging to production in your own organization.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- Exporting your workflow
- Uploading to Atlassian Marketplace
- Importing from Atlassian Marketplace
- Importing from a local instance
- Custom fields in workflow imports

Exporting your workflow

If you wish to share your JIRA Workflow with another instance of JIRA or upload it to the Atlassian Marketplace, you first need to download it. Follow this procedure.

1. Choose



> **Issues.**

2. Find the workflow you wish to share by clicking on the Workflows section in the left-hand panel.
3. Click **View** or **Edit** under the Operations column.
4. Select **Export > As Workflow**.
5. Click **Next** to continue.
6. In the Add Notes field, add any special configuration notes; for example, information about plugins that should be installed. JIRA auto-populates these notes for you when it discards parts of your workflow (for example, plugins, post functions, conditions, validators).
7. Click **Export** and select a download location. Ensure the location is publicly accessible.

Uploading to Atlassian Marketplace

To share your workflow with other JIRA users, upload it to the Atlassian Marketplace.

1. Create an account on [Atlassian Marketplace](#).
2. Log in to the Atlassian Marketplace and choose **Manage Add-ons**. See this page for more details: [Step-by-step Paid-via-Atlassian Listing](#).
3. Click **Create new add-on**.
4. Choose **My add-on is not directly installable**.
5. Ensure 'Add-on Type' is listed as 'Not a Plugin.'
6. You will need to host the workflow on your own servers, and add information about where the workflow export can be accessed in the Binary URL textbox. This should be the location you specified in step 7 of the prior instruction set.
7. When you fill out the submission form, be sure to note the following:
 - a. The Summary field contains the information that will be displayed to users searching the Marketplace.
 - b. The Category for your workflow must be Workflow Bundles.
 -  Choosing Workflow Bundles ensures other JIRA users will have visibility to your workflow.
 - c. The Add-on Key must be unique.

 This is something that uniquely identifies your application; it will become the application URL.

You don't have to complete the form in one session. You can save your form and come back to it later. Once you accept the [Atlassian Marketplace Vendor Agreement](#), the system submits your add-on for review by Atlassian's Developer Relations team.

Importing from Atlassian Marketplace

1. Choose



> **Issues.**

2. Click on the Workflows section in the left-hand panel.
3. Select **Import > Import Workflow** in the top right of the screen.
4. The **From Atlassian Marketplace** option should be selected by default.
5. Find the workflow you want and click the **Select** button.
6. Follow steps 5 through 8 of the 'Importing from a local instance' procedure.

Importing from a local instance

This procedure covers importing a workflow from a local instance. For importing from Marketplace, see the procedure above, **Importing from Atlassian Marketplace.**  You must be logged in as System Administrator to perform this function.

1. Click on the Workflows section in the left-hand panel.
2. Select **Import > Import Workflow.**
3. Select a workflow from your computer to upload, and then click **Next.**
4. JIRA automatically generates a workflow name, but you can change this if you like. Click **Next.**
5. Next, you are presented with a screen that details your workflow statuses, as shown below. You can map the steps of the workflow to your existing workflow statuses or create new statuses at this point. When you are finished, click **Next** to continue.
6. You will be presented with a screen that presents a summary of the workflow changes, as shown below. Click **Import** at the bottom of this screen to accept these changes and import the workflow.
7. Your workflow is imported and you are presented with a screen with additional configuration details. Click **Done** to exit this process.

 All custom fields will have brand new custom fields created. This is regardless of a custom field of the same name / type already existing. See:

JRASERVER-37358 - Workflow import creates duplicate custom fields
GATHERING INTEREST

for the request

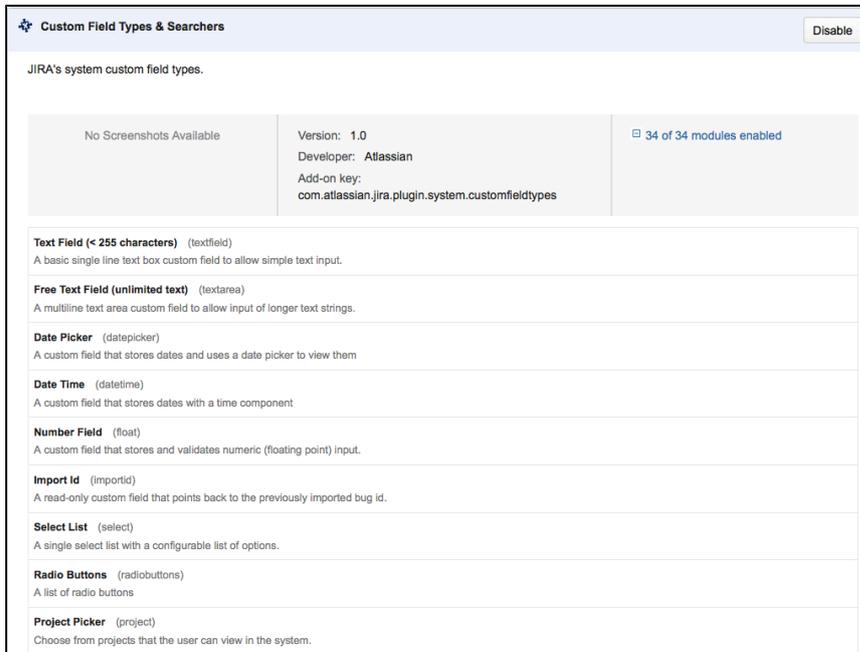
to improve this.

Custom fields in workflow imports

If the workflow that you are importing contains custom fields that are disabled, the workflow importer will not create these fields unless they are enabled before importing.

You will receive a warning about this. To fix this, you need to enable the missing custom fields before proceeding with the import.

1. Click on the highlighted Custom Field Types & Searchers plugin in the displayed warning. This opens the plugin in a new window and scrolls to the right place to make the necessary changes:



2. Click to expand the list of enabled modules.
3. Find the modules that are disabled and enable them.

After enabling the corresponding modules of the Custom Field Types & Searchers plugin, return to the summary page and proceed. You may need to refresh the page first.

For information on installing add-ons, see [Viewing installed apps](#).

Advanced workflow configuration

This page describes configuring transitions in JIRA workflows. For information about the basics of workflows – see [Working with workflow](#).

As a JIRA administrator, you can control the following aspects of a transition's behavior:

- **Triggers** – transition JIRA issues when certain events occur in a connected development tool, such as Atlassian's [Bitbucket](#) or [Stash](#).
- **Conditions** – check that a transition should be performed by the user.
- **Validators** – check that any input to the transition (for example, by a user) is valid, *before* the transition is performed.
- **Post functions** – carry out additional processing, *after* a transition is performed.
- **Properties** – are key-value pairs that can be used to further customize transitions.

Also on this page:

- [Customize how transitions appear](#)
- [Global transitions](#)

Triggers

JIRA administrators can configure triggers in JIRA workflows that respond to events in your linked development tools. This allows you to set up your development tools and JIRA workflows so that, for example, when a developer creates a branch to start work on an issue in Atlassian's [Bitbucket](#) or [Stash](#), the issue will automatically be transitioned from 'Open' to 'In progress'.

i If you haven't set up a trigger before or you want to learn about triggers in more detail, see our guide on triggers here: [Configuring workflow triggers](#). The guide also shows you how to configure a workflow with triggers, similar to this sample development workflow: [Development Workflow with Triggers](#) (from Atlassian Marketplace).

Configure triggers

To see, or to set, triggers for a transition, edit the workflow that contains the transition, select the transition, then click **Triggers** in the properties panel for the transition.

▼ [Not sure about that? Click here to see how...](#)

To add a trigger to a transition:

1. Log in as a user with the 'JIRA Administrators' [global permission](#).
2. Choose 
 - > **Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click **Edit** for the workflow that has the transition you wish to change.
4. In the Workflow Designer, select the transition.
5. Click **Triggers** in the properties panel to show the triggers configured for the transition.
6. Click **Add trigger** on the **Triggers** tab to configure a trigger.

Conditions

Conditions control whether a transition should be executed by the user. As examples, conditions can be used to:

- allow only the reporter to execute a transition.
- allow only users with a certain permission to execute a transition.
- allow execution only if code has, or has not, been committed against this issue.

If a condition fails, the user will not see the transition button on the 'View issue' page, and so will not be able to execute the transition.

Conditions cannot validate input parameters gathered from the user on the transition's screen – you need a [validator](#) to do this.

The following sections describe:

- [Adding a condition](#)
- [Grouping conditions](#)

Adding a condition

To add a condition to a transition, edit the workflow that contains the transition, select the transition, then click **Conditions** in the properties panel for the transition.

▼ [Not sure about that? Click here to see how...](#)

To add a condition to a transition:

1. Log in as a user with the 'JIRA Administrators' [global permission](#).
2. Choose 
 - > **Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click **Edit** for the workflow that has the transition you wish to change.
4. In the Workflow Designer, select the transition:
5. Click **Conditions** in the properties panel.

On the **Conditions** tab, you can see any conditions that have already been set.

When you click **Add condition**, you can choose from the available conditions, and set any necessary parameters for the condition. Additional conditions may be available from installed plugins. or you can create your own conditions using the [plugin system](#); see the [Workflow Plugin Modules](#) for details.

Note that you can also edit the transition in ['text' mode](#).

Grouping conditions

You can construct complex conditions by grouping and nesting conditions. Change any condition into a group by clicking the 'Add grouped condition' icon for the condition. Now you can add further conditions to this new group, as described [above](#).

You can toggle the logic for how the conditions in a group are applied between **All** and **Any**.

Validators

Validators check that any input made to the transition is valid *before* the transition is performed. Input can include that gathered from the user on the transition's screen.

If a validator fails, the issue does not progress to the destination status of the transition, and the transition's [post functions](#) are not executed.

Adding a validator

To add a validator to a transition, edit the workflow that contains the transition, select the transition, then click **Validators** in the properties panel for the transition.

▼ [Not sure about that? Click here to see how...](#)

To add a validator to a transition:

1. Log in as a user with the 'JIRA Administrators' [global permission](#).
2. Choose 
 - > **Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click **Edit** for the workflow that has the transition you wish to change.
4. In the Workflow Designer, select the transition:
5. Click **Validators** in the properties panel.

On the **Validators** tab, you can see any validators that have already been set.

When you click **Add validator** you can choose from the available validators and set any necessary parameters for the validator.

Note that you can also edit the transition in ['text' mode](#).

Post functions

Post functions carry out any additional processing required after a transition is executed, such as:

- updating an issue's fields
- generating change history for an issue
- adding a comment to an issue
- generating an event to trigger email notifications

The following sections describe:

- [Essential post functions](#)
- [Optional post functions](#)
- [Using post functions with the initial transition](#)
- [Using a post function to set a field](#)
- [Using a post function to send HipChat notifications](#)
- [Using a post function to send email notifications](#)

Essential post functions

Every JIRA transition has the following essential post functions, which are performed in this order:

1. Set issue status to the linked status of the destination workflow status.
2. Add a comment to an issue if one is entered during a transition.
3. Update change history for an issue and store the issue in the database.
4. Reindex an issue to keep indices in sync with the database.
5. Fire an event that can be processed by the listeners.

These essential post functions cannot be deleted from a transition or reordered. However, you can insert other (optional) post functions between them.

Optional post functions

JIRA includes several optional post functions that can be added to transitions.

▼ [Click to see a list of optional post functions...](#)

Optional post function	Description
Assign to Current User	Assigns the issue to the user who is executing the transition. This post function is ignored unless the user has the Assignable User permission . Create a condition to give the logged-in user 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.
Create Perforce Job Function	Creates a Perforce Job (if required) after completing the workflow transition.
Notify HipChat	Sends a notification to one or more HipChat rooms. See Using a post function to send HipChat notifications for more information.
Trigger a Webhook	Triggers the specified webhook after completing the workflow transition. When you add this post function, you will be asked to specify a webhook. This webhook must already be defined in JIRA (see Managing webhooks).
Update Issue Field	Updates one of the issue's fields to a given value. Fields that can be updated include: <ul style="list-style-type: none"> • Assignee • Description • Environment • Priority • Resolution • Summary • Original Estimate • Remaining Estimate <p>This post function cannot update custom fields and must be positioned after the other optional post functions.</p>

Additional post functions may be available from installed plugins. or you can create your own post functions using the [plugin system](#); see the [Workflow Plugin Modules](#) for details.

Adding a post function

To add a post function to a transition, edit the workflow that contains the transition, select the transition, then click **Post functions** in the properties panel for the transition.

▼ [Not sure about that? Click here to see how...](#)

To add a post function to a transition:

1. Log in as a user with the 'JIRA Administrators' [global permission](#).
2. Choose  **> Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click **Edit** for the workflow that has the transition you wish to change.
4. In the Workflow Designer, select the transition:
5. Click **Post functions** in the properties panel.

On the **Post functions** tab, you can see any post functions that have already been set. When you click **Add**

post function you can choose from the available post functions, and set any necessary parameters. Options for editing or deleting a post function, and for changing the execution order, are at the right of the tab (hover there to see them).

Note that you can also edit the transition in **'text' mode**.

Using post functions with the initial transition

You can add post functions to a workflow's initial transition when you need to perform processing tasks – such as setting a particular field's value – when an issue is created. The initial transition is called 'Create' (if you created a blank workflow) or 'Create Issue' (if you copied the system workflow).

JIRA includes the following **essential post functions** that are specific to a workflow's initial transition and that are performed in this order:

1. Create the issue.
2. Fire an event that can be processed by the listeners.

The following optional post functions are available specifically for the initial transition:

Optional post function (initial transition only)	Description
Create Comment	Adds a comment to an issue if one is entered during a transition.
Update Issue Status	Sets the issue's status to the linked status of the destination workflow status.
Store Issue	Stores updates to an issue (no change history is created).

Additionally, the standard **optional post functions** can also be added to an initial transition,

Optional post functions added to the Create transition must be placed *before* the 'Create the issue originally' post function.

If you wish, you can configure the initial status for your workflow to go to a different initial transition. See [Configuring the initial status](#) for details.

Notes

If you need to set the 'Resolution' field when creating an issue, add the 'Update Issue Field' post function *after* the 'Create the issue' post function and *after that*, use the 'Store Issue' post function. The 'Store Issue' post function is useful for setting the Resolution field during issue creation.

However, only use the Store Issue post function where necessary, since it:

- does not generate change history
- is unable to persist fields that have a one-to-many relationship with the issue (for example, 'Version' or 'Component')

Using a post function to set a field

You can use the 'Update Issue Field' post function to set the value of an issue's field after a particular transition is executed.

For example, you might want a transition that moves the issue to a *closed* status to automatically set the 'Resolution' field.

Example: Using a post function to set the Resolution field:

1. Edit the workflow that has the transition, and drag from status to another to create a new transition.
2. Select either **None** or a screen that does not contain the **Resolution** field:

3. Add a new post function of type 'Update Issue Field' and:
 - a. Select **Resolution** from the **Issue Field** list.
 - b. Select a suitable resolution from the **Field Value** list.

To create a transition that clears the **Resolution** field, follow the same steps above for adding an 'Update Issue Field' post function to your transition. However, select **None** from the **Field Value** list.

The list of post functions for this transition includes 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 cleared, as specified in these post functions.

Using a post function to send HipChat notifications

You can use a 'Notify HipChat' post function to send a notification to one or more HipChat rooms whenever an issue passes through a transition with this post function. You can also add a JQL query to the 'Notify Hipchat' post function to filter for the issues that will trigger the HipChat notification.

To send HipChat notifications:

1. Create or edit your transition.
2. Add a new post function of type 'Notify HipChat'.
3. On the 'Add Parameters to Function' page:
 - a. Optionally, specify a JQL query. Only issues that match the query will send notifications. Leave this field empty to send notifications to *all* issues that pass through this transition.
 - b. Select the HipChat rooms you want to link with your workflow transition.

Using a post function to send email notifications

Use the 'Fire an event that can be processed by the listeners' post function to fire the 'Generic Event', which is a built-in **JIRA event** that can be used to trigger the sending of **email notifications** after a particular transition is executed.

Alternatively, you could fire a **custom event** that you've 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.

i The fired event is also propagated to all registered **listeners**.

Example: Using a post function to fire the Generic Event to send email notifications:

1. Create or edit your transition.
2. Click the transition's **Post Functions** tab and edit the 'Fire an event that can be processed by the listeners' post function.
3. Select **Generic Event** from the list of events.

Transition properties

Properties are key-value pairs that can be used to further customize transitions. For example, transition properties can help to extend a copied system workflow to allow language translations.

To view and edit the properties of a transition:

1. Select a transition in the diagram.
2. Click **Properties** in the Properties panel.
3. Either:
 - Add a new property to the transition.
 - Delete a property, by clicking the icon to the right of the property.

Important

It is not possible to edit a transition's properties on this page. To change any property's key or value (or both), you must first delete the property you wish to change and add the new updated property.

Note that you can also edit the transition in 'text' mode.

It is possible to implement restrictions on transitions using transition properties. For more information, see [Workflow properties](#).

Customize how transitions appear

When viewing an issue, most of the operations and workflow transitions are available from a row of buttons at the top of the issue.

To change the number of transition buttons from the default of two:

By default, the first two transitions appear as separate buttons in the set of transition buttons. Additional transitions appear in the **Workflow** menu. The order in which these buttons appear is based on the order defined in the system workflow.

1. Shutdown JIRA.
2. Edit the `jira-config.properties` file in your [JIRA application home directory](#). See [Making changes to the jira-config.properties file](#) for more information.
3. Change the value of 'X' in the `ops.bar.group.size.opsbar-transitions = X` property of this file to be the number of transition buttons required *before* the **Workflow** menu.
 - If this property does not exist in your `jira-config.properties` file, add it. Otherwise, a default value of 2 is assumed.
4. Save the updated `jira-config.properties` file.
5. Restart JIRA.

To change the order of transition buttons:

To change the order of transition buttons, including additional transitions in the **Workflow** menu, add the property key `opsbar-sequence` to each [workflow transition](#) that you wish to reorder. Each `opsbar-sequence` property key requires a property value that defines the order of the transition action on issue views.

1. Go to the transition's properties, as described in [Transition properties](#) above.
2. Type `opsbar-sequence` into the **Property Key** field, under 'Add New Property'.
3. Type a value in the **Property Value** field. The value must be a positive integer (starting at '0'); it defines the order of the transition buttons on issue views.

Consider using a sequence of `opsbar-sequence` property values like 10, 20, 30... to allow new transitions to be easily added later.
4. Click **Add**.

• Adding the `opsbar-sequence` property to a workflow transition does not change the order of these transitions in the workflow in Text edit mode. The addition of this property only affects the order of transitions on the **View issue** page.

Global transitions

Global transitions allow any status in a workflow to transition to a particular status.

You can add a global transition:

- When creating a new status (adding an existing status) – check the **Add global transition to status** option.
- By selecting a status and checking **Allow all statuses to transition to this one** in the properties panel for the status.



To create two global transitions that point to the same destination step:

1. From the workflow designer, create the first global transition as normal by selecting a step and checking "Allow all statuses to transition to this one"
2. Create the second global transition on any *other* step that does not currently have a global transition pointing to it
3. Then from text editor, select the second global transition that you created
4. Click on the 'Edit' button and change the 'Destination Step' to the same step that you selected for your first global transition, and then click 'Update'

Working in text mode

Text mode is an advanced way of working with workflows, and it shows the difference between steps and statuses. In text mode, you work directly with steps.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission and start from the Workflows page.

On this page:

- [Basic procedures](#)
- [Advanced procedures](#)

To open a workflow in text mode

1. Choose  > **Issues**.
2. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.

3. Select **Edit** to open the workflow for editing.
4. Select the **Text** button to edit in text mode. A list of existing steps that comprise the workflow and each step's Linked Status and Outgoing Transitions (under Transitions (id)), is shown.
5. Follow the relevant procedure below to edit the workflow.

Note that some workflow elements cannot be edited in text mode, such as global transitions. If you cannot change a workflow element, try editing in Diagram mode.

Basic procedures

Editing a step

Click the following link of any step:

- **Add Transition** — to add an Outgoing Transition to that step.
- **Delete Transitions** — to delete one or more Outgoing Transitions of that step.
- **Edit** — to edit the step's Step Name or Linked Status.
- **View Properties** — to view and edit the step's Properties.
- **Delete Step** — only available if the step has no Incoming Transitions.

Adding a step

The Add New Step form appears below the list of steps when you are editing an inactive workflow.

To add a new step to a workflow:

1. In the Step Name field, type a short name for the step.
2. In the Linked Status field, select the status that corresponds to this step.
 - **i** Each status can only correspond to one step in each workflow.
3. Click the **Add** button. Your new step appears in your workflow's list of steps in Text edit mode.

i If you do not see Add New Step, this means that all available statuses defined in your JIRA installation have been used in your workflow and you need to [define a new status](#).

Deleting a step

A step can only be deleted if it has no incoming transitions.

Click the **Delete Step** link that corresponds to the relevant step.

i This link is not displayed if the step has no incoming transitions or if it only has incoming **Global Transitions**.

Adding a transition

1. Identify the step from which your new transition will originate, and click the **Add Transition** link next to the step.
2. In the Transition Name field, type a short name for the transition.
 - **i** This name will be shown to users on the relevant transition button on the View issue page.
3. *(Optional)* In the Description field, type a short description of the purpose of the transition.
4. In the Destination Step field, choose the step to which issues will move when this transition is executed.
5. In the Transition View field, select either:
 - No view for transition — choose this if you do not 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).
 - 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 or any other screen you have created. If no existing screen is suitable, you may wish to create a new screen for the transition.

Editing or deleting a transition

1. In the Transitions (id) column, click the link of the **Outgoing Transition** of the step you wish to edit. The Transition page is displayed.
2. From this point, you can:
 - Click the buttons at the top of the page to edit or delete the transition.
Note: You will only be able to delete a transition if this step has at least one outgoing transition

- indicated in the Workflow Browser section. In the image above, this is not the case.
- Click **View Properties** to edit the transition's properties. See [Advanced workflow configuration](#) for details.
- Add a new condition, validator, or post function. See [Advanced workflow configuration](#) for details.

Advanced procedures

Preventing issues from being edited

You can use a workflow step's properties to prevent issues from being edited in a particular workflow step. For example, in a copied system workflow, **Closed** issues cannot be edited, even by users who have the Edit Issue [project permission](#).

Note:

- Issues that cannot be edited cannot be updated using bulk edit.
- You can only edit the properties of a workflow's step if that workflow is editable (i.e. if that workflow is either [inactive](#) or a draft of an active workflow).

To stop issues from being editable in a particular workflow step or to set any property of a step:

1. Click the **View Properties** link that corresponds to the relevant step.
2. In the **Property Key** field, type: `jira.issue.editable` (or any other **Property Key** you wish to add).
3. In the **Property Value** field, type: `false` (or any other **Property Value** you wish to add).
4. Click the **Add** button.

Note:

- It is not possible to edit a step's properties on this page. To change any property's key or value, you must first delete the property you wish to change and then add the new, updated property.
- It is possible to implement restrictions on steps using step properties. For more information, see [Workflow properties](#).

Using a screen with a transition

When a user clicks a particular transition, a screen can be used to gather input from the user before the 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 moves the issue to a Closed status. To do this:

1. Create or edit your transition.
2. Select the **Resolve Issue Screen** in the **Transition View** field.
3. Click **Add** when you are finished editing the workflow transition. You will be back on the **Text** view screen of the project's workflow.

See also:

- [Working with workflows](#)
- [Advanced workflow configuration](#)

Adding a custom event

JIRA uses an event-listener mechanism to alert the system that something has happened, and to perform 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.

- [System events](#)
- [Custom events](#)
- [Configuring notifications for a custom event](#)

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).

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** — Custom events are used to generate an email notification (or invoke a listener) from a particular workflow transition's post function. You can add and 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 [Managing your workflows](#)).

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 Working with workflows).

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.

Configuring notifications for a custom event

Custom events are most commonly used to generate notifications for custom workflow transitions. For example, your organization 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.

Adding a custom event

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose  **> System**. Select **Advanced > Events** to open the View Events page.
3. In the Add New Event form at the bottom of the page, add a name and description for the custom event.
4. In the Template field, select the default email template to be associated with the event.
5. Click the **Add** button.

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.

Configuring the notification scheme to send mail

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose 

- > **System**. Select **Advanced > Events** to open the View Events page.
- 3. Select the notification scheme to edit, by clicking the notification scheme's name or its **Notifications** link (under Operations).
- 4. Add the recipients for the custom event as required. See [Creating a notification scheme](#) for more information.

Configuring a post function to fire the custom event

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



- > **Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
- 3. Navigate to workflow transition post function screen to be edited. See [Working with workflows](#) and [Advanced workflow configuration](#) for more information.
- 4. Update the post function to fire the custom event.
- 5. Activate or associate the workflow (and scheme) with the appropriate project. See [Managing your workflows](#) for more information.

Configuring the initial status

Use this procedure to configure the initial status for your workflow. You can start off with an active workflow, which you can then switch to draft mode, or any other workflow in your system.

1. Click **Open** under the Step Name column to view or edit a step's properties:

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Created	Start Progress (4) >> In Progress Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	Add Transition · Delete Transitions · Edit · View Properties

2. Click the **Create Issue** incoming transition:

Workflows / Susan's Simple Project Workflow (Draft)

Step: Open Edit Delete View Properties ?

Add outgoing transition Delete outgoing transitions

Create Issue (selected) → OPEN → Start Progress

Stop Progress

Close issue post testing

Resolve Issue

Close Issue

Note: If you happen to be in an active workflow, which you cannot edit, you will be asked to switch to a draft workflow to continue.

3. Click **Edit** to set the new destination step:

⚠ You are editing a draft workflow. [View the original workflow](#) or [publish this draft](#).

Workflows / Susan's Simple Project Workflow (Draft)

Transition: Create Issue Edit View Properties ?

Create Issue → OPEN

This is the initial transition in the workflow.

Screen: None - Initial transition does not have a view.

Validators 1 Post Functions 2

The transition requires the following criteria to be valid [Add validator](#)

Only users with **Create Issues** permission can execute this transition.

4. Select a new **Destination Step**, and then click **Update** to save it:

5. When a new issue is created, it will go straight to the **In Progress** step.

Configuring workflow triggers

You must have JIRA Software to start using workflow triggers.

Triggers are a powerful tool for keeping your JIRA issues synchronized with the information in your development tools (FishEye/Crucible, Bitbucket and GitHub). Instead of relying upon developers to manually update the status of issues after committing code, completing reviews, creating branches, etc, you can configure triggers in your workflow to automatically transition issues when these events occur in your development tools. For example, you could configure a trigger to automatically transition an issue from 'To Do' to 'In Progress' when a branch is created.

On this page:

- [Before you begin](#)
- [Guide: Setting up triggers](#)
- [Understanding triggers](#)
- [Troubleshooting](#)

This page will help you get started using triggers. We will show you how to set up triggers in a workflow and demonstrate how an automatic transition works. We will also provide some guidelines on how to best configure a trigger and help you troubleshoot your triggers.

Before you begin

Before you can start using triggers, you need to connect your development tools to JIRA Software. At a minimum, you will need a JIRA Server instance, plus at least one of the following:

- Bitbucket Server (all [current versions](#))
- FishEye/Crucible (all [current versions](#))
- GitHub Enterprise 11.10.290 (or later)
- Bitbucket
- GitHub

For instructions on how to connect these tools to JIRA, see [Integrating with development tools](#). This page also includes details on other functionality you can enable by connecting the various development tools Atlassian offer.

Guide: Setting up triggers

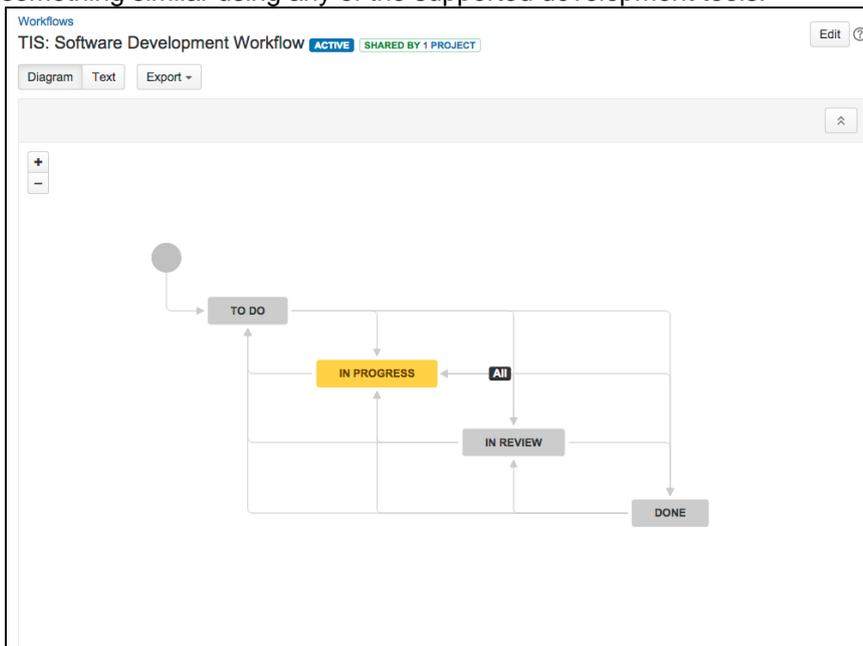
In this example, you will be configuring a JIRA workflow with triggers. By the end of this section, you will have an understanding of how to configure triggers and what a typical development workflow with triggers looks like.

- [Introduction](#)
- [Step 1. Create/Edit a workflow](#)
- [Step 2. Add a trigger to a transition](#)
- [Step 3. Test the trigger](#)
- [Step 4. Add the rest of the triggers](#)

Introduction

The screenshot and table below show a workflow and triggers similar to what you will be configuring. They reflect the typical interactions between JIRA and development tools in a software development lifecycle. JIRA

Software, Bitbucket Server and FishEye/Crucible (3.5.2) are used for this example, but you can configure something similar using any of the supported development tools.



Transition	Triggers
Start progress (<i>To Do</i> <i>In Progress</i>)	Branch created (Bitbucket Server) Commit created (Bitbucket Server)
Start review (<i>In Progress</i> <i>In Review</i>)	Pull request created (Bitbucket Server) Pull request reopened ((Bitbucket Server) Review started (Crucible)
Restart progress (<i>In Review</i> <i>In Progress</i>)	Pull request declined (Bitbucket Server) Review rejected (Crucible) Review abandoned (Crucible)
Done (<i>In Review</i> <i>Done</i>)	Pull request merged (Bitbucket Server) Review closed (Crucible)

Step 1. Create/Edit a workflow

The easiest way to create a software development workflow is to [create a new project](#), choosing a relevant project type. This will set up your new project with the software development workflow, which is identical to the one shown above.

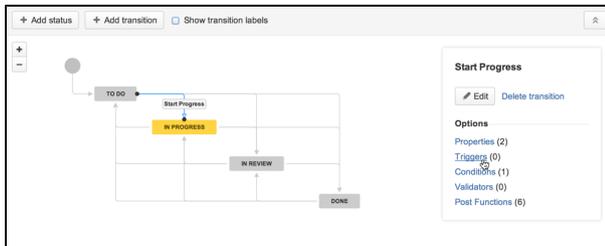
If you already have a similar workflow, navigate to it and edit it: JIRA administration console > **Issues** > **Workflows** > **Edit**

Step 2. Add a trigger to a transition

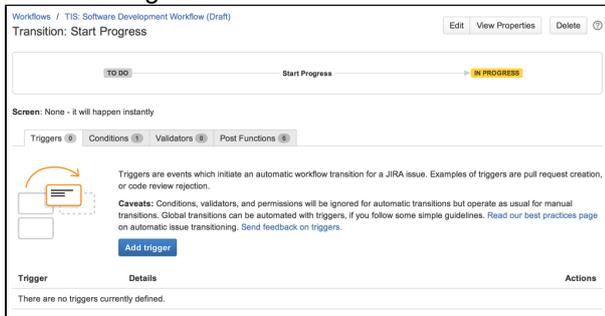
We'll start by adding a 'Commit created' trigger to the 'Start progress' transition. Ensure that you are editing (not viewing) the workflow.

1. Select the **Start progress** transition in the workflow, i.e. the line from 'To Do' to 'In Progress'. A panel will display on the right, showing the details of the transition.

Related topic: [Why you shouldn't configure triggers on global transitions](#)



2. Click **Triggers** in the panel. The 'Transition: Start Progress' screen will display with the 'Triggers' tab showing.

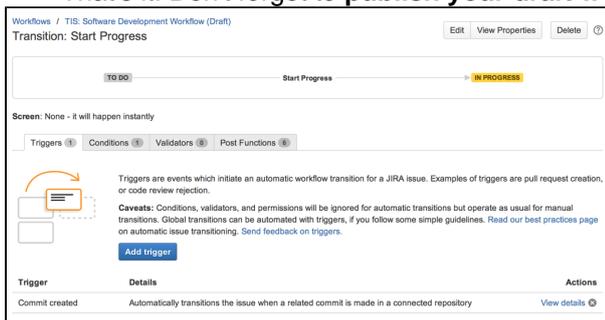


3. Click **Add trigger**, then select **Commit created** in the dialog that appears. A diagnostics window will display — you'll notice that the trigger will be added for all development tools that JIRA is connected to.

Related topic: [How to enable different events for triggers](#)

4. Click **Add trigger** to add the trigger. It will appear in a list at the bottom of the 'Triggers' tab. You can check whether it is working by clicking **View Details**.

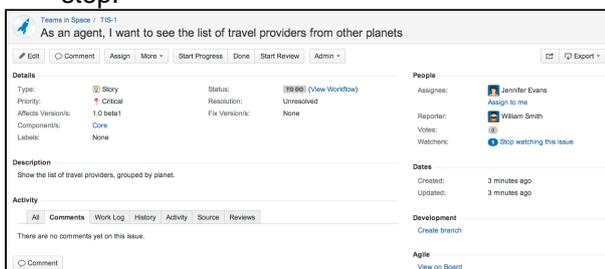
That's it! Don't forget to **publish your draft workflow**.



Step 3. Test the trigger

Now that you have added the 'Commit created' trigger to the 'Start progress' transition, let's test it by making a commit.

1. Create an issue in your JIRA project. This project needs to be using the workflow that you just edited. The status of your new issue should be 'To Do'. Take note of the issue key, as you'll need it for the next step.



2. Commit some code to your Bitbucket repository. You can commit anything, however you will need to include the issue key in your commit message.

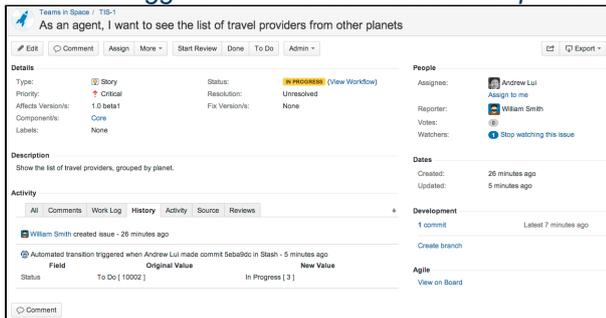
In this example, the issue key is TIS-1, which is referenced in the commit message shown in the screenshot.

Related topic: [Referencing a JIRA issue in a commit, branch, pull request, or review](#)

```
alui:tis-core alui$ git commit -am "TIS-1 Initial commit"
[master (root-commit) 274323b] TIS-1 Initial commit
 1 file changed, 7 insertions(+)
 create mode 100644 provider-list.html
alui:tis-core alui$ git push origin master
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To http://admin@localhost:7990/scm/tis/tis-core.git
 * [new branch]      master -> master
alui:tis-core alui$
```

3. Check your issue in JIRA again. The status should have changed from 'To Do' to 'In Progress'. If you click the **History** tab or **Activity** tab, you can see the automatic transition that changed the issue's status.

*Related topics: [How the user is mapped from the development tool to JIRA](#);
[Event handling and event limits](#);
[How triggers relate to other workflow operations/constraints](#)*



Step 4. Add the rest of the triggers

Now that you've added and tested a trigger, follow the same process to add the rest of the triggers in the [list above](#).

Don't want to set all of this up? Good news! You can download a similar workflow (pre-configured with triggers) from the Atlassian Marketplace: **download 'Development Workflow with Triggers'**.

Congratulations! You have now set up a workflow with triggers.

- If you are having problems configuring your trigger or getting it working, check the [Troubleshooting section](#) below.
- If you want to learn more about how triggers work, see the [Understanding triggers section](#) below.

Understanding triggers

The following topics explain how triggers work in more detail, so you can use them more effectively.

Trigger events

Events (e.g. Commit created) are made available for triggers by integrating JIRA with particular development tools. The table below lists the events that are enabled for each development tool.

Dev tool	Bitbucket, GitHub, GitHub Enterprise	Crucible	FishEye
Events	<ul style="list-style-type: none"> • Pull request created • Pull request merged • Pull request declined (Bitbucket only) • Pull request reopened (Bitbucket Server only) • Commit created • Branch created 	<ul style="list-style-type: none"> • Review started • Submitted for approval • Review rejected • Review abandoned • Review closed • Review summarized 	<ul style="list-style-type: none"> • Commit created • Branch created

There is a known issue where the 'Branch created' event isn't supported for GitHub, which is being tracked under

JWSERVER-14473 - Implement 'Create Branch' feature in DVCS connector plugin for Github integration

RESOLVED

— please keep this in mind when configuring trigger events.

Triggers and global transitions

We recommend that you *do not configure triggers for global transitions*, unless you are confident that you understand exactly how the trigger will affect the behavior of the issue.

A global transition allows any status in a workflow to transition to a particular status. This is represented in the workflow viewer/editor by a black **All** lozenge pointing to the status that the global transition targets. For more information about global transitions, see [Advanced workflow configuration](#).

Configuring triggers for global transitions can often result in an issue unexpectedly transitioning to the target status for the global transition. For example, consider if you configured a 'Commit created' trigger for the global transition to the 'In Progress' status. Committing code can happen at many stages during an issue's lifecycle (e.g. writing the initial code, changing code after a review, etc). This could result in the issue incorrectly transitioning to 'In Progress' out of a number of statuses, like 'In Review' or 'Done'.

Tip: If you do use global transitions in your workflow, you will probably have multiple transitions into a status. This means that users will have multiple workflow options on an issue (e.g. both 'Start Progress' and 'In Progress'). To hide options, add the 'Hide transition from user' condition to the relevant transitions.

Referencing a JIRA issue in a commit, branch, pull request, or review

The table below describes how to reference a JIRA issue in a commit, branch, pull request, or review, so that these events will trigger transitions for the issue (provided that you have set up triggers on the transitions).

Event	Instructions
Create commit	<p>Include the issue key in the commit message.</p> <p>For example, a commit message like this "TIS-1 Initial commit" will automatically transition the TIS-1 issue from 'To Do' to 'In Progress'.</p>
Create branch	<p>Include the issue key in the branch name, when you create the branch.</p> <p>For example, if you name your branch "TIS-2-feature", it will automatically transition the TIS-2 issue from 'To Do' to 'In Progress'.</p>
Create/Reopen/Decline Merge pull request	<p>Ensure that the pull request includes commits that reference the issue (in their commit messages).</p> <p>For example, if you create a pull request that has "TIS-3" in the title, it will automatically transition the "TIS-3" issue from 'In Progress' to 'In Review'. If you reopen, decline or merge the pull request, it will also transition the "TIS-3" issue accordingly.</p>
Start/Reject/Abandon/Close review	<p>Include the issue key in the review title, when you create the review.</p> <p>For example, if you name your review "TIS-4 New story" and start the review, it will automatically transition the TIS-4 issue from 'In Progress' to 'In Review'. If you reject, abandon or close the review, it will also transition the "TIS-4" issue accordingly.</p>

User mapping from the development tools to JIRA

The following process describes how a development tool user is mapped to a JIRA user for workflow triggers. It applies to all events, however each development tool uses a different email address and username for the mapping (see the bullet point following the process description below).

- Process: The user initiating the event in the development tool is mapped to a JIRA user by matching the email address, then the username, i.e.
 - *Single JIRA user with a matching email address* — Transition the issue as the JIRA user.
 - *No JIRA users with a matching email address* — Transition the issue as an anonymous user.

- *Multiple users with a matching email address in JIRA* — Try to find a matching username in that group of users. If there is a JIRA user with a matching username, transition the issue as the JIRA user. If there is no matching username, transition the issue as an anonymous user.
- Email address and username used for user mapping:

▼ [Stash](#)

Event(s)	Email address and username used for user mapping
All pull request events	The Bitbucket Server email address and username of the user who actioned the pull request.
Commit created	The email address associated with the commit and the Bitbucket Server username that the email address maps to. If the email address does not map to a username, the authors "name" from the commit will be used.
Branch created	The Bitbucket Server email address and username of the authenticated user that pushed the branch to Bitbucket Server.

▼ [FishEye/Crucible](#)

Event(s)	Email address and username used for user mapping
Commit created	The email address associated with the commit and the FishEye username that the email address maps to. If the email address does not map to a username, the authors "name" from the commit will be used.
Branch created	This event is not mapped to a JIRA user. This means that the issue will be transitioned as an anonymous user.
All review events	The Crucible email address and username of the authenticated user that actioned the review.

▼ [Bitbucket](#)

Event(s)	Email address and username used for user mapping
All pull request events	The Bitbucket email address and username of the user who actioned the pull request. Note, the Bitbucket user needs to have made at least one commit (with that email address configured for their profile), otherwise the pull request cannot be mapped to a JIRA user. This means that the issue will be transitioned as an anonymous user.
Commit created	Email address associated with the commit and the Bitbucket username that the email address maps to. If the email address does not map to a username, the authors "name" from the commit will be used.
Branch created	This event is not mapped to a JIRA user. This means that the issue will be transitioned as an anonymous user.

▼ [GitHub / GitHub Enterprise](#)

Event(s)	Email address and username used for user mapping
Pull request created / Pull request merged	GitHub email address and username of the user who actioned the pull request. Note, the GitHub user needs to have made at least one commit (with that email address configured for their profile), otherwise the pull request cannot be mapped to a JIRA user. This means that the issue will be transitioned as an anonymous user.

Commit created	Email address associated with the commit and the GitHub username that the email address maps to. If the email address does not map to a username, the authors "name" from the commit will be used.
Branch created	This event is not mapped to a JIRA user. This means that the issue will be transitioned as an anonymous user.

Event handling and event limits

In most cases, the processing of events from your development tools into automatic issue transitions should be seamless. However, sometimes there may be delays in issues transitioning or issues not transitioning at all, due to how events are handled or event limits.

- Event handling — Events are handled differently depending on whether the development tool connects to JIRA via the DVCS connector or an application link. This can affect whether events are delayed or lost when JIRA is unavailable:
 - ▼ [Bitbucket and GitHub](#)

Events from Bitbucket and GitHub are processed via the DVCS connector in JIRA. The DVCS connector processes events from Bitbucket and GitHub via two synchronization mechanisms: a webhook-triggered synchronization and a scheduled synchronization.

 - Webhook-triggered synchronization: the DVCS connector uses webhooks in Bitbucket and GitHub to post data to JIRA when an event occurs. This is the standard mechanism for processing events, which means that issues should be automatically transitioned almost immediately after a Bitbucket/GitHub event.
 - Scheduled synchronization: if JIRA cannot be contacted when a Bitbucket/GitHub event occurs, the event is stored by the DVCS connector and sent at the next scheduled synchronization (every 60 minutes, by default). This is a backup mechanism in case the webhook-triggered synchronization fails.
 - ▼ [Stash and FishEye/Crucible](#)

Events from Bitbucket Server and FishEye/Crucible are processed via the application link. However, Bitbucket Server and FishEye/Crucible are responsible for ensuring that events are sent, and they send them once at the time that the event occurs. This means that if JIRA is unavailable when the events are sent, the events will be lost.
- Event limits — Event limits are imposed on all of the development tools so that JIRA is not overloaded with too many events. Any events sent after the event limit is exceeded are lost. Event limits for each development tool are listed below:
 - ▼ [Bitbucket and GitHub](#)
 - Webhook-triggered synchronization: 10 branches; 100 commits
 - Scheduled synchronization: 600 branches (sync interval in minutes x 10); 6000 commits (sync interval in minutes x 100)

The event limits for scheduled synchronizations can be less than 600 branches and 6000 commits, if the synchronization interval is reduced, but never greater.
 - ▼ [Stash](#)

10 branches; 100 commits per synchronization

A further constraint that applies on top of the 10 branches and 100 commits limits is a 100,000 issue changed event limit. For example, if 100 commits each reference more than 1000 issue keys, the issue changed limit would be exceeded.
 - ▼ [FishEye/Crucible](#)

6000 events per synchronization

How triggers relate to other workflow operations/constraints

When a transition is triggered automatically, it ignores any conditions, validators or permissions configured on the transition.

However, post functions are still executed. You need to be careful that if your post function requires a user, that your transition will not be executed by an anonymous user (see [user mapping section](#) above).

Troubleshooting

If you are having problems setting up a trigger or getting a trigger to work, follow the steps below to troubleshoot your problem.

- 1. [Use the trigger diagnostics](#)
- 2. [Check for common problems](#)
- 3. [Get help](#)

1. Use the trigger diagnostics

Your first step in troubleshooting a trigger is to check the diagnostics for it in JIRA. The diagnostics can tell you if there is a problem with the connection to your development tools or whether an issue did not automatically transition as expected.

1. Navigate to the JIRA administration console > **Issues** > **Workflows** > Find your workflow and click **View (Operations column)**
2. In **Text** mode (not **Diagram** mode), click the desired transition.
3. On the transition screen (**Triggers** tab will be showing), click **View details** for the desired trigger to show the diagnostics information.
 - The 'Trigger sources' section lists problems related to the integration between JIRA and your development tools. For example, whether you have the correct type of authentication configured.
 - The 'Transition failures' section lists issues that have failed to automatically transition despite the trigger firing. For example, an anonymous user was mapped to the transition but the transition has a post function that requires a non-anonymous user.

2. Check for common problems

If you cannot resolve your problem with the information from the trigger diagnostics, check the list of common problems below for possible causes and solutions.

I cannot add a trigger to a transition:

▼ [Possible causes...](#)

Cause	Solution
JIRA or your development tools are not the correct version	Install/Upgrade to the correct version. <i>You must have JIRA 6.3.3+ and one of the following development tools to enable workflow triggers: Bitbucket Server (Stash 3.2.0+), FishEye/Crucible 3.5.2+, Bitbucket, GitHub</i>
Your development tools are not connected to JIRA correctly	Check the configuration of your connection: <ul style="list-style-type: none"> • JIRA + Bitbucket Server/FishEye/Crucible: You need to configure a two-way application link using Oauth with 2LO and 3LO. • JIRA + Bitbucket/GitHub: You need to configure the DVCS connector correctly. For more details, see Integrating with development tools .
The trigger that you are trying to add has already been added to the transition	Do nothing. <i>All triggers are unique per transition, that is, you can only add a trigger to a transition once.</i>

The issue does not transition:

▼ [Possible causes...](#)

Cause	Solution
Your project is not using the workflow that has been configured with triggers	Navigate to your project's summary > Administration > Workflows , and check that your project is using the workflow that you have configured with triggers.

You have not saved your workflow changes where the triggers were added	Navigate to the workflow that you added triggers to. Check that it has been published by viewing the workflow transitions and confirming that your triggers are present.
JIRA cannot be reached by your DVCS	<p>Wait an hour. If it still cannot be reached after an hour, check that the connection to your DVCS is configured correctly, see Integrating with development tools.</p> <p><i>If triggers are not configured or JIRA is not reachable from Bitbucket/GitHub, then the delay might be up to one hour, as there is still an hourly synchronization of commits/branches/pull requests happening regardless of the triggers configuration. For more information, see the Event handling and event limits section above.</i></p>
Your DVCS repository is not linked to the synchronized DVCS account	<p>Navigate to the JIRA administration console > Add-ons > DVCS Accounts and enable your repository.</p> <p><i>If you have not configured Bitbucket or GitHub to autolink new repositories, you may have repositories that are not enabled (i.e. linked to your DVCS account). This means that events from the unlinked repository will not be sent to JIRA, hence the issue will not transition automatically, even if you have configured a trigger.</i></p>
Your commits are too old	<p>Only commits less than 21 days old will cause a transition. This is to prevent bulk uploads from causing bulk transitions.</p> <p><i>If you want to work around this, you can change the 21 day constraint by editing the jira-config.properties file (in your JIRA home directory) and adding the following property:</i></p> <pre>jira.devstatus.commitcreated.age.timeout=P2D</pre> <p><i>where P2D is an example ISO-8601 duration representing 2 days.</i></p>
The operation is not permitted for anonymous users	<p>Check that each user in your development tools maps to a JIRA user.</p> <p><i>Certain issue operations will throw exceptions when the transition is performed by an anonymous user. These are:</i></p> <ul style="list-style-type: none"> • The CreateIssue event (this probably relates to 'Create' or 'Create Issue' transition in your workflow) • Post functions that assume a user is performing the transition <p><i>A triggered transition is performed by an anonymous user if the event in the development tool cannot be mapped to a JIRA user. For more information, see the section on user mapping above.</i></p>
The maximum number of automatic transitions permitted for an issue has been exceeded	<p>Check that your workflow transitions do not end in an infinite loop.</p> <p><i>By default, only 50 automatic transitions are permitted per issue. This is to prevent issues from becoming stuck in infinite loops. If your workflow actually requires more than 50 automatic transitions per issue, you can override this constraint by editing the jira-config.properties file (in your JIRA home directory) and adding/updating the following property:</i></p> <pre>jira.automatic.transitioning.issue.limit</pre>

Automatic issue transition events are incorrectly suppressed by the development tool	<p>Change the repository/project settings to allow events to be sent.</p> <p><i>You may have configured Bitbucket Server (Stash 3.3 - 3.5) or FishEye (3.5+) repositories to suppress events sent to JIRA for workflow triggers, if duplicate events were being sent. Duplicate repository events may be sent to JIRA when you have the same repository indexed by multiple development tools. Note, JIRA will automatically remove duplicate commit events (JIRA 6.3.3+) and branch creation events (JIRA 6.3.11+) when processing workflow triggers.</i></p> <p><i>You shouldn't suppress repository events from Bitbucket Server or FishEye, unless duplicate events are causing issues to transition incorrectly.</i></p>
--	---

The issue transitions but not as expected:

▼ Possible causes...

Cause	Solution
You have configured a trigger on a global transition	<p>Investigate how the trigger event affects issues in different statuses. Consider removing the trigger from the global transition.</p> <p><i>We recommend that you do not configure triggers for global transitions, unless you are confident that you understand exactly how the trigger will affect the behavior of the issue. See Triggers and global transitions above for more information.</i></p>
Workflow conditions, validators and permissions are intentionally ignored for automatic issue transitions	<p>Do nothing.</p> <p><i>If you were expecting workflow conditions, validators or permissions to be applied to an automatic issue transition, then please note that none of these apply. Related to this, post functions do apply to automatic issue transitions.</i></p>
Your workflow is shared across multiple projects	<p>You may need to copy your workflow, if you want triggers to apply to the workflow for some projects but not others.</p> <p><i>Triggers apply to the workflow. If a workflow is shared across multiple projects, it will include all triggers that have been configured for it.</i></p>
Duplicate automatic issue transition events are being sent by multiple development tools	<p>Change the repository/project settings in one (or more) of your development tools to prevent events from being sent.</p> <p><i>Duplicate repository events may be sent to JIRA when you have the same repository indexed by multiple development tools. JIRA will automatically remove duplicate commit events (JIRA 6.3.3 and later) and branch creation events (JIRA 6.3.11 and later).</i></p> <p><i>If you are not using the latest JIRA version and have duplicate repository events causing incorrect issue transitions, you can configure Bitbucket Server (Stash 3.3 - 3.5) and FishEye (3.5+) repositories to suppress events sent to JIRA for workflow triggers.</i></p>

The information recorded for the transition is not correct:

▼ Possible causes...

Cause	Solution
-------	----------

<p>The users in your development tools do not map to users in JIRA</p>	<p>Check that each user in your development tools maps to a JIRA user.</p> <p><i>If users are not mapped correctly, then the user for the issue transition will be anonymous. For more information, see the section on user mapping above.</i></p>
<p>Known issue: The correct user is only shown on the 'History' and 'Activity' tabs for issues in JIRA, and in notification emails. In other notifications, e.g. 'Transitions' tab for issues, HipChat notifications, etc, an anonymous user is shown.</p>	<p>Do nothing.</p> <p><i>This is a known issue that will be fixed in a future release.</i></p>

3. Get help

If you still cannot resolve your problem, there are a number of other help resources available, including our applications forums, [Atlassian Answers](#), and our [support team](#).

Using validators with custom fields

Use the 'Fields Required' workflow validator that is packaged in the [JIRA Suite Utilities](#).

Please note the following caveats regarding validation of data by the 'Fields Required' workflow validator at the time of issue creation:

- fields that you set up as "required fields" are not flagged as such in the form to the end-user
- such fields can be cleared at a later time, which is not what you may have intended
- plugins will not detect the requirement as implemented by the workflow validator, so may fail later during usage

The reason 3rd party tools are needed is because JIRA's interpretation of "required" from a project's field configuration on some custom field means that the field is now required across all screens available to that project, regardless if the screen doesn't actually display that particular field. 3rd party tools, like the JIRA Suite Utilities' 'Fields Required' validator, are effectively a more granular means to control fields at the step or screen level at a project, instead of at the project level by the project's field configuration.

Using XML to create a workflow

JIRA's workflow editor generates OSWorkflow XML definition files that are stored in JIRA's database. If you need to take advantage of an OSWorkflow-based feature that is not available in JIRA's workflow editor, 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.

Importing an XML workflow into JIRA

1. Log in as a user with the **JIRA System Administrators** [global permission](#).
2. Choose  **> Issues**. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.
3. Click the **Import from XML** button to open the Import Workflow dialog box.
4. In the Name field, type a name (usually 2-3 words) to identify your new workflow.
5. (*Optional*) In the Description field, type a detailed description of your new workflow.
6. For the Workflow Definition option, you can do either of the following:
 - Upload an XML workflow definition file — to do this, choose the **Provide a full path to an XML**

file... option, and in the File Path field, type the full path to your XML workflow definition file.

⚠ This path must be local one, so your XML workflow definition file must be located on your JIRA server.

- Paste the contents of an XML workflow definition file into JIRA — to do this, choose the **Paste the workflow XML definition** option, copy the contents of your XML workflow definition file, and in the Workflow Definition (XML) field, paste this copied content.

7. 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 via the 'import a workflow from XML' link as described above.

– 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.

– 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. 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.

Workflow properties

You can use workflow properties to implement restrictions on certain steps or transitions of a workflow ([below](#)).

– Please Note: Not everything on this page is recommended!

- We do not recommend using all of these types of workflow properties as we cannot guarantee that some data and operations (e.g. bulk operations) will not be broken. **Hence, use these types of workflow properties at your own risk!**

i For details on how to implement workflow properties (i.e. step and transition properties) in your workflow, please refer to [Working with workflows](#).

Available JIRA workflow properties

There are a few workflow properties which you can use in a transition or step of a workflow. Here are some helpful links:

- [JIRA API Documentation - JiraWorkflow constant values](#)

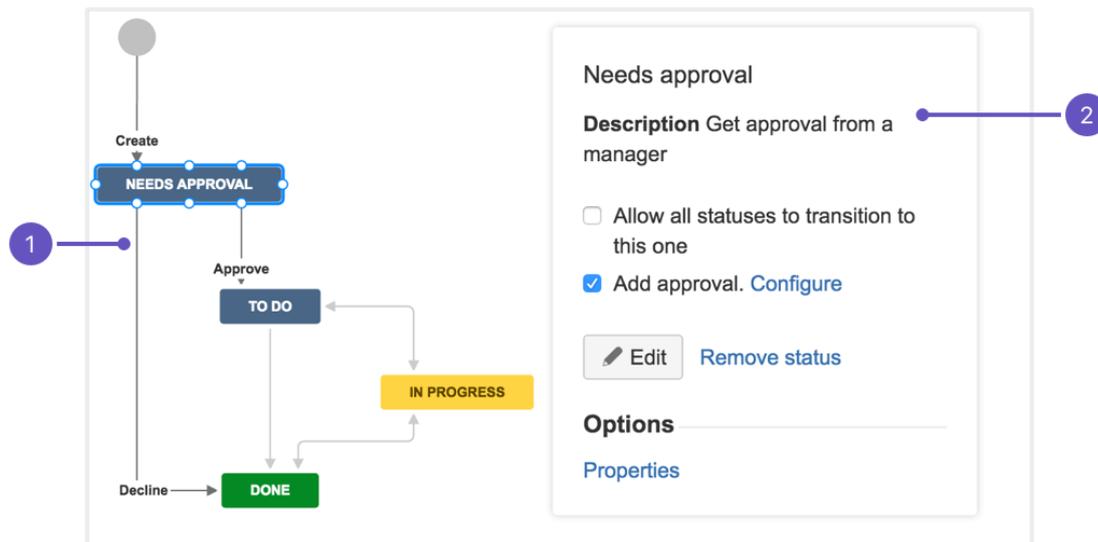
Name	Values	Related Issues	References	Notes
jira.field.resolution.exclude	Resolution id			Add comma-separated resolution ids to the transition properties where you want to not show certain resolutions
jira.field.resolution.include	Resolution id	JRA-16443		Add comma-separated resolution ids to the transition properties

<code>jira.i18n.submit</code>	i18n property key	JRA-6798		Transition (usage: action submit button name)
<code>jira.i18n.title</code>	i18n property key	JRA-6798		Transition (usage: action name, etc.)
<code>jira.issue.editable</code>	true, false		Working with workflows	Step
<code>jira.permission.*</code>	user1, user2 / group1, group2 / role / ...?	JRA-6381 JRA-34621 JRA-35917	<ul style="list-style-type: none"> • WorkflowBasedPermissionManager class description (API documentation) • Permissions based on Workflow Status • For link permissions • <code>jira.permission.edit.group=jira-administrators</code> means that only JIRA administrators can edit an issue (blog) 	Step (usage: only to restrict permission to either roles, group, or users when issue is in that step)
<code>opsbar-sequence</code>	Integer value greater than or equal to 0		Advanced workflow configuration (Customizing transitions)	Transitions on the 'View Issue' page

Configuring JIRA Service Desk approvals

JIRA Service Desk allows you to add an approval step to a status in a workflow, which allows you to specify if an approval is needed for issue types (and their associated request types) that are mapped to the workflow. The option to add an approval step is only available if the workflow is associated with at least one JIRA Service Desk project. If you add an approval step to the workflow, and that workflow is also used by a non-JIRA Service Desk project, the issues in the non-JIRA Service Desk projects can still be worked on in the usual way, but the approval step will *not* be enforced for those issues.

You can configure more than one approval step on a workflow, and you can specify if the approval step requires one or more approvers. For example, you may want a manager to initially approve a request, and then two members of your finance team may need to make the final approval on the request. You can also decide whether you want the customer to select the approver, or if you want the approver to be selected from a pre-defined list. If you want to *enforce* an approval, so that an approver *must* make a decision before the request can be progressed, you should set the approval step on a status that only has two outgoing transitions, one to Decline and one to Approve. You can set the approval step on a status that has more than two outgoing transitions, but you can only enforce two of these, the additional transitions can be actioned by a service desk agent. For example, there may be exceptional circumstances where you can't wait for an approval, so you need a way to progress the request manually. Having additional transitions on the status with the approval step allow your agents to progress the request. Transitions used by the approval step skip transition screens, so if your Decline or Approve transitions are going to a 'Done' status, add a post function that sets the resolution of the request so it's considered closed by JIRA Service Desk.



1. **Decline:** This transition leads to a Done status, and should have a post function that sets the resolution to close the request.
2. **Description:** This is generated from the status description, but it won't show on your customer portal.

How it works

An approval step is an additional configuration item that you can place on a workflow status. When a request is transitioned into this status, the nominated approver/s will receive an email informing them that they have pending approvals. If the approval step only has two exiting transitions, it can't be actioned by anyone until the approvers have either approved or declined the request. An agent can add, edit or remove approvers, but can't transition the request until it's been approved. If there are more than two exiting transitions, an agent will be able to transition the issue via the transitions *not* used in the approval step for Decline or Approve.

To add a user picker custom field to your JIRA instance and configure an approval step on a workflow, you need to have the JIRA administrator [global permission](#). To add a field to a request type, you need to have the JIRA administrator [global permission](#) or the administrator [project role](#) for that project.

1. Ensure your instance has a user picker custom field defined.

A user picker custom field (multi-user) must be defined in your instance, and it must be added to the create and edit screens associated with the workflow that you want to add the approval step to. You can use the same field for more than one approval step, but we do recommend that if you need more than one approval step on the same workflow, you use different user picker custom fields for each approval step to avoid confusion. Make sure you know the name of the user picker custom field so that you can add it to your customer portal for your request type.

- To add a user picker custom field, refer to the [Adding a custom field page](#).
- To add a user picker custom field to a screen, refer to the [Configuring a custom field page](#).

▼ 2. Add the approval step to your workflow.

JIRA Service Desk request types are mapped to JIRA issue types, and the issue types are in turn mapped to a workflow. When you add the approval step to a workflow, it will be applied to **all** issue types mapped to that workflow, and hence all request types mapped to those issue types. It's important to make sure your approval step is valid for all issue types. If you want to add an approval step for one request type that's mapped to a workflow which is applied to multiple request types, you should create a separate workflow and issue type that you can map to the request type individually. For more information on how to do this, read up on [creating issue types](#), [associating issue types with projects](#), and [working with workflows](#).

Before you add the approval step, you should ensure that the status has at least two outgoing transitions. If either of the transitions you want to use for Decline or Approve lead to a [status in the Done category](#), you should also configure a [post function on the transition to set the resolution](#) of the request.

To add an approval step:

1. Choose



> **Projects**, and select the relevant project.

2. In the **Project settings** menu, select **Workflow**. All workflows associated with your project will display.
3. Select **Edit**



in the **Actions** column of the workflow you want to modify.

4. Select the status you wish to add the approval step to. The status dialog will display.
5. Check the **Add approval** check box in the status dialog. The Add approval dialog will display.
6. Fill in the Add approval dialog with your user picker custom field, how many approvers are required, and which transitions to use for approval and decline.
7. Click **Create** to add your approval step.

Don't forget to click **Publish** to make your workflow available! You should also check that if the transitions you're using for the approval step lead to a status in the Done category, they should have a post function to set the resolution of the request so that it shows as closed.

3. Configuring the user picker custom field.

Depending on your use case, there are 3 ways to configure the user picker custom field that you selected on your approval step.

1. You want your customers to select the approvers - this is useful if you don't know who the approver may be, but the customer does, for example their manager. In this scenario, you need to make sure the user picker custom field is available on the portal so that the customer can make the selection when they create the request.
To add the user picker custom field to your request type:
 - a. Choose  **> Projects**, and select the relevant project.
 - b. Select the required project from the project list.
 - c. In the **Project settings** menu menu, select **Request types**.
 - d. Select **Edit fields** in the **Actions** column of the request type you want to add the field to.
 - e. Click **+ Add field** and select the field you want to add, and click **Apply**. The field will be added to your request type.
 - f. Edit the **Display name** (this is the name that will display on your customer portal) and **Field help** (this is the information that displays next to the field on the customer portal) fields by selecting them and entering your text. You can also decide whether to make the field optional or not at this point. Click **Update** to save your changes.
 - g. The field is added as **Shown**. You can change the order of the fields by dragging and dropping.
2. You want to set a pre-defined list of approvers - This is useful if the customer doesn't know who the approver may be, or if you know that there is a set list of approvers for the request. The customer won't need to enter any approvers when they create the request, and when the request is transitioned to the status that has the approval step, the pre-defined set of approvers will be notified. In this scenario, you need to add the user picker custom field to your request type as described above, and then select the option to Hide it. At this point, JIRA Service Desk will ask you to populate a list of approvers. Save this, and your field is hidden and your list of approvers is set.
3. You want your agents to specify the approvers - This is useful if the customer doesn't know the approver, and the approver depends on the information on the request. In this case, the agent will need to review the request, and manually add an approver. View the request in your service desk project, and the user picker custom field is displayed in the **People** section of the request. You can make any changes you need to make inline. Note that there is also an **Approval** section that lists all approvers, however you can't make any edits here. If the field isn't showing, you may need to get a JIRA administrator to check the field is still [available on your project screens](#).

Importing and exporting data

At times, you may need to import or export data to or from JIRA. You may want to import data from another tool (like Github or Fogbugz), another JIRA instance, or from a manually prepared file such as a CSV or JSON file. You may want to export your data so that you can perform some manual manipulation on it, or to move a project from one instance to another. This section of the documentation explains how to perform imports and exports of your data. If you'd like more information on [backing up your data](#), and [restoring a backup](#), please refer to the [System administration](#) section of the documentation.

Search the topics in 'Importing and exporting data':

Migrating data from

[Learn more](#) about importing data from various tools, such as Github or Fogbugz. We also have information on how to structure CSV or JSON files

other tools	for import. These files could be manually prepared by you, or may be exports from tools that we currently don't have a dedicated importer for.
Moving or archiving projects	Learn more about how you can move or archive individual or multiple projects between JIRA instances.
Importing to a Cloud instance	Learn more about importing a Server export into a Cloud instance.

Migrating from other issue trackers

When migrating from another issue tracking application to JIRA, you may wish to take your data with you. Depending on what issue tracker you are migrating from, we recommend using the relevant instructions to import data from your other issue tracker into JIRA.

i Our website highlights some top reasons why people migrate from other issue trackers to JIRA.

On this page:

- [Built-in importers](#)
- [CSV importer](#)
- [Requests for non-supported importers](#)

Built-in importers

The [JIRA Importers](#) plugin, which is bundled with JIRA, allows you to import data from the importers listed below.

- [Importing data from CSV](#)
- [Importing data from Excel](#)
- [Importing data from Bitbucket](#)
- [Importing data from Github](#)
- [Importing Data from Asana](#)
- [Importing data from TFS or Visual Studio](#)
- [Importing data from Rally](#)
- [Importing data from VersionOne](#)
- [Importing data from YouTrack](#)
- [Importing data from Axosoft](#)
- [Importing data from Pivotal Tracker](#)
- [Importing data from Bugzilla](#)
- [Importing data from FogBugz On Demand](#)
- [Importing data from FogBugz for your Server](#)
- [Importing data from Trac](#)
- [Importing data from Redmine](#)
- [Importing data from BaseCamp](#)
- [Importing data from JSON](#)
- [Importing data from Mantis](#)

CSV importer

If you are migrating from a system for which JIRA does not provide a built-in importer, you may be able to import your data into JIRA via CSV format instead. Your system must be able to export your data into a CSV (comma-separated value) file. You can then import the CSV file into JIRA using JIRA's CSV importer:

- [Importing data from CSV](#)

There is also a workaround for [importing comments](#).

Requests for non-supported importers

We are also tracking requests to add other systems to our built-in importers. We encourage users to vote and comment on the systems they are interested in:

- [Gemini](#)
- [Code Spaces](#)

Importing data from CSV

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import your data from a comma-separated value (CSV) file. CSV files are text files representing tabulated data and are supported by most applications that handle tabulated data (for e.g. Microsoft Excel, databases, etc.).

The CSV import feature allows you to import issues from an external (issue tracking) system which:

- JIRA does not provide a dedicated import tool for and
- Can export its data in a structured/tabulated format (preferably CSV).

i Our main website highlights some top reasons why people [migrate from such an external issue tracking system to JIRA](#).

The CSV import process consists of:

1. [Preparing your CSV file \(below\)](#).
2. [Running the CSV file import wizard \(below\)](#).
 - You can choose to map individual fields and field values during the import process.
 - At the end of the CSV file import wizard, you will be given the option of creating a CSV configuration file, which contains the settings you configured while running through the CSV file import wizard. This is useful if you need to test your CSV file import on a test JIRA server first before performing the import on a production system.

! Please note:

- Several methods are available for importing data from other issue tracking systems into JIRA. Depending on your other issue tracking system, it may be more appropriate to use one of these other methods than to first export your data from that system to a CSV file and then import that CSV file into JIRA. If your other issue tracking system is listed on the [Migrating from other issue trackers](#) page, try using the appropriate method for that issue tracker (which is accessible from that page) to import data into JIRA.
- If you want to raise a bug report or improvement suggestion about this feature, please do so within the [JIRA Importers plugin](#) project.

Preparing your CSV file

The JIRA Importers plugin assumes that your CSV file is based off a default Microsoft Excel-styled CSV file. Fields are separated by commas and any content that must be treated literally, such as commas and new lines/'carriage returns' themselves are enclosed in quotes.

i For Microsoft Excel and OpenOffice, it is not necessary to quote values in cells as these applications handle this automatically.

CSV file requirements

In addition to being 'well-formed', CSV files have the following requirements.

Each CSV file must possess a heading row with a Summary column

The CSV file import wizard ([below](#)) uses a CSV file's header row to determine how to map data from the CSV file's 2nd row and beyond to fields in JIRA.

The header row *should avoid containing any punctuation* (apart from the commas separating each column) or the importer may not work correctly.

On this page:

- [Preparing your CSV file](#)
- [Running the CSV file import wizard](#)
- [Tips for importing CSV data into JIRA fields](#)

The header row *must* contain a column for 'Summary' data.

Commas (as column/field separators) cannot be omitted

For example, this is valid:

```
Summary, Assignee, Reporter, Issue Type, Description, Priority
"Test issue", admin, admin, 1, ,
```

... but this is not valid:

```
Summary, Assignee, Reporter, Issue Type, Description, Priority
"Test issue", admin, admin, 1
```

Encapsulating JIRA data structure in your CSV file

Capturing data that spans multiple lines

Use double-quote marks (") in your CSV file to capture data that spans multiple lines. For example, upon import, JIRA will treat the following as a valid CSV file with a single record:

```
Summary, Description, Status
"Login fails", "This is on
a new line", Open
```

Treating special characters literally

Use double-quote marks (") around a section of text to treat any special characters in that section literally. Once this data is imported into JIRA, these special characters will be stored as part of JIRA's field data. Examples of special characters include carriage returns/enter characters (as shown in the example above), commas, etc.

To treat a double quote mark literally, you can 'escape' them with another double quote mark character. Hence, the CSV value:

- "Clicking the ""Add"" button results in a page not found error" once imported, will be stored in JIRA as:
- Clicking the "Add" button results in a page not found error

Aggregating multiple values into single JIRA fields

You can import multiple values into a JIRA field that accepts multiple values (e.g. **Fix (for) Version, Affects Version, Component, Labels**). To do this, your CSV file must specify the same column name for each value you wish to aggregate into the mapped JIRA field. The number of column names specified must match the maximum number of values to be aggregated into the mapped field. For example:

```
IssueType, Summary, FixVersion, FixVersion, FixVersion, Component,
Component
bug, "First issue", v1, , , Component1,
bug, "Second issue", v2, , , Component1, Component2
bug, "Third issue", v1, v2, v3, Component1,
```

In the above example, the **Component** field of the second issue and the **Fix Version** field of the third issue will generate multiple values in appropriate JIRA fields upon import.

 Be aware that only a limited number of JIRA fields support multiple values. The CSV importer will not

allow you to import aggregated data into JIRA fields which only support a single value.

Importing attachments

You can attach files to issues created from your CSV file. To do this, specify the URL of your attachment in an 'Attachments' column within your CSV file.

```
Assignee, Summary, Description, Attachment, Comment
Admin, "Issue demonstrating the CSV attachment import", "Please check
the attached image below.",
"https://jira-server:8080/secure/attachment/image-name.png",
"01/01/2012 10:10;Admin; This comment works"
Admin, "CSV attachment import with timestamp,author and filename",
"Please check the attached image below.", "01/01/2012
13:10;Admin;image.png;file://image-name.png", "01/01/2012
10:10;Admin; This comment works"
```

i URLs for attachments support the HTTP and HTTPS protocols and can be any location that your JIRA server *must* be able to access. You can also use the FILE protocol to access files in the `import/attachments` subdirectory of your JIRA home directory.

Creating sub-tasks

You can create sub-tasks of issues through a CSV file import, by encapsulating this structure in your CSV file. To do this:

- Your CSV file requires two additional columns whose headings should be named similarly to **Issue Id** and **Parent Id**.
- Ensure each regular (non sub-task) issue is given a unique (sequential) number in the **Issue Id** column. Do not include any value in the **Parent Id** fields for regular issues.
- To create a sub-task of a regular issue in your CSV file, reference the unique **Issue Id** number of the regular issue in the **Parent Id** column. Do not include any value in the **Issue Id** fields for sub-tasks.

For example:

```
IssueType, Summary, FixVersion, FixVersion, FixVersion, Component,
Component, Issue ID, Parent ID, Reporter
Bug, "First issue", v1, , , Component1, , 1, , jbloggs
Bug, "Second issue", v2, , , Component1, Component2, 2, , fferdinando
Bug, "Third issue", v1, v2, v3, Component1, , 3, , fferdinando
Sub-task, "Fourth issue", v1, v2, , Component2, , , 2, jbloggs
```

In the example above, the fourth issue will be sub-task of the second issue upon import, assuming you match the 'Issue ID' and 'Parent ID' fields in your CSV file to the **Issue Id** and **Parent Id** JIRA fields, respectively during the [CSV file import wizard](#).

Importing issues into multiple JIRA projects

You can import issues from your CSV file into different JIRA projects through a CSV file import. To do this:

- Your CSV file requires two additional columns whose headings should be named similarly to **Project Name** and **Project Key**.
 - Ensure that every issue represented in your CSV file contains the appropriate name and key in these columns for the JIRA projects to which they will be imported.
- i** The project name and key data is the *minimum JIRA project data* required for importing issues from a CSV file into specific JIRA projects.

```
IssueType, Summary, Project Name, Project Key
bug, "First issue", Sample, SAMP
bug, "Second issue", Sample, SAMP
task, "Third issue", Example, EXAM
```

In the example above, the first and second issues will be imported into the 'Sample' project (with project key 'SAMP') and the third issue will be imported into the 'Example' project (with project key 'EXAM') , assuming you match the 'Project Name' and 'Project Key' fields in your CSV file to the **Project name** and **Project key** JIRA fields, respectively during the [CSV file import wizard](#).

How to handle unresolved issues

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.

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 four fields, there are two special options in the select list in addition to JIRA's available values:

- 'Import as blank' — this causes the JIRA value to be blank for that field. Note that, if you are importing Unresolved issues, you should create a field mapping for the Resolution field and set the value 'Unresolved' to 'Import as blank'.
- 'No mapping' — this 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 'Import as blank' option is selected.

Importing worklog entries

Your CSV file can contain worklog entries. For example:

```
Summary,Worklog
Only time spent (one hour),3600
With a date and an author,2012-02-10 12:30:10;wseliga;120
With an additional comment,Testing took me 3 days;2012-02-10
12:30:10;wseliga;259200
```

To track time spent, you need to use seconds.

Importing to multi select custom fields

Your CSV file can contain multiple entries for the one Multi Select Custom Field. For example:

```
Summary,Multi Select,Multi Select,Multi Select
Sample issue,Value 1,Value 2,Value 3
```

This will populate the Multi Select Custom Field with multiple values.

Importing cascading choice custom fields

You can import values to a cascading choice custom field using the following syntax:

```
Summary, My Cascading Custom Field
Example Summary, Parent Value -> Child Value
```

The '->' separator allows you to import the hierarchy.

NOTE: Currently JIRA does not support importing multi-level cascading select fields via CSV (**JRASERVER-34202** - Allow CSV import to support Multi-Level Cascading Select plugin fields **GATHERING INTEREST**).

Updating existing issues

From version 4.3 of JIRA Importers plugin you can update existing issues. Your CSV file needs to contain a column that during the import wizard is mapped to Issue Key. If an issue exists for a given key it will be updated. For example:

```
issue key,summary,votes,labels,labels
TT-1,Original summary,1,label1,label2
TT-1,,7,label-1,label-2
TT-1,Changed summary,,,
TT-2,Original summary 2,1,label-1,label-2
TT-2,,<<!clear!>>,<<!clear!>>,
```

First row will create an issue, second row will set votes to 7, and add two labels. Following row will change the summary. Issue TT-2 will be created with two labels, the second row will remove those labels with a special marker <<!clear!>>.

 Importing a CSV to update existing issues will **reset columns to their default values** if they are not specified in the CSV.

Running the CSV file import wizard

Before you begin, please [back up](#) your JIRA data.

1. Log in to JIRA as a user with the **JIRA Administrators** [global permission](#).
2. Select **Administration > System > Import & Export > External System Import**.
3. Select **CSV** to open the **CSV File import** page.
4. On the **CSV File import** page, select your **CSV Source File**. If you want to change the file's encoding and CSV delimiter format, click the **Advanced** heading to reveal this option (as shown in the above screenshot).
 - Note:**
 - The file will be imported using the **File encoding** you specify here (which is **UTF-8** by default).
 - If your CSV file uses a different separator character other than a comma, specify that character in the **CSV Delimiter** field.
5. Leave the **Use an existing configuration file** checkbox cleared if you do not have a configuration file or if you want to create a new configuration file. Configuration files specify a mapping between column names in your CSV file's header row and fields in your JIRA installation.
 - Note:**
 - If you select this option, you will be asked to specify an **Existing Configuration File**.
 - If you do not select this option, then at the end of the CSV file import wizard, JIRA will create a configuration file which you can use for subsequent CSV imports (at this step of the CSV file import wizard).
6. Click the **Next** button to proceed to the **Setup project mappings** step of the CSV file import wizard.
7. On the **Setup project mappings** page, you can either import *all* your issues into either one JIRA project (new or existing), or multiple JIRA projects (by ensuring that your CSV file includes the minimum JIRA project data required — i.e. the JIRA project name and key). Complete the following fields/options:

Import to JIRA Project	<p>Choose either of the following:</p> <ul style="list-style-type: none"> • Select a project and then do either of the following: <ul style="list-style-type: none"> • Start typing the name (or key) of a project that already exists in JIRA or use the drop-down menu to select an existing JIRA project. • Select Create New from the drop-down menu and in the resulting Add A New Project dialog box, type the following: <ol style="list-style-type: none"> a. A new project Name b. A new project Key <ul style="list-style-type: none"> 📘 This will be used as the prefix for all issue IDs in your JIRA project. c. The Project Lead. • Defined in CSV. Ensure that every issue in your CSV file includes data for the JIRA Project Name and Project Key. <ul style="list-style-type: none"> ✅ This option is useful if you want to import issues from your CSV file into multiple JIRA projects. See Importing issues into multiple JIRA projects for details.
E-mail Suffix for New Users	Enter the email address domain for any new users specified in the CSV file which will be added to JIRA during the import.
Date format in import file	Specify the date format used in your CSV file. Use the syntax that complies with the Java SimpleDateFormat .

8. Click the **Next** button to proceed to the **Setup field mappings** step of the CSV file import wizard.
9. On the **Setup field mappings** page, specify each **CSV Field** (determined by your CSV file's header row) you want to import into your chosen JIRA project by selecting their checkboxes under the **Import** column on the left.
 - 📘 **Please note:**
 - At least one of these fields must contain data for JIRA's **Summary** field.
 - If your CSV file contains more than one of the same field name specified in its header row, the CSV file import wizard will aggregate these into a single field, which will be marked by a ⚠️ symbol at this step of the wizard.
10. In the **JIRA field** column, select the JIRA fields you want to match to fields defined in your CSV file (i.e. each **CSV Field** you selected in the previous step). For more information about matching CSV fields to JIRA fields, see [Tips for importing CSV data into JIRA fields](#) below.
 - 📘 **Please note:**
 - The **Summary** field must be specified for one of your JIRA fields and the **Next** button will remain unavailable until you do so.
 - For CSV fields which have been aggregated by the CSV file import wizard, you will only be able to select JIRA Fields that support multiple values.
 - If you are importing sub-tasks, remember to match the **Issue ID** and **Parent ID** fields in JIRA to those in your CSV file.
 - If you are importing issues into multiple projects, ensure that you selected **Defined in CSV** during the **Setup project mappings** step above and remember to match the **Project Name** and **Project Key** fields in JIRA to those in your CSV file.
11. To modify the values of any fields' data in the CSV file *before* they are imported into JIRA, select the **Map field value** checkboxes next to the appropriate fields.
12. Click the **Next** button to proceed to the **Setup value mappings** step of the CSV file import wizard.
13. On the **Setup value mappings** page, specify the JIRA field values for each CSV file field value (which has been detected by the CSV file import wizard).
 - 📘 **Please note:**
 - Any fields whose **Map field value** checkboxes were selected in the previous step of the CSV file import wizard will be presented on this page.
 - Leave a field cleared or clear any content within it if you wish to import the value 'as is'.
 - You can create new **Priority**, **Resolution** and **Issue Type** values in JIRA (i.e. based on the data in your CSV file) by clicking the **Add new ...** link (e.g. **Add new issue type 'subtask'** shown in the screenshot above) next to the appropriate field.
 - If you are importing a username-based CSV field (e.g. **Reporter** or **Assignee**) and you do not select the **Map field value** checkbox for this field in the previous step of the CSV file import

wizard, then the importer will automatically map imported usernames from the CSV file to (lowercase) JIRA usernames.

i Regardless of whether or not you select the **Map field value** checkbox, JIRA will automatically create usernames based on the data in your CSV file if they have not already been defined in JIRA.

- Click the **Begin Import** button when you are ready to begin importing your CSV data into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.

i **Note:**

- If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the CSV file import process.
- If you need to import another CSV file with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a CSV configuration file, which you can use at the [first step](#) of the CSV file import wizard.

Congratulations, you have successfully imported your CSV data into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing CSV data into JIRA fields

Below are some helpful tips when importing data from your CSV file into specific JIRA fields:

JIRA Field	Import Notes
Project	CSV data is imported on a per-project basis. You can either specify an existing JIRA project(s) as the target, or the importer will automatically create a new project(s) for you at time of import.
Summary	This is the only required field.
Issue Key	You can set the issue key for an imported issue. If an issue with a given key already exists in JIRA, it will be updated instead.
Component(s)	You can import issues with multiple components by entering each component in a separate column.
Affects Version(s)	You can import issues with multiple 'Affects Versions' by entering each version in a separate column.
Fix Version(s)	You can import issues with multiple 'Fix Versions' by entering each version in a separate column.
Comment Body	You can import issues with multiple comments by entering each comment in a separate column.
Date Created	Please use the date format specified on the second step of the CSV import wizard.
Date Modified	Please use the date format specified on the second step of the CSV import wizard.
Due Date	Please use the date format specified on the second step of the CSV import wizard.
Issue Type	If not specified in your CSV file, imported issues will be given the default (i.e. first) Issue Type as specified in your JIRA system Defining issue type field values . You can also create new JIRA values on-the-fly during the import process.
Labels	Import issues with multiple labels by: <ul style="list-style-type: none"> entering each label in a separate column or putting all labels in one column, delimited by a space
Priority	If not specified in your CSV file, imported issues will be given the default (i.e. first) Priority as specified in your JIRA system Defining priority field values . You can also create new JIRA values on-the-fly during the import process.

Resolution	If not specified in your CSV file, imported issues will be given the default (i.e. first) Resolution as specified in your JIRA system Defining resolution field values . You can also create new JIRA values on-the-fly during the import process. Also, see How to handle unresolved issues for helpful tips.
Status	Can only be mapped to existing workflow statuses in JIRA. If not specified in your CSV file, imported issues will be given the default (i.e. first) Status as specified in your JIRA system.
Original Estimate	The value of this field needs to be specified as number of seconds.
Remaining Estimate	The value of this field needs to be specified as number of seconds.
Time Spent	The value of this field needs to be specified as number of seconds.
Users	You can choose to have the importer automatically create JIRA users for any values of the Assignee or Reporter field. <ul style="list-style-type: none"> • Users will be created as active accounts in JIRA. Users will need to get their passwords emailed to them the first time they log into JIRA. • Users with no real name will get the portion of their email address (login name) before the "@" character as their Full Name in JIRA. • 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. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created. • If Assignee and Reporter are not mapped, then no usernames are created
Watchers	If you have users specified as Watchers in your CSV file, and these users do not exist in JIRA, they will not be imported. A user must be available in JIRA before you can import them as a watcher on a specific issue.
Other fields	If you wish to import any other fields, you can choose to map them to specific JIRA custom field(s). If your custom fields don't yet exist in JIRA, the importer can automatically create them for you. If your custom field is a date field, please use the date format specified on the second step of the CSV import wizard.

Commonly asked CSV questions and known issues

This page answers some of the commonly asked CSV questions our technical support staff have encountered. If you are not able to find an answer from this page and our [issue tracker](#), feel free to [create a support issue](#).

Commonly Asked Questions

The importer simply doesn't work on my CSV file!

Please make sure that it is a valid and not-bad-formatted CSV file. You should be able to spot this with by turning on detailed [logging and profiling](#). Also, please double check your configuration file and ensure that it's properly configured, e.g. exact delimiter, date format, etc.

The importer fails at date fields, why?

If you are seeing error message similar to this:

```
[00:55:28] FAILED: Customfield value 01/Nov/06 12:00 AM is invalid
[00:55:28]
com.atlassian.jira.issue.customfields.impl.FieldValidationException:
Invalid date format. Please enter the date in the format "MMM/dd/yy".
at
com.atlassian.jira.issue.customfields.converters.DatePickerConverter.get
Timestamp(DatePickerConverter.java:57)
at
com.atlassian.jira.issue.customfields.impl.DateCFType.getSingularObjectF
romString(DateCFType.java:46)
at
com.atlassian.jira.imports.importer.impl.DefaultJiraDataImporter.importI
ssues(DefaultJiraDataImporter.java:531)
at
com.atlassian.jira.imports.importer.impl.DefaultJiraDataImporter.doImport
(DefaultJiraDataImporter.java:104)
at
com.atlassian.jira.imports.importer.impl.ImporterThread.run(ImporterThre
ad.java:21)
```

There are a few possible reasons:

- The format of dates is not correctly set in the import configuration file. The date format for custom fields must match the "Date format in input file" which has a default format of yyyyMMddHHmmss
- JIRA system date fields such as Created, Updated and Due Date use "yyyy-MM-dd HH:mm:ss" but may need an offset adding
- Date Picker and Date Time Picker formats are not consistent, e.g.

```
jira.date.picker.java.format=dd/MMM/yy
jira.date.time.picker.java.format=MMM/dd/yy hh:mm a
```

should be corrected to,

```
jira.date.picker.java.format=dd/MMM/yy
jira.date.time.picker.java.format=dd/MMM/yy hh:mm a
```

Why does the importer always ask me to map values to column (at Step 3 of 5)?

It is because you have selected *Map Field Value* for the particular columns. To use the values from the CSV, you need just to map the column to the *Corresponding JIRA field*, otherwise, select the *Map field value* checkbox

.

Known Issues

Why couldn't I import from cascading select fields?

This is an open issue being tracked at [JIM-231](#). Feel free to comment and vote on it.

Why couldn't I import component/version Custom Fields?

This issue is being tracked at [JIM-233](#). Feel free to comment on it.

Known JBoss issue

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.

How to import CSV data with PVCS command

The content on this page relates to platforms which are not supported for JIRA. Consequently, Atlassian can not guarantee providing any support for it. Please be aware that this material is provided for your information only and using it is done so at your own risk.

Importing from PVCS is not supported yet, but there is a feature request being tracked [here](#). The above problem occurs when the pvcs command is not configured in the CSV configuration.

Resolution

In order to import the author of the comment and the date of the comment successfully, there are a few required conditions:

- Append the settings in the csv configuration file which you have saved the configuration through the wizard

```
settings.advanced.mapper.comment :  
com.atlassian.jira.imports.csv.mappers.PvcsComment
```

- **i** For the latest plugin version 2.6.1, please use the configuration below:

```
settings.advanced.mapper.comment :  
com.atlassian.jira.plugins.importer.imports.csv.mappers.PvcsComment
```

- Username (Example: eddie) must exist in JIRA
- The format of the comment should be as below:

```
"QA Note on Close: eddie: 4/28/2004 11:54:35 AM: Closing this  
defect as it is no longer relevant"
```

Importing data from Excel

Unfortunately, right now we don't have a built-in JIRA importer for native Microsoft Excel files. **But it is still possible to perform a two-stage import using CSV import mechanisms.**

How to transform an MS Excel files into a CSV file

Microsoft Excel is capable of saving the spreadsheet in multiple file formats, including CSV. Before you save, we recommend you clean up the spreadsheet from all unnecessary information or macros and make sure that the table columns are labeled correctly.

When ready, select File / Save As and choose the CSV format from the "save as type" drop-down list.

In case of problems please refer to Microsoft documentation for help.

How to import CSV data back to JIRA

If you want to create issues as well as projects, users, etc. please refer to our [CSV importer help](#). If you don't have administrative privileges in JIRA, you can also import CSV data into a single project through the user CSV importer, if enabled. In both cases, the importer wizards will guide you through the steps of mapping fields and values and validating the data before the import.

If you are looking for an easier solution

The import through CSV has some limitations. If the results aren't satisfactory, there are complete third party solutions available which might help you. Please check out the following offers solution from Atlassian Marketplace:

- [Excel Connector for JIRA](#) from Transition Technologies S.A.

You can also contact your local Atlassian Expert for help.

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the future of JIRA. And for that we thank you.

Importing data from Bitbucket

The [JIRA Bitbucket Importer plugin](#) allows you to import data from Bitbucket into your local JIRA instance.

Before you begin

⚠ Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

- Install the [JIRA Bitbucket Importer plugin](#). For instructions on how to install a plugin, see [Installing Marketplace apps](#).
- Ensure that you are using version 6.0.4 or later of the [JIRA Importers Plugin](#). This plugin is bundled with JIRA. For instructions on how to update a plugin, see [Updating apps](#).
- [Back up](#) your existing JIRA data.
- Be sure that you have enabled the [issue tracker](#) on your Bitbucket repository and that you have administrator permission on it.

Import your Bitbucket data

The JIRA Bitbucket Importer plugin provides a wizard that walks you through the process of importing data and integrating it with JIRA. After you've installed it, run the wizard to import your Bitbucket data:

1. Choose



> **System.**

2. In the **Import & Export** section, select **Bitbucket**.
3. Complete the fields as prompted in the wizard. Depending on how your sites are configured, you might be redirected to Bitbucket in order to set the authorization needed to export data.

If you are importing Bitbucket issues into an existing JIRA project, you must choose the JIRA workflow scheme used by that existing JIRA project when you are prompted to select the workflow scheme. Otherwise, your import may not complete successfully.

In addition, you must map Bitbucket statuses to JIRA statuses in order for JIRA workflows to work with the issues.

Tips for importing Bitbucket data into JIRA fields

The import process converts Bitbucket data as follows:

In Bitbucket	In JIRA	Import Notes
--------------	---------	--------------

Repository	Project	Bitbucket data is imported on a per-project basis. You can either specify an existing JIRA project as the target, or the importer will automatically create one or more projects during the import. See Defining a project for more information about JIRA projects.
Title	Summary	Bitbucket subject is imported as the JIRA issue summary.
Worklog	Worklog	See Configuring time tracking .
Reporter	Reporter	Bitbucket issue author is mapped as JIRA Issue Reporter.
Attachments	Attachments	Attachments are extracted from Bitbucket and saved. Information on the date the file was attached and the user who attached it is retained, as well. To specify the location where the attachments are stored, see Configuring file attachments .
Kind	Issue Type	You can configure the mapping of specific kinds to specific JIRA issue types.
Priority	Priority	You can configure the mapping of specific Bitbucket values to specific JIRA values.
Status	Status	<p>You can configure the mapping of specific Bitbucket values to specific JIRA values, provided you create your workflows in JIRA before running the importer.</p> <ul style="list-style-type: none"> • The JIRA status field is integral to JIRA workflow. • To create a JIRA workflow, see Working with workflows. • To create a JIRA workflow scheme (which you can then associate with appropriate projects and Issue Types), see Managing your workflows.

User	User	<p>You can choose to have the importer automatically create JIRA users for any Bitbucket users who do not already exist in JIRA.</p> <ul style="list-style-type: none"> • Users who interacted with the Bitbucket system will be created as active accounts in JIRA. Other users will be imported into a special group called "bitbucket-import-unused-users" and will be deactivated. • Passwords from Bitbucket are not imported. Users from Bitbucket must have their passwords emailed to them the first time they log into JIRA. • 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 starting the import. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will open and list the users that can't be created.
------	------	---

Importing data from Github

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **GitHub** by connecting to a live GitHub database.

The GitHub importer is compatible with JIRA 6.1 and above.

i Our main website highlights some top reasons why people [migrate from GitHub to JIRA](#).

The GitHub import process consists of running the GitHub Import Wizard, which will step you through the process to connect to GitHub, and map and import your data to JIRA. The GitHub importer will connect to GitHub using your GitHub username and password (which you must provide) **or** with a [Personal Access Token](#). If you are using GitHub Enterprise, you will also have to provide your GitHub Enterprise URL (which you can obtain in GitHub under your Enterprise Settings). The GitHub importer will be able to access and import data from your personal and public repositories, and any other repositories that you have starred, so you should make sure you've starred any other repositories you want to import data from. You don't have to select all your personal, public and starred repositories, the GitHub importer will display all repositories it can access and you can pick and choose which ones you want to import. If your GitHub instance has 2 factor authentication, you will be required to either provide the 6 digit access code that you will be sent, or a back-up code.

i If you have attachments in GitHub and you want to import these too, you must ensure you have [attachments enabled in JIRA](#). Attachments are enabled by default.

configuration file (this will be a text file), which you can use at the [first step](#) of the GitHub Import Wizard.

Congratulations, you have successfully imported your GitHub repository data into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Importing Data from Asana

The Asana importer for the JIRA Importers plugin allows you to import data from **Asana** by connecting to a live Asana API.

The Asana import process consists of running the Asana Import Wizard, which will step you through the process to connect to Asana, and map and import your data to JIRA. The Asana importer will connect to Asana using your Asana API Key. You can find it in your Asana account settings on <https://asana.com> - select the "APPS" tab, then click "API Key...".

The Asana importer will be able to access and import data from your all of your Asana workspaces. You don't have to select all your workspaces, the Asana importer will display all projects in all workspaces and allow you to map the Asana projects onto JIRA projects.

i If you have attachments in Asana and you want to import these too, you must ensure you have [attachments enabled in JIRA](#). Attachments are enabled by default.

Running the Asana Import Wizard

If your JIRA installation has existing data, then before you begin, back up your existing JIRA data.

1. Log in to JIRA as a user with the **JIRA Administrators** [global permission](#).
2. The import wizard allows you to import data from other sources. Choose



> **System**. Select **Import & Export > External System Import** to open the Import external projects page.

3. Select **Asana** to open the **Asana Import Wizard**.
4. On the **Asana Connection** page, instead of providing regular credentials (login and password), please provide the Asana API key. Asana API Key is a single authentication token that can be used by external systems to connect to Asana for data extraction and insertion. You can find it in your Asana account settings on <https://asana.com> - select the "APPS" tab, then click "API Key...".
5. Click **Next**. The **Map projects** page displays, and will show a list of all your Asana projects within all the workspaces that you have in Asana.
6. On the **Map projects** page, select the projects you want to import data from, and where you want to import it to.
 - i** All Asana projects are initially set to "Don't import this project". To import a project, you must either select an existing compatible project to import the data to, or create a new project. To create a new project, select **Create New** from the drop-down menu and in the resulting **Add A New Project** dialog box, type the following:
 - a. A new project **Name**
 - b. A new project **Key**
 - i** This will be used as the prefix for all issue IDs in your JIRA project.
 - c. The **Project Lead**.
7. Click **Next**. On the **Map fields** page, select the workflow scheme you want to use for your newly created JIRA projects.
8. Click **Next**. The **Map values** page allows you to map your Asana tasks' scheduling status to JIRA priorities, and the completion flag to JIRA workflow steps.
9. Click **Next**. On the **Links** page, you can choose how issues that are subtasks of one another will be linked in JIRA.
10. Click **Begin import**.
11. Success! You have completed importing your Asana data to JIRA. If there were any errors or warnings, these will be displayed to make you aware that you may need to check some details.

i Note:

- If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the Asana Import Wizard process.
- If you need to import data from another Asana project with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a Asana configuration file (this will be a text file), which you can use at the [first step](#) of the Asana Import

Wizard.

- Before performing another import to the same project, you will want to remove the *External issue ID* field created in the previous import. Otherwise, the importer will associate existing imported issues with the next import, and will skip importing new issues up to the existing ID. For example, if you import 50 issues at first, they will be assigned IDs from 1 to 50. In the next imports, first 50 issues will be skipped, as their IDs will have already existed in JIRA.
- Asana tasks can be created without a name. As JIRA doesn't allow issues with no summary, such tasks will be given a default name and a warning will displayed in the log.

Congratulations, you have successfully imported your Asana repository data into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Importing data from TFS or Visual Studio

Unfortunately, right now we don't have a built-in JIRA importer for data from Microsoft Team Foundation Server for Visual Studio. **But it is still possible to perform a two-stage import using Visual Studio's export mechanisms.**

How to export data from Visual Studio into a CSV file

This process has two steps.

In step one, you need to create a query with the work items that you want to export. When this a query is created, you can save its results into the Excel spreadsheet. (You might need to install Microsoft Excel add-in to Team Foundation Server first.)

In step two, you need to save the resulting spreadsheet into a CSV format.

Please refer to Microsoft Team Foundation Server and Visual Studio documentation for help.

How to import CSV data back to JIRA

If you want to create issues as well as projects, users, etc. please refer to our [CSV importer help](#). If you don't have administrative privileges in JIRA, you can import CSV data directly into a single project with the user CSV importer. In both cases the importer wizards will guide you through the steps of mapping fields and values and validating the data before the import.

If you are looking for an easier solution

The import through CSV has certain limitations. If the results aren't satisfactory, there are complete third party solutions available which might help you. Please check out the following solutions from Atlassian Marketplace:

- [TFS4JIRA](#) from Spartz
- [UseTFS](#) from Pigsty
- [JIRA Connector for ConnectALL](#) from Go2Group

You can also contact your local Atlassian Expert for help.

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the future of JIRA. And for that we thank you.

Importing data from Rally

Unfortunately, right now we don't have a built-in JIRA importer for data from Rally. **But it is still possible to perform a two-stage import using Rally's CSV export mechanisms.**

How to export data from Rally into a CSV file

It's possible to export data from Rally into CSV or XML files. However, CSV files are more reliable and we

recommend them for the purpose of the migration.

In order to create a CSV file, go to the Rally's summary page and from the "actions" menu select the "CSV" option. Save the resulting file. This kind of export will only contain the data visible on the summary page. If you want to export all data you need to create a custom view first. Refer to Rally's documentation for help.

How to import CSV data back to JIRA

If you want to create issues as well as projects, users, etc. please refer to our [CSV importer help](#). If you don't have administrative privileges in JIRA, you can also import CSV data into a single project through the user CSV importer, if enabled. In both cases, the importer wizards will guide you through the steps of mapping fields and values and validating the data before the import.

If you are looking for an easier solution

The import through CSV has some limitations. If the results aren't satisfactory, there are complete third party solutions available which might help you. Please check out the following solutions from Atlassian Marketplace:

- [Rally to JIRA Enterprise Migration Tool](#) from cPrimeLabs
- [JIRA Connector for ConnectALL](#) from Go2Group
- [agosense.symphony](#) from agosense GMBH

You can also contact your local Atlassian Expert for help.

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the future of JIRA. And for that we thank you.

Importing data from VersionOne

Unfortunately, right now we don't have a built-in JIRA importer for data from VersionOne. **But it is still possible to perform a two-stage import using VersionOne's CSV export mechanisms.**

How to export data from VersionOne into a CSV file

In order to create a CSV file you need to use the VersionOne's custom reporting. Custom reporting allows you to perform an export to different file formats, including CSV. Refer to VersionOne's documentation for help on how to use custom reporting and exporting to CSV. Save the resulting file.

How to import CSV data back to JIRA

If you want to create issues as well as projects, users, etc. please refer to our [CSV importer help](#). If you don't have administrative privileges in JIRA, you can also import CSV data into a single project through the user CSV importer, if enabled. In both cases, the importer wizards will guide you through the steps of mapping fields and values and validating the data before the import.

If you are looking for an easier solution

The import through CSV has some limitations. If the results are not satisfactory, there are complete third party solutions available which might help you. Please check out the [JIRA Connector for ConnectALL](#) from Go2Group on Atlassian Marketplace.

You can also contact your local Atlassian Expert for help.

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the future of JIRA. And for that we thank you.

Importing data from YouTrack

Unfortunately, right now we don't have a built-in JIRA importer for data from YouTrack. But it is still possible to perform a two-stage import using VersionOne's CSV export mechanisms.

How to export data from YouTrack into a CSV file

In order to create a CSV file you need to select the "Issues in CSV" option from your reports menu in YouTrack. Make sure to prepare the search criteria first so that exported data set is exactly what you want to have imported into JIRA. Refer to YouTrack's documentation for help on how to use filters and reports. Save the resulting CSV file.

How to import CSV data back to JIRA

If you want to create issues as well as projects, users, etc. please refer to our [CSV importer help](#). If you don't have administrative privileges in JIRA, you can also import CSV data into a single project through the user CSV importer, if enabled. In both cases the importer wizards will guide you through the steps mapping fields and values and validating the data before the import.

If you are looking for an easier solution

The import through CSV has some limitations. If the results aren't satisfactory, please contact your local Atlassian Expert for help.

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the future of JIRA. And for that we thank you.

Importing data from Axosoft

Unfortunately, right now we don't have a built-in JIRA importer for data coming from Axosoft. **But it is still possible to perform a two-stage import using Axosoft's CSV export mechanisms.**

How to export data from Axosoft into a CSV file

In order to create a CSV file you need to go to your list of items or work logs and select the "Export" option from the "More" menu. Make sure to select all fields for the export, otherwise some information may not be visible in JIRA. Save the resulting CSV file.

How to import CSV data back to JIRA

If you want to create issues, projects, users, etc. please refer to our [CSV importer help](#). If you don't have administrative privileges in JIRA, you can also import CSV data into a single project through the user CSV importer, if enabled. In both cases, the importer wizards will guide you through the steps of mapping fields and values and validating the data before the import.

If you are looking for an easier solution

The import through CSV has some limitations. If the results aren't satisfactory, please contact your local Atlassian Expert for help.

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the

future of JIRA. And for that we thank you.

Importing data from Pivotal Tracker

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **Pivotal Tracker**, a 'Software as a Service' (SaaS) issue tracker application.

i Our main website highlights some top reasons why people [migrate from Pivotal Tracker to JIRA](#). Version 2.5 or later of the JIRA Importers Plugin is required.

On this page:

- Preparing Pivotal Tracker for data import into JIRA
- Running the Pivotal Tracker Import Wizard
- Tips for importing Pivotal Tracker data into JIRA fields

Preparing Pivotal Tracker for data import into JIRA

In Pivotal Tracker, please ensure you have switched on **Allow API Access** in your Pivotal Project's Settings.

The screenshot shows the Pivotal Tracker interface for a project named 'MY TEST PROJECT'. The top navigation bar includes 'PROJECTS', 'DASHBOARD', 'REPORTS', 'TIME', 'PROFILE', and 'ACCOUNT'. Below this, there are tabs for 'OVERVIEW', 'SETTINGS', 'MEMBERS', 'REPORTS', 'EXPORT', 'IMPORT', 'PROJECT PROFILE', and 'INTEGRATIONS'. The 'SETTINGS' tab is active. The 'General' section includes fields for 'Project Title' (My Test Project), 'Description', 'Account' (Time Tracking Test), and checkboxes for 'Allow Attachments' and 'Enable Tasks'. The 'Iterations and Velocity' section includes dropdowns for 'Start Iterations On' (Monday), 'Project Start Date', 'Project Time Zone' (GMT+01:00 Warsaw), 'Iteration Length' (1 weeks), 'Point Scale' (Fibonacci), 'Initial Velocity' (10), 'Velocity Strategy' (Average of 3 iterations), and 'Number of Done Iterations to Show' (12). The 'Access' section at the bottom has a checkbox for 'Allow API Access' which is checked and highlighted with a green box and a green circle labeled '2'. A green arrow points to the 'SETTINGS' tab in the top navigation bar, which is also highlighted with a green circle labeled '1'. A green text overlay reads 'Click here and then Projects tab and Settings "cog"'.

Running the Pivotal Tracker Import Wizard

Before you begin, please [back up](#) your JIRA data.

1. Log in to JIRA as a user with the **JIRA Administrators** global permission.
2. Choose



> **System**. Select **Import & Export > External System Import** to open the Import external projects page.

3. Select **Pivotal Tracker** to open the **Connect with Pivotal Tracker** page.
4. On the **Connect with Pivotal Tracker** page, specify the following:

Pivotal Username or Email	Specify the user account that JIRA will use to access issues on your Pivotal Tracker site.
Pivotal Password	Specify the password of the user (above).
Map user names (in expanded Advanced tab)	Select this checkbox if you want to modify the name details of Pivotal Tracker users (which would be associated with Pivotal Tracker issues) when these users are created in JIRA.
Use an existing configuration file (in expanded Advanced tab)	<p>Leave this checkbox cleared if you do not have a configuration file or if you want to create a new configuration file. Configuration files specify a mapping between fields in Pivotal Tracker and those in JIRA.</p> <p>Note:</p> <ul style="list-style-type: none"> • If you select this option, you will be asked to specify an Existing Configuration File. • If you do not select this option, then at the end of the Pivotal Tracker Import Wizard, JIRA will create a configuration file which you can use for subsequent Pivotal Tracker imports (for re-use at this step of the Pivotal Tracker Import Wizard).

- Click the **Next** button to proceed to the **Setup project mappings** step of the Pivotal Tracker Import Wizard.
- On the **Setup project mappings** page, select which Pivotal Tracker projects you wish to import into JIRA.
 - i** All Pivotal Tracker projects are selected by default, so clear the checkboxes under **Import** of the Pivotal Tracker projects you *do not* wish to import into JIRA.

For Pivotal Tracker projects you wish to import into JIRA, click in **Select a project** and then do either of the following:

 - Select **Create New** from the drop-down menu and in the resulting **Add A New Project** dialog box, type the following:
 - A new project **Name**.
 - A new project **Key**.
 - i** This will be used as the prefix for all issue IDs in your JIRA project.
 - The **Project Lead**.
 - Start typing the name (or key) of a project that already exists in JIRA or use the drop-down menu to select an existing JIRA project.
 - i** Only JIRA projects that use the **PT Workflow Scheme** (which is created with your first Pivotal Tracker import into JIRA) can be chosen from the **Select a project** list. The **PT Workflow Scheme** consists of the:
 - **PT Workflow** — mapped to all standard issue types.
 - **PT Subtask Workflow** — mapped to JIRA's sub-task issue type.
 - Tip:** If you have not yet performed a Pivotal Tracker import into JIRA but you would like to import your Pivotal Tracker issues into an existing JIRA project, consider doing the following:
 - Use the Pivotal Tracker Import Wizard to import your issues into a new JIRA project. Upon doing so, JIRA will create the **PT Workflow Scheme** and **PT Issue Type Scheme**. The **PT Issue Type Scheme** consists of additional issue types that do not exist in a default JIRA installation, such as **Chore** and **Release**.
 - (*Optional*) Delete this project if you do not intend to use it any further.
 - Apply the **PT Workflow Scheme** and **PT Issue Type Scheme** to the existing JIRA project you want to import your Pivotal Tracker issues into. (See [Defining a project](#) for details.)
 - Re-use the Pivotal Tracker Import Wizard to import your issues into this existing JIRA project.
- Click the **Next** button to proceed to the **Setup user mappings** step of the Pivotal Tracker Import Wizard.
 - i** If you did not select **Map user names** option above, skip to step 8. (The **Next** button will not be available.)

8. On the **Setup user mappings** step of the Pivotal Tracker Import Wizard, in the **Target value in JIRA** field:
 - Specify the *username* of a JIRA user to match Pivotal Tracker users to existing JIRA users.
 - Leave blank to add the Pivotal Tracker user's name details 'as is'. The user's Full Name in JIRA is derived from the Pivotal Tracker's username value and the JIRA username is derived from this Full Name (made lower-case).
 - Specify the Full Name in JIRA to change a Pivotal Tracker's user's name details. The JIRA username is derived from this Full Name (made lower-case).
9. Click the **Begin Import** button when you are ready to begin importing your Pivotal Tracker data into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.
 - Note:**
 - If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the Pivotal Tracker Import Wizard process.
 - If you need to import data from another Pivotal Tracker project or site with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a Pivotal Tracker configuration file, which you can use at the [first step](#) of the Pivotal Tracker Import Wizard.

Congratulations, you have successfully imported your Pivotal Tracker project(s) into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing Pivotal Tracker data into JIRA fields

The import process converts Pivotal Tracker data as follows:

Pivotal Tracker	JIRA	Import Notes
Project	Project	Each Pivotal Tracker project is imported into a new JIRA project. You can optionally import into an existing project if you have used the importer before.
Story	Issue	Pivotal Tracker story types are recreated in JIRA.
Summary	Summary	
Comments	Comments	
Attachments	Attachments	Attachments are extracted from the Pivotal Tracker database and saved to disk. The dates and user attaching the attachments will be retained.
Status	Status	JIRA will recreate the Pivotal Tracker workflow and statuses during import.
Labels	Labels	Pivotal Tracker labels with spaces are imported with underscores (JIRA does not support spaces in labels).
Story ID	External issue ID and External issue URL	JIRA will create these as custom fields.
Iterations	Fix Version/s	Past iterations in Pivotal are imported as released versions in JIRA.
Story Estimates	Story Points	
Order of stories	Rank	You will need to configure this custom field in JIRA after the import. If you are using JIRA Software, you may wish to activate issue ranking. This can be done either before or after importing your Pivotal Tracker data.

Time Tracker	Time Tracking	If you use time tracking in Pivotal this data will be automatically imported into a new JIRA issue type called 'Chore' with a Summary field value of "Placeholder for imported time tracking data".
User	User	<p>The importer will automatically create JIRA users for any Pivotal Tracker users who do not exist in JIRA.</p> <ul style="list-style-type: none"> • Passwords from Pivotal Tracker are not imported (as they are hashed in the database). Users from Pivotal Tracker will need to get their passwords emailed to them. • 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. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created.
User Roles	Project Roles	Viewer = User ; Member = Developers ; Owner = Administrators

Importing data from Bugzilla

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **Bugzilla** by connecting to a live Bugzilla database.

i Our main website highlights some top reasons why people [migrate from Bugzilla to JIRA](#). Version 4.1 or later of the JIRA Importers plugin is compatible with Bugzilla 2.20 to 4.4.4. Users of older Bugzilla versions will need to first upgrade the Bugzilla database tables to a supported version, using Bugzilla's `checksetup.pl` script. The JIRA Importers plugin requires that your Bugzilla database is MySQL, PostgreSQL or Microsoft SQL Server.

i JIRA is able to import data from Bugzilla 2.20 **only** if it's using a [supported database](#).

i JIRA does not bundle the MySQL driver anymore. If the Bugzilla data is located in a MySQL database, follow the instructions in [Connecting JIRA to MySQL](#) to install the MySQL database driver before attempting to import from Bugzilla

The Bugzilla import process consists of simply running the Bugzilla Import Wizard ([below](#)).

- You can choose to map individual fields and field values during the import process, some of which are mandatory.
- At the end of the Bugzilla Import Wizard, you will be given the option of creating a Bugzilla configuration file, which contains the settings you configured while running through the Bugzilla Import Wizard. This is useful if you need to test your Bugzilla import on a test JIRA server first before performing the import on a production system.

⚠ Please note:

- JIRA's character encoding is set to UTF-8 by default. If, however, your JIRA installation's character encoding is set to something other than UTF-8, you may encounter problems with importing data from Bugzilla. For more information, please refer to [JIM-5](#). Importing Bugzilla data into a non-UTF-8 JIRA installation is not supported.

Running the Bugzilla Import Wizard

Before you begin, please [back up](#) your JIRA data.

On this page:

- [Running the Bugzilla Import Wizard](#)
- [Tips for importing Bugzilla data into JIRA fields](#)

1. In your Bugzilla system, run the Bugzilla 'Sanity Check' to ensure your data is error-free.
2. Log in to JIRA as a user with the **JIRA Administrators** [global permission](#).
3. Choose 
 - > **System**. Select **Import & Export > External System Import** to open the Import external projects page.
4. Select **Bugzilla** to open the **Bugzilla Import Wizard: Setup** page.
5. On the **Bugzilla Import Wizard: Setup** page, complete the following fields/options:

Bugzilla URL	Specify the URL of your Bugzilla site. This is the URL you would normally use to access Bugzilla through a web browser.
Specify credentials	Select this checkbox if you want to import Bugzilla issues into JIRA, which require user credentials on your Bugzilla site to access them. Selecting this checkbox reveals/hides the Bugzilla Login and Bugzilla Password fields, into which you should specify these user credentials. If your Bugzilla site requires credentials and you do not specify them here, Bugzilla "Big File" attachments will not be imported.
Database type	Select the type of database that your Bugzilla installation uses: <ul style="list-style-type: none"> • PostgreSQL • Microsoft SQL Server • MySQL
Hostname	Specify the hostname or IP address of the server running your Bugzilla site's database server.
Port	Specify the TCP/IP port that the Bugzilla site's database server is listening on. This field is automatically populated with the default port value based on the Database Type you choose above.
Database	Specify the name of your Bugzilla database (into which Bugzilla saves its data). This database name can usually be found in the 'localconfig' file in Bugzilla's root directory, for example, <code>/etc/bugzilla/</code>
Username	Specify the database user that Bugzilla uses to connect to its database. This database user can usually be found in the 'localconfig' file in Bugzilla's root directory, for example, <code>/etc/bugzilla/</code>
Password	Specify the password of the database user (above) that Bugzilla uses to connect to its database. This password can usually be found in the 'localconfig' file in Bugzilla's root directory, for example, <code>/etc/bugzilla/</code>
Use an existing configuration file	Leave this checkbox cleared if you do not have a configuration file or if you want to create a new configuration file. Configuration files specify a mapping between fields in Bugzilla and those in JIRA. <p>Note:</p> <ul style="list-style-type: none"> • If you select this option, you will be asked to specify an Existing Configuration File. • If you do not select this option, then at the end of the Bugzilla Import Wizard, JIRA will create a configuration file which you can use for subsequent Bugzilla imports (for re-use at this step of the Bugzilla Import Wizard).

JDBC connection parameters (in expanded Advanced tab)	<p>The Bugzilla Import Wizard will construct a JDBC-based database URL from the Bugzilla database server details you specify above. JIRA uses this URL to connect to and import issues from Bugzilla. If you need to specify any additional connection parameters to your Bugzilla database, specify them here.</p> <p>i If you chose MySQL (above), the Bugzilla Import Wizard will add several additional connection parameters by default.</p>
--	--

6. Click the **Next** button to proceed to the **Setup project mappings** step of the Bugzilla Import Wizard.
7. On the **Setup project mappings** page, select which Bugzilla projects you wish to import into JIRA.
 - i** All Bugzilla projects are selected by default, so clear the checkboxes under **Import** of the Bugzilla projects you *do not* wish to import into JIRA.

For Bugzilla projects you wish to import into JIRA, click in **Select a project** and then do either of the following:

 - Start typing the name (or key) of a project that already exists in JIRA or use the drop-down menu to select an existing JIRA project.
 - Select **Create New** from the drop-down menu and in the resulting **Add A New Project** dialog box, type the following:
 - a. A new project **Name**
 - b. A new project **Key**
 - i** This will be used as the prefix for all issue IDs in your JIRA project.
 - c. The **Project Lead**.
8. Click the **Next** button to proceed to the **Setup custom fields** step of the Bugzilla Import Wizard.
 - i** This step will almost always appear because at least one Bugzilla field is not likely match an existing JIRA field.
9. On the **Setup custom fields** page, for each **External field** in Bugzilla which the Bugzilla Import Wizard cannot match to an existing JIRA field, you can choose to either:
 - have the Bugzilla Import Wizard automatically create new [custom fields in JIRA](#) based on the names of Bugzilla's fields. This is the default option - whereby the names of the JIRA custom fields to be automatically created appear in the **JIRA field** drop-down lists.
 - create your own custom fields in JIRA to map data from Bugzilla's fields. To do this, choose **Other** from the **JIRA field** drop-down list and specify the name of your custom field in the new field appearing immediately below **Other**.
 - i** For more information about matching Bugzilla fields to JIRA fields, see [Tips for importing Bugzilla data into JIRA fields](#) below.
10. Click the **Next** button to proceed to the **Setup field mappings** step of the Bugzilla Import Wizard.
11. On the **Setup field mappings** page, if there **External fields** in Bugzilla whose values you wish to modify *before* they are imported into JIRA, select the **Map field value** checkboxes next to the appropriate fields.
 - i** Please note that it is mandatory to map Bugzilla's **bug_status** (i.e. **Status**) field to specific JIRA **Status** field values as the JIRA **Status** field is an integral part of [JIRA workflows](#).
 - Other **External fields** in Bugzilla which are likely to appear on the **Setup field mappings** page are:

External field in Bugzilla	Not choosing the 'Map field value' checkbox
login_name	The Bugzilla Import Wizard will automatically map Bugzilla usernames to JIRA usernames (lowercase).
priority	The Bugzilla Import Wizard will automatically create missing values in JIRA and will ensure that the issues are migrated with the correct priority (e.g. "Normal" in Bugzilla to newly-created "Normal" in JIRA).
resolution	The importer will create corresponding Resolutions in JIRA instead of using the existing ones.

- Select the appropriate JIRA **Workflow Scheme** in that will be used by the Bugzilla issues you will import into your JIRA project.
 - i** If you are importing your Bugzilla issues into an existing JIRA project, ensure that you

choose the JIRA workflow scheme used by that existing JIRA project.

12. Click the **Next** button to proceed to the **Setup value mappings** step of the Bugzilla Import Wizard.
13. On the **Setup value mappings** page, specify JIRA field values for each Bugzilla field value (as detected by the Bugzilla Import Wizard).
 - i** Any fields whose **Map field value** checkboxes were selected in the previous step of the Bugzilla Import Wizard will be presented on this page, including the mandatory **bug_status** Bugzilla field.
14. Click the **Next** button to proceed to the **Setup links** step of the Bugzilla Import Wizard.
15. On the **Setup links** page, specify the JIRA link type for each Bugzilla link type (as detected by the Bugzilla Import Wizard). To learn more about JIRA link types, see [Configuring issue linking](#).
16. Click the **Begin Import** button when you are ready to begin importing your Bugzilla data into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.

i **Note:**

- If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the Bugzilla Import Wizard process.
- If you need to import data from another Bugzilla product/project or site with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a Bugzilla configuration file, which you can use at the [first step](#) of the Bugzilla Import Wizard.

Congratulations, you have successfully imported your Bugzilla projects into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing Bugzilla data into JIRA fields

During the import process, the following data is copied from the Bugzilla database into JIRA:

In Bugzilla	In JIRA	Import Notes
Product	Project	Bugzilla data is imported on a per-project basis. You can either specify an existing JIRA project as the target, or the importer will automatically create a project(s) for you at time of import. See Defining a project for more information on JIRA projects.
External Project	Project Category	
Version	Affects Version	
Component	Component	You can choose to have the importer automatically create your Bugzilla component(s) in JIRA, or choose to have bugs imported into no component in JIRA.
Milestone	Fix Version	Versions are imported from Bugzilla (if you choose) and are set to the Un-Released and Un-Archived state.
Bug	Issue	Every Bugzilla bug becomes a JIRA issue of type 'Bug', with one exception: a Bugzilla issue with severity 'Enhancement' becomes a JIRA issue of type 'Improvement' and priority 'Major'.
ID	External issue ID	Each imported issue will be given a new JIRA ID, and the old Bugzilla ID will be saved into a JIRA custom field called 'External issue ID'. This custom field is searchable, so you can search for JIRA issues by their old Bugzilla ID. If you don't need this custom field, delete it or 'hide' it (as described in Specifying field behavior).
Summary	Summary	
Description	Description	
Comments	Comments	

Attachments	Attachments	Attachments are extracted from the Bugzilla database and saved to disk. To specify the location on disk, see Configuring file attachments .
Priority	Priority (or a custom field)	You can choose to map one of either the Bugzilla Priority field or the Bugzilla Severity field (see above) to the built-in JIRA Priority field, and the other to a custom field. (Alternatively, you can choose to map both the Bugzilla Priority field and the Bugzilla Severity field to JIRA custom fields.) When importing into the JIRA Priority field, you can configure mapping of specific Bugzilla values to specific JIRA values.
Severity	Priority (or a custom field)	You can choose to map one of either the Bugzilla Priority field (above) or the Bugzilla Severity field to the built-in JIRA Priority field, and the other to a custom field. (Alternatively, you can choose to map both the Bugzilla Priority field and the Bugzilla Severity field to JIRA custom fields.) When importing into the JIRA Priority field, you can configure mapping of specific Bugzilla values to specific JIRA values.
Status	Status	You can configure mapping of specific Bugzilla values to specific JIRA values. <ul style="list-style-type: none"> The JIRA 'Status' field is integral to JIRA workflow. To learn more, please see Working with workflows and Managing your workflows.
Resolution	Resolution	You can configure mapping of specific Bugzilla values to specific JIRA values.
Duplicates Depends on Blocks	Link	You can configure mapping of specific Bugzilla link types to JIRA link types. <ul style="list-style-type: none"> In JIRA, you can configure different types of links (please see Configuring issue linking).
Work History	Work Log	Each Bugzilla worklog report will appear in JIRA as a separate worklog entry.
Estimated	Original Estimate	See Configuring time tracking .
Remaining	Remaining Estimate	See Configuring time tracking .
Logged	Time Spent	See Configuring time tracking .
Votes	Voters	If a user has voted one or more times for a Bugzilla issue, a JIRA vote is stored for that user.
CC List	Watchers	

User	User	<p>You can choose to have the importer automatically create JIRA users for any Bugzilla users who do not already exist in JIRA.</p> <ul style="list-style-type: none"> • Users who interacted with the Bugzilla system will be created as active accounts in JIRA. Other users will be imported into a special group called "bugzilla-import-unused-users" and will be deactivated. • 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. • 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). • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created.
Status Whiteboard	Status Whiteboard	A JIRA custom field called 'Status Whiteboard' will be created.
Other fields	Custom fields	If your Bugzilla system contains any custom fields, you can choose to map them to specific JIRA custom field(s). If your custom fields don't yet exist in JIRA, the importer can automatically create them for you.

Importing data from FogBugz On Demand

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **FogBugz On Demand**, a 'Software as a Service' (SaaS) issue tracker application.

i Our main website highlights some top reasons why people [migrate from FogBugz to JIRA](#). Version 3.1 or later of the JIRA Importers Plugin is required.

On this page:

- [Running the FogBugz On Demand Import Wizard](#)
- [Tips for importing FogBugz On Demand data into JIRA fields](#)

These instructions refer to **FogBugz On Demand**, which is a SaaS implementation of FogBugz.

Running the FogBugz On Demand Import Wizard

Before you begin: If your JIRA installation has existing data — [Back up](#) your existing JIRA data.

Tip: FogBugz On Demand supports hierarchical issues. During the FogBugz On Demand Import Wizard, you are given the option to recreate this issue hierarchy through JIRA issue links. Hence, before commencing the FogBugz On Demand Import Wizard, you may wish to [configure a custom issue link](#) to replicate this hierarchy — for example:

- **Name** — 'Hierarchy'
- **Outward Link Description** — 'parent of'

- **Inward Link Description** — 'child of'

To import issues FogBugz On Demand:

1. Log in to JIRA as as a user with the **JIRA Administrators** [global permission](#).
2. Choose



> **System**. Select **Import & Export > External System Import** to open the Import external projects page.

3. Select **FogBugz On Demand** to open the **Connect with FogBugz** page.
4. On the **Connect with FogBugz** page, complete the following fields:

FogBugz On Demand URL	Specify the URL of your FogBugz On Demand site. This is the URL you would normally use to access FogBugz On Demand through a web browser. This is usually of the format <code>http://myfogbugzondemand.fogbugz.com</code>
FogBugz Username	Specify the user account that JIRA will use to access issues on your FogBugz On Demand site.
FogBugz Password	Specify the password of the user (above).

5. Click the **Next** button to proceed to the **Setup project mappings** step of the FogBugz On Demand Import Wizard.
6. On the **Setup project mappings** page, select which FogBugz On Demand projects you wish to import into JIRA.
7. All FogBugz On Demand projects are selected by default, so clear the checkboxes under **Import** of the FogBugz On Demand projects you *do not* wish to import into JIRA. For FogBugz On Demand projects you wish to import into JIRA, click in **Select a project** and then do either of the following:
 - Start typing the name (or key) of a project that already exists in JIRA or use the drop-down menu to select an existing JIRA project.
 - Select **Create New** from the drop-down menu and in the resulting **Add A New Project** dialog box, type the following:
 - a. A new project **Name**
 - b. A new project **Key**
 - This will be used as the prefix for all issue IDs in your JIRA project.
 - c. The **Project Lead**.
8. Click the **Next** button to proceed to the **Setup field mappings** step of the FogBugz On Demand Import Wizard.
9. On the **Setup field mappings** page, if there **External fields** in FogBugz On Demand whose values you wish to modify *before* they are imported into JIRA, select the **Map field value** checkboxes next to the appropriate fields.
 - Please note that it is mandatory to map FogBugz On Demand's **sStatus** (i.e. **Status**) field to specific JIRA **Status** field values as the JIRA **Status** field is an integral part of [JIRA workflows](#).
 - The FogBugz On Demand field **sStatus (Resolution)** (i.e. **Resolution**), which will be mapped to the JIRA **Resolution** field, may also appear on this page.
 - Select the appropriate JIRA **Workflow Scheme** in that will be used by the FogBugz On Demand issues you will import into your JIRA project.
 - If you are importing your FogBugz On Demand issues into an existing JIRA project, ensure that you choose the JIRA workflow scheme used by that existing JIRA project. Otherwise, your import may not complete successfully.
10. Click the **Next** button to proceed to the **Setup value mappings** step of the FogBugz On Demand Import Wizard.
11. On the **Setup value mappings** page, specify JIRA field values for each FogBugz On Demand field value (as detected by the FogBugz On Demand Import Wizard).
 - Any fields whose **Map field value** checkboxes were selected in the previous step of the FogBugz On Demand Import Wizard will be presented on this page, including the mandatory **sStatus** FogBugz On Demand field.
12. Click the **Next** button to proceed to the **Setup links** step of the FogBugz On Demand Import Wizard.
13. On the **Setup links** page, specify how want to map FogBugz On Demand's Parent / Subcase relationships through a JIRA issue link. To learn more about JIRA link types, please see [Configuring](#)

[issue linking](#).

✔ You may wish to choose the 'Hierarchy' custom issue link you created before running the FogBugz On Demand Import Wizard.

- Click the **Begin Import** button when you are ready to begin importing your FogBugz On Demand data into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.

Note: If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the FogBugz On Demand Import Wizard process.

Congratulations, you have successfully imported your FogBugz On Demand projects into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing FogBugz On Demand data into JIRA fields

The import process converts FogBugz On Demand data as follows:

FogBugz On Demand	In JIRA	Import Notes
Project	Project	FogBugz data is imported on a per-project basis. You can either specify an existing JIRA project as the target, or the importer will automatically create a project(s) for you at time of import. See Defining a project for more information about JIRA projects.
Area	Component	You can choose to have the importer automatically create your FogBugz components in JIRA, or choose to have bugs imported into no component in JIRA.
Milestone	Fix Version	Versions are imported from FogBugz (if you choose). After importing, you can manually set appropriate versions to the Released state in JIRA if you wish.
Case	Issue	Every FogBugz case becomes a JIRA issue.
Case ID ixBug	External issue ID and External issue URL	Each imported issue ('case') will be given a new JIRA ID, and the old FogBugz ID will be saved into a JIRA custom field called 'External issue ID'. This custom field is searchable, so you can search for JIRA issues by their old FogBugz ID. If you don't need this custom field, delete it or 'hide' it (as described in Specifying field behavior).
Summary	Summary	
Comments	Comments	FogBugz allows for links to other issues to be automatically generated by using the format "bug issued" 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".
Attachments	Attachments	Attachments are extracted from the FogBugz database and saved to disk. 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. To specify the location on disk, see Configuring file attachments .
Category	Issue Type	You can configure mapping of specific Case Categories to specific Issue Types.
Priority	Priority	You can configure mapping of specific FogBugz values to specific JIRA values.

Status	Status	<p>You can configure mapping of specific FogBugz values to specific JIRA values, provided you create your workflows in JIRA before running the importer.</p> <ul style="list-style-type: none"> • The JIRA Status field is integral to JIRA workflow. • To create a JIRA workflow, please see Working with workflows. • To create a JIRA workflow scheme (which you can then associate with appropriate projects and Issue Types), please see Managing your workflows.
Resolution	Resolution	You can configure mapping of specific FogBugz values to specific JIRA values.
Duplicates BugRelations		They are not imported due to limitations of FogBugz Remote API
Computer	Computer	The FogBugz Computer field is imported into a JIRA Custom Field called 'Computer'.
Customer Email	Customer Email	The FogBugz Customer Email field is imported into a JIRA Custom Field called 'Customer Email'.
User	User	<p>You can choose to have the importer automatically create JIRA users for any FogBugz users who do not already exist in JIRA.</p> <ul style="list-style-type: none"> • Users who interacted with the FogBugz system will be created as active accounts in JIRA. Other users will be imported into a special group called "fogbugz-import-unused-users" and will be deactivated. • Passwords from FogBugz are not imported (as they are hashed in the database). Users from FogBugz will need to get their passwords emailed to them the first time they log into JIRA. • Users with no real name stored in FogBugz will get the portion of their email address (login name) before the "@" character as their Full Name 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. • 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. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created.
<i>Other fields</i>	Custom fields	If your FogBugz system contains any custom fields, you can choose to map them to specific JIRA custom field(s). If your custom fields don't yet exist in JIRA, the importer can automatically create them for you. Please note that the <i>FogBugz Custom Field plugin</i> is not supported.

Importing data from FogBugz for your Server

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **FogBugz for Your Server** by connecting to a live FogBugz for Your Server database.

 Our main website highlights some top reasons why people [migrate from FogBugz to JIRA](#). Version 4.2 or later of the JIRA Importers plugin is compatible with Fogbugz for Your Server versions 7.3.6 to 8.8.39. The JIRA Importers plugin requires that your FogBugz for Your Server database is MySQL, Microsoft SQL Server or Microsoft SQL Server Express.

On this page:

- Running the FogBugz for your Server Import Wizard
- Tips for importing FogBugz for your Server data into JIRA fields

The **FogBugz for Your Server** import process consists of simply running the FogBugz Import Wizard (below):

- You can choose to map individual fields and field values during the import process, some of which are mandatory.
- At the end of the FogBugz Import Wizard, you will be given the option of creating a FogBugz configuration file, which contains the settings you configured while running through the FogBugz Import Wizard. This is useful if you need to test your FogBugz import on a test JIRA server first before performing the import on a production system.

i These instructions refer to a **FogBugz for Your Server**, which is an installable implementation of FogBugz that operates behind your firewall. To import from a **FogBugz On Demand** (SaaS) issue tracker site, please follow the instructions for [here](#).

Running the FogBugz for your Server Import Wizard

Before you begin, please [back up](#) your JIRA data.

1. Log in to JIRA as a user with the **JIRA Administrators** [global permission](#).
2. Choose  **> System**. Select **Import & Export > External System Import** to open the Import external projects page.
3. Select **FogBugz for Your Server** to open the **FogBugz Import Wizard: Setup** page.

FogBugz Setup

Setup
Projects
Custom Fields
Fields
Values
Links

Database Type *

Hostname *
Hostname or IP address of the database server.

Port *
TCP Port Number for the database server.

Database *
Database to connect to.

Username *
The username used to access the database.

Password
The password used to access the database.

Use an existing configuration file
If you have used this importer before, you may have saved a previous configuration file. You can use that here to save time.

> **Advanced**

i Passwords will not be imported. Users will have to create new password at first login.

[Powered by Atlassian](#) | [Terms of Use](#) | [Answers](#) | [Got Feedback?](#)

4. On the **FogBugz Setup** page, complete the following fields/options:

Database Type	Select the type of database that your FogBugz for Your Server installation uses: <ul style="list-style-type: none"> PostgreSQL Microsoft SQL Server MySQL
Hostname	Specify the hostname or IP address of the server running your FogBugz site's database server.
Port	Specify the TCP/IP port that the FogBugz site's database server is listening on. This field is automatically populated with the default port value based on the Database Type you choose above.
Database	Specify the name of your FogBugz database (into which FogBugz for Your Server saves its data). If you need to specify an instance ID for your database, do so using the syntax <code>fogbugz ; instance=sqlexpress</code> (where <code>fogbugz</code> is the name of your FogBugz database and <code>sqlexpress</code> is your FogBugz database's instance ID). The database name can usually be found in the Windows registry. See http://bugs.movabletype.org/help/topics/setup/WindowsWhatSetupDoes.html and then search for 'Initialize Registry Settings' (for details on how to access the relevant registry keys and values).

Username	Specify the database user that FogBugz uses to connect to its database.
Password	Specify the password of the database user (above) that FogBugz uses to connect to its database.
Use an existing configuration file	<p>Leave this check box cleared if you do not have a configuration file or if you want to create a new configuration file. Configuration files specify a mapping between fields in FogBugz for Your Server and those in JIRA.</p> <p>Note:</p> <ul style="list-style-type: none"> • If you select this option, you will be asked to specify an Existing Configuration File. • If you do not select this option, then at the end of the FogBugz Import Wizard, JIRA will create a configuration file which you can use for subsequent imports (for re-use at this step of the FogBugz Import Wizard).
JDBC connection parameters (in expanded Advanced tab)	<p>The FogBugz Import Wizard will construct a JDBC-based database URL from the FogBugz database server details you specify above. JIRA uses this URL to connect to and import issues from FogBugz for Your Server. If you need to specify any additional connection parameters to your FogBugz database, specify them here.</p> <p>If you chose MySQL (above), the FogBugz Import Wizard will add several additional connection parameters by default.</p>

5. Click the **Next** button to proceed to the **Set up project mappings** step of the FogBugz Import Wizard.
6. On the **Set up project mappings** page, select which FogBugz projects you wish to import into JIRA.
7.  All projects are selected by default, so clear the check boxes under **Import** of the FogBugz projects you *do not* wish to import into JIRA.
For FogBugz projects you wish to import into JIRA, click in **Select a project** and then do either of the following:
 - Start typing the name (or key) of a project that already exists in JIRA or use the dropdown menu to select an existing JIRA project.
 - Select **Create New** from the dropdown menu and in the resulting **Add A New Project** dialog box, type the following:
 - a. A new project **Name**
 - b. A new project **Key**
 -  This will be used as the prefix for all issue IDs in your JIRA project.
 - c. The **Project Lead**.
8. Click the **Next** button to proceed to the **Set up custom fields** step of the FogBugz Import Wizard.
9. On the **Set up custom fields** page, for each **External field** in FogBugz which the FogBugz Import Wizard cannot match to an existing JIRA field, you can choose to either:
 - have the FogBugz Import Wizard automatically create new **custom fields in JIRA**, based on the names of FogBugz's fields. This is the default option - whereby the names of the JIRA custom fields to be automatically created appear in the **JIRA field** dropdown lists.
 - create your own custom fields in JIRA to map data from FogBugz's fields. To do this, choose **Other** from the **JIRA field** dropdown list and specify the name of your custom field in the new field appearing immediately below **Other**.
10. Click the **Next** button to proceed to the **Set up field mappings** step of the FogBugz Import Wizard.
11. On the **Set up field mappings** page, if there **External fields** in FogBugz whose values you wish to modify *before* they are imported into JIRA, select the **Map field value** check boxes next to the appropriate fields.
 -  Please note that it is mandatory to map FogBugz's **sStatus** (i.e. **Status**) field to specific JIRA **Status** field values as the JIRA **Status** field is an integral part of **JIRA workflows**.
 - Other **External fields** in FogBugz which are likely to appear on the **Set up field mappings** page are:

External field in FogBugz	Not choosing the 'Map field value' check box
----------------------------------	---

sCategory	The FogBugz Import Wizard will automatically create missing issue types in JIRA and will ensure that the issues are migrated with the correct issue type.
sCustomerEmail	The FogBugz Import Wizard will not map values for this field.
sComputer	The FogBugz Import Wizard will not map values for this field.
sFullName	The FogBugz Import Wizard will automatically map FogBugz usernames to JIRA usernames (lowercase).
sPriority	The FogBugz Import Wizard will automatically create missing values in JIRA and will ensure that the issues are migrated with the correct priority (e.g. "Normal" in FogBugz to newly-created "Normal" in JIRA).
sStatus (Resolution)	The importer will create corresponding Resolutions in JIRA instead of using the existing ones.

- Select the appropriate JIRA **Workflow Scheme** in that will be used by the FogBugz issues you will import into your JIRA project.
 - **i** If you are importing your FogBugz issues into an existing JIRA project, ensure that you choose the JIRA workflow scheme used by that existing JIRA project.
- 12. Click the **Next** button to proceed to the **Set up value mappings** step of the FogBugz Import Wizard.
- 13. On the **Set up value mappings** page, specify JIRA field values for each FogBugz field value (as detected by the FogBugz Import Wizard).
 - **i** Any fields whose **Map field value** check boxes were selected in the previous step of the FogBugz Import Wizard will be presented on this page, including the mandatory **sStatus** FogBugz field.
- 14. Click the **Next** button to proceed to the **Set up links** step of the FogBugz Import Wizard.
- 15. On the **Set up links** page, specify the JIRA link type for each FogBugz link type (as detected by the FogBugz Import Wizard). To learn more about JIRA link types, please see [Configuring issue linking](#).
- 16. Click the **Begin Import** button when you are ready to begin importing your FogBugz data into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.
 - **i Note:**
 - If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the FogBugz Import Wizard process.
 - If you need to import data from another FogBugz product/project or site with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a FogBugz configuration file, which you can use at the **first step** of the FogBugz Import Wizard.

Congratulations, you have successfully imported your FogBugz projects into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing FogBugz for your Server data into JIRA fields

During the import process, the following data is copied from the FogBugz Server database into JIRA:

In FogBugz	In JIRA	Import Notes
Project	Project	FogBugz data is imported on a per-project basis. You can either specify an existing JIRA project as the target, or the importer will automatically create a project(s) for you at time of import. See Defining a project for more information about JIRA projects.
Area	Component	You can choose to have the importer automatically create your FogBugz components in JIRA, or choose to have bugs imported into no component in JIRA.

Milestone	Fix Version	Versions are imported from FogBugz (if you choose). After importing, you can manually set appropriate versions to the Released state in JIRA if you wish.
Case	Issue	Every FogBugz case becomes a JIRA issue.
Case ID ixBug	Bug Import ID	Each imported issue ('case') will be given a new JIRA ID, and the old FogBugz ID will be saved into a JIRA custom field called 'Bug Import ID'. This custom field is searchable, so you can search for JIRA issues by their old FogBugz ID. If you don't need this custom field, delete it or 'hide' it (as described in Specifying field behavior).
Summary	Summary	
Comments	Comments	FogBugz allows for links to other issues to be automatically generated by using the format "bug issued" 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".
Attachments	Attachments	Attachments are extracted from the FogBugz database and saved to disk. 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. To specify the location on disk, see Configuring file attachments .
Category	Issue Type	You can configure mapping of specific Case Categories to specific Issue Types.
Priority	Priority	You can configure mapping of specific FogBugz values to specific JIRA values.
Status	Status	You can configure mapping of specific FogBugz values to specific JIRA values, provided you create your workflows in JIRA before running the importer. <ul style="list-style-type: none"> • The JIRA 'Status' field is integral to JIRA workflow. • To create a JIRA workflow, please see Working with workflows. • To create a JIRA workflow scheme (which you can then associate with appropriate projects and Issue Types), please see Managing your workflows.
Resolution	Resolution	You can configure mapping of specific FogBugz values to specific JIRA values.
Duplicates BugRelations	Links	You can configure mapping of specific FogBugz link types to JIRA link types. <ul style="list-style-type: none"> • In JIRA, you can configure different types of links (please see Configuring issue linking).
Computer	Computer	The FogBugz Computer field is imported into a JIRA Custom Field called 'Computer'.
Customer Email	Customer Email	The FogBugz Customer Email field is imported into a JIRA Custom Field called 'Customer Email'.

User	User	<p>You can choose to have the importer automatically create JIRA users for any FogBugz users who do not already exist in JIRA.</p> <ul style="list-style-type: none"> • Users who interacted with the FogBugz system will be created as active accounts in JIRA. Other users will be imported into a special group called "fogbugz-import-unused-users" and will be deactivated. • Passwords from FogBugz are not imported (as they are hashed in the database). Users from FogBugz will need to get their passwords emailed to them the first time they log into JIRA. • Users with no real name stored in FogBugz will get the portion of their email address (login name) before the "@" character as their Full Name 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. • 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. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created.
Other fields	Custom fields	<p>If your FogBugz system contains any custom fields, you can choose to map them to specific JIRA custom field(s). If your custom fields don't yet exist in JIRA, the importer can automatically create them for you. Please note that the <i>FogBugz Custom Field plugin</i> is not supported.</p>

Importing data from Trac

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **Trac** from a compressed Trac environment.

 Our main website highlights some top reasons why people [migrate from Trac to JIRA](#). Version 2.6.1 or later of the JIRA Importers Plugin is compatible with Trac versions 0.12.2 to 1.0.1.

On this page:

- [Preparing Trac data for import into JIRA](#)
- [Running the Trac Import Wizard](#)
- [Tips for importing Trac data into JIRA fields](#)

Preparing Trac data for import into JIRA

Compress your Trac environment:

1. Access your Trac environment.
2. If you use SQLite (the Trac default), PostgreSQL or MySQL for your Trac database, ensure your database URL (defined in Trac's `conf/trac.ini` file) is also reachable from JIRA server (using 'localhost' or a UNIX socket will not work).
3. Zip the contents of Trac Environment without any leading directories.

Running the Trac Import Wizard

Before you begin, please [back up](#) your JIRA data.

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

2. Select **Administration > System > Import & Export > External System Import**.
3. Select **Trac** to open the **Trac Import Wizard: Setup** page.
4. On the **Trac Import Wizard: Setup** page, select your compressed Trac environment file, which you prepared [above](#).
5. Leave the **Use an existing configuration file** checkbox cleared if you do not have a configuration file or if you want to create a new configuration file. Configuration files specify a mapping between fields in Trac and those in JIRA.
 - If you select this option, you will be asked to specify an **Existing Configuration File**.
 - If you do not select this option, then at the end of the Trac Import Wizard, JIRA will create a configuration file which you can use for subsequent Trac imports (for re-use at this step of the Trac Import Wizard).
6. Click the **Next** button to proceed to the **Setup project mappings** step of the Trac Import Wizard.
7. On the **Setup project mappings** page, select which Trac projects you wish to import into JIRA.
 - Start typing the name (or key) of a project that already exists in JIRA or use the drop-down menu to select an existing JIRA project.
 - Select **Create New** from the drop-down menu and in the resulting **Add A New Project** dialog box, type the following:
 - i. A new project **Name**.
 - ii. A new project **Key**.
 -  This will be used as the prefix for all issue IDs in your JIRA project.
 - iii. The **Project Lead**.
8. Click the **Next** button to proceed to the **Setup custom fields** step of the Trac Import Wizard.
 -  This step will almost always appear because at least one Trac field is not likely match an existing JIRA field.
9. **Custom Fields:** If your Trac system contains any custom fields, you can either choose to import into an existing JIRA custom field or have the importer automatically create a new custom field in JIRA.
10.  Regardless of whether you specify mapping, the importer will automatically create a JIRA custom field for each extra Trac field, unless you un-check the '**Create new custom fields**' option on the final '**Import Data**' screen (see [Screenshot 2](#) below).
11. **Field Value Mappings:**
 - **'Priority' field** — If you don't specify mappings, the importer will automatically create missing values in JIRA and will ensure that the issues are migrated with the correct priority
 - **Usernames** — If you don't specify mapping, the importer will automatically map Trac usernames to JIRA usernames (lowercase).
 -  Regardless of whether you specify mapping, JIRA will automatically create usernames for missing users.
 - **'Status' field** — It is mandatory to map the Trac '**Status**' field to specific values of the JIRA '**Status**' field, as the JIRA '**Status**' field is integral to JIRA workflow (to learn more, please see [Working with workflows](#) and [Managing your workflows](#)).
 - **'Resolution' field** — If you don't specify mapping, the importer will create corresponding Resolutions in JIRA instead of using the existing ones.
 - **'Maximum issues and failures'** — If you wish, specify a maximum number of failed issues after which the importer will stop. If you want the import to continue regardless of any failures, leave this field blank. If your Trac instance has a large number of issues, it's generally a good idea to run first the importer on a limited number of issues (e.g. 100), then manually inspect the imported issues to confirm whether your configuration file was specified correctly. When the results are satisfactory, you can run the import with no limit.
12. The importer will display updates as the import progresses, then a success message when the import is complete. You can download the import log if you wish.

Congratulations, you have successfully imported your Trac projects into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing Trac data into JIRA fields

The import process converts Trac data as follows:

In Trac	In JIRA	Import Notes
---------	---------	--------------

Project Environment	Project	Each Trac Environment is imported as a JIRA project. You can either specify an existing JIRA project as the target, or the importer will automatically create a project for you at time of import.
Ticket Type	Issue Type	You can configure mapping of Trac Ticket Types to specific JIRA Issue Types.
Ticket #	External Issue ID	The Trac Ticket number is captured in a JIRA custom field. The import is not designed to have the JIRA issue number match the Trac ticket number.
Status	Status	You can configure mapping of specific Trac values to specific JIRA values.
Summary	Summary	
Description	Description	
Versions	Versions	Versions are imported from Trac (if you choose), and are set to the Un-Released and Un-Archived state.
Component	Components	You can choose to have the importer automatically create your Trac components in JIRA, or choose to have bugs imported into no component in JIRA.
Comments	Comments	
Priority	Priority (or a custom field)	You can choose to map one of either the Trac Priority field or the Trac Severity field (see below) to the built-in JIRA Priority field, and the other to a custom field. (Alternatively, you can choose to map both the Trac Priority field and the Trac Severity field to JIRA custom fields.) When importing into the JIRA Priority field, you can configure mapping of specific Trac values to specific JIRA values.
Severity	Priority (or a custom field)	You can choose to map one of either the Trac Priority field or the Trac Severity field (see below) to the built-in JIRA Priority field, and the other to a custom field. (Alternatively, you can choose to map both the Trac Priority field and the Trac Severity field to JIRA custom fields.) When importing into the JIRA Priority field, you can configure mapping of specific Trac values to specific JIRA values.
Milestone	Milestone	JIRA will create this as a custom field.
Attachments	Attachments	Attachments are extracted from the Trac Environment and saved to disk. To specify the location on disk, see Configuring file attachments .
Resolution	Resolution	You can configure mapping of specific Trac values to specific JIRA values.
CC	Watcher	
Keywords	Labels	

User	User	<p>The importer will automatically create JIRA users for any Trac users who do not exist in JIRA.</p> <ul style="list-style-type: none"> • Passwords from Trac are not imported. Users from Trac will need to get their passwords emailed to them. • 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. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created.
Other fields	Custom fields	<p>If your Trac system contains any custom fields, you can choose to map them to specific JIRA custom fields. If your custom fields don't yet exist in JIRA, the importer can automatically create them for you.</p>

Importing data from Redmine

The [JIRA Redmine Importer plugin](#) allows you to import data from the **Redmine Issue Tracker** application into your local JIRA instance. Version 5.0.2 or later of the JIRA Importers Plugin is compatible with Redmine versions 1.3.0+ and 2.5.2.

Before you begin

- Ensure that you are using Redmine versions 1.3.0+ and 2.5.2.
- Ensure that you are using version 5.0.2 or later of the [JIRA Importers Plugin](#). This plugin is bundled with JIRA. For instructions on how to update a plugin, see [Updating apps](#).
- Install the [JIRA Redmine Importer plugin](#), if you haven't installed it already. For instructions on how to install a plugin, see [Installing Marketplace apps](#).
- Enable the REST web service in Redmine in **Administration > Settings > Authentication > Enable REST web service**, if you haven't already enabled it.
- If your JIRA installation has existing data, [back up](#) your existing JIRA data.
-  **Tip:** Redmine supports hierarchical issues. In the Redmine Import Wizard, you are given the option to recreate this issue hierarchy through JIRA issue links. Therefore, before importing Redmine data, you may want to configure a custom issue link to replicate this hierarchy. For example:
 - **Name** — 'Hierarchy'
 - **Outward Link Description** — 'parent of'
 - **Inward Link Description** — 'child of'

Import your Redmine data

The JIRA Redmine Importer plugin provides a wizard that walks you through the process of importing data and integrating it with JIRA. To access the import wizard:

1. Log into JIRA as a user with the **JIRA Administrators** [global permission](#).
2. Choose



> **System**. Select **Import & Export > External System Import** to open the Import external projects page.

3. Select **Redmine** to open the Redmine import wizard.
4. Complete the fields as prompted in the wizard.

If you are importing your Redmine issues into an existing JIRA project, you must choose the JIRA workflow scheme used by that existing JIRA project when you are prompted to select the workflow scheme. Otherwise, your import may not complete successfully.

Please note that it is *mandatory* to map Redmine status field to a specific JIRA status field and

Redmine tracker field to a JIRA issue type field since these JIRA fields are an integral part of [JIRA workflows](#).

Tips for importing Redmine On Demand data into JIRA fields

The import process converts Redmine data as follows:

In Redmine	In JIRA	Import Notes
Project	Project	Redmine data is imported on a per-project basis. You can either specify an existing JIRA project as the target, or the importer will automatically create a project(s) for you at time of import. See Defining a project for more information about JIRA projects.
Target Version	Affects Version	Redmine target version is mapped to JIRA "affects version".
Priority	Priority	You can configure mapping of specific Redmine values to specific JIRA values.
Summary	Subject	Redmine subject is imported as the JIRA issue summary.
Worklog	Worklog	See Configuring time tracking .
Author	Reporter	Redmine issue author is mapped as JIRA Issue Reporter.
Attachments	Attachments	Attachments are extracted from Redmine and saved. Information on the date the file was attached and the user who attached it is retained, as well. To specify the location where the attachments are stored, see Configuring file attachments .
Tracker	Issue Type	You can configure the mapping of specific trackers to specific JIRA issue types.
Priority	Priority	You can configure the mapping of specific Redmine values to specific JIRA values.

Status	Status	<p>You can configure the mapping of specific Redmine values to specific JIRA values, provided you create your workflows in JIRA before running the importer.</p> <ul style="list-style-type: none"> • The JIRA status field is integral to JIRA workflow.d • To create a JIRA workflow, see Working with workflows. • To create a JIRA workflow scheme (which you can then associate with appropriate projects and Issue Types), see Managing your workflows.
Category	Component/s	This mapping is hard-coded and cannot be changed.
User	User	JIRA will automatically import all users from Redmine, unless you are using external user management, in which case JIRA will not import any users. JIRA will then try to work out if the users are required, for example if the user has made a comment, or reported an issue. If the users are not required, JIRA will add them to a group called <code>redmine-import-unused-users</code> , and you can then decide what you'd like to do with these users (you may decide to delete or deactivate them).
<i>Other fields</i>	Custom fields	If your Redmine system contains any custom fields, you can choose to map them to specific JIRA custom field(s). If your custom fields don't yet exist in JIRA, the importer can automatically create them for you.

Importing data from BaseCamp

Unfortunately, right now we don't have a built-in JIRA importer for data from Basecamp. It's also not possible to export the data from Basecamp into a file format which can be directly imported back to JIRA.

There are third party solutions available which might help you. Please check out the [TaskAdapter](#) solution from Atlassian Marketplace. You can also contact your local Atlassian Expert for help or [develop](#) a solution based on Basecamp and [JIRA public APIs](#).

Why do we have this page?

We are tracking the visits to this page. The intensity of visits will help us prioritize the work on the next set of JIRA importers. It will not help you today, but think of yourself as of a democratic voter who will change the future of JIRA. And for that we thank you.

Importing data from JSON

Version 4.3 or later of the [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from a JavaScript Object Notation (JSON) file.

JSON files are easy to read and encapsulate more structure and information than CSV files.

The JSON import feature allows you to import issues from an external (issue tracking) system which:

- JIRA does not provide a dedicated import tool for and
- Can export its data in a JSON format.

You may also wish to prepare your JSON file manually.

⚠ Please note that the import format used by the JIRA Importers plugin is more basic than the import format available when using the JIRA REST API.

Creating a JSON file for Import

If your current issue tracking system is unable to export in the JSON format, you may wish to create the file manually. To prepare the JSON file, you should use the standard **JSON** format, and follow the pattern detailed below.

On this page:

- Creating a JSON file for Import
- Running the JSON File Import Wizard

JSON File Example

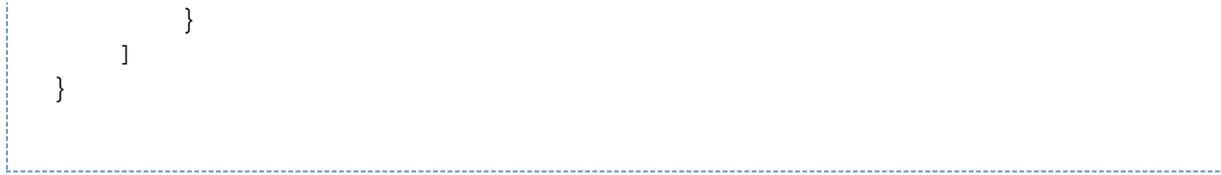
```
{
  "users": [
    {
      "name": "alice",
      "fullname": "Alice Foo"
    },
    {
      "name": "bob",
      "fullname": "Bob Bar"
    }
  ],
  "links": [
    {
      "name": "sub-task-link",
      "sourceId": "2",
      "destinationId": "1"
    },
    {
      "name": "Duplicate",
      "sourceId": "3",
      "destinationId": "2"
    }
  ],
  "projects": [
    {
      "name": "A Sample Project",
      "key": "ASM",
      "type": "software",
      "description": "JSON file description",
      "versions": [
        {
          "name": "1.0",
          "released": true,
          "releaseDate": "2012-08-31T15:59:02.161+0100"
        },
        {
          "name": "2.0"
        }
      ]
    }
  ]
}
```

```

    ],
    "components": [
        "Component",
        "AnotherComponent"
    ],
    "issues": [
        {
            "priority" : "Major",
            "description" : "Some nice description
here\nMaybe italics or bold?",
            "status" : "Closed",
            "reporter" : "alice",
            "labels" : [ "impossible", "to", "test" ],
            "watchers" : [ "bob" ],
            "issueType" : "Bug",
            "resolution" : "Resolved",
            "created" : "2012-08-31T17:59:02.161+0100",
            "updated" : "P-1D",
            "affectedVersions" : [ "1.0" ],
            "summary" : "My chore for today",
            "assignee" : "bob",
            "fixedVersions" : [ "1.0", "2.0" ],
            "components" : [ "Component", "AnotherComponent" ],
            "externalId" : "1",
            "history" : [
                {
                    "author" : "alice",
                    "created":
"2012-08-31T15:59:02.161+0100",
                    "items": [
                        {
                            "fieldType" : "jira",
                            "field" : "status",
                            "from" : "1",
                            "fromString" : "Open",
                            "to" : "5",
                            "toString" : "Resolved"
                        }
                    ]
                }
            ],
            "customFieldValues": [
                {
                    "fieldName": "Story Points",
                    "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:float",
                    "value": "15"
                },
                {
                    "fieldName": "Business Value",
                    "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:float",
                    "value": "34"
                }
            ],
            "attachments" : [
                {

```

```
        "name" : "battarang.jpg",
        "attacher" : "admin",
        "created" :
"2012-08-31T17:59:02.161+0100",
        "uri" :
"http://optimus-prime/~batman/images/battarang.jpg",
        "description" : "This is optimus prime"
    }
]
},
{
    "status" : "Open",
    "reporter" : "bob",
    "issueType": "Sub-task",
    "created" : "P-3D",
    "updated" : "P-1D",
    "summary" : "Sub-task",
    "externalId": "2"
},
{
    "status" : "Closed",
    "reporter" : "alice",
    "issueType": "Sub-task",
    "created" : "P-3D",
    "updated" : "P-1D",
    "resolution" : "Duplicate",
    "summary" : "Duplicate Sub-task",
    "externalId": "3"
}
]
```



Custom Fields

The JSON Importers plugin supports custom fields. Below is a list of custom fields that come bundled with JIRA. If you have installed any additional plugins that have custom fields, these fields will also be supported, however they are not included in this list.

▼ Bundled Custom Fields List

1. com.atlassian.jira.plugin.system.customfieldtypes:textfield
2. com.atlassian.jira.plugin.system.customfieldtypes:textarea
3. com.atlassian.jira.plugin.system.customfieldtypes:datepicker
4. com.atlassian.jira.plugin.system.customfieldtypes:datetime
5. com.atlassian.jira.plugin.system.customfieldtypes:float
6. com.atlassian.jira.plugin.system.customfieldtypes:select
7. com.atlassian.jira.plugin.system.customfieldtypes:radiobuttons
8. com.atlassian.jira.plugin.system.customfieldtypes:project
9. com.atlassian.jira.plugin.system.customfieldtypes:multiversion
10. com.atlassian.jira.plugin.system.customfieldtypes:version
11. com.atlassian.jira.plugin.system.customfieldtypes:userpicker
12. com.atlassian.jira.plugin.system.customfieldtypes:url
13. com.atlassian.jira.plugin.system.customfieldtypes:multiselect
14. com.atlassian.jira.plugin.system.customfieldtypes:multicheckboxes
15. com.atlassian.jira.plugin.system.customfieldtypes:multiuserpicker
16. com.atlassian.jira.plugin.system.customfieldtypes:multigrouppicker
17. com.atlassian.jira.plugin.system.customfieldtypes:grouppicker
18. com.atlassian.jira.plugin.system.customfieldtypes:cascadingselect
19. com.atlassian.jira.plugin.system.customfieldtypes:readonlyfield
20. com.atlassian.jira.plugin.system.customfieldtypes:labels

The custom field example below shows some syntax for adding custom fields, including an example of a cascading custom field. If the custom field is not listed above, the "fieldType" can be obtained from the Custom Fields configuration page, by inspecting the source HTML. The "value" is specific to each custom field, and you can find this by inspecting the Edit Issue page's source HTML.

Custom Field Example

```

"customFieldValues": [
    //Custom Fields which accepts single values:
    {
        "fieldName": "My Awesome Text Field
(single line)",
        "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:textfield",
        "value": "some text"
    },
    {
        "fieldName": "My Awesome Select List
(single choice)",
        "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:select",
        "value": "some select"
    },
    //Custom Fields which accepts multiple values:
    {
        "fieldName": "My Awesome Checkboxes",
        "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:multicheckboxes",
        "value": [ "multiple", "checkboxes" ]
    },
    {
        "fieldName": "My Awesome User Picker
(multiple users)",
        "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:multiuserpicker",
        "value": [ "admin", "fred" ]
    },
    //Custom Fields which accepts Options in
hierarchy. That's only cascading select from standard JIRA pool.
    {
        "fieldName": "My Awesome Select List
(cascading)",
        "fieldType":
"com.atlassian.jira.plugin.system.customfieldtypes:cascadingselect",
        "value":
        {
            "": "Parent Value",
            "1": "Child Value"
        }
    }
]

```

Specific JSON File Examples

Further specific JSON file examples include:

Supported Field	Notes	Example
-----------------	-------	---------

Users	<p>This example covers a full user. In this example, two groups have been specified. If a group does not exist already, the JIRA Importers plugin will create it.</p>	<div style="border: 1px dashed blue; padding: 10px;"> <div style="text-align: center; background-color: #f0f0f0; border: 1px solid #ccc; margin-bottom: 5px;">User Example</div> <pre> "users": [{ "name" : "someuser", "groups" : ["jira-users", "my-custom-group"], "active" : true, "email" : "user1@example.com", "fullname" : "User 1" }] </pre> </div>
Project Key and Issue Key	<p>You can assign a key to both the project and the issue. These keys can be different. This example will create a project with one issue, "SAM-123".</p>	<div style="border: 1px dashed blue; padding: 10px;"> <div style="text-align: center; background-color: #f0f0f0; border: 1px solid #ccc; margin-bottom: 5px;">Project Key and Issue Key Example</div> <pre> { "projects": [{ "name": "Sample data", "key": "SAM", "type": "software", "issues": [{ "key" : "SAM-123", "status" : "Open", "reporter" : "admin", "summary" : "Parent case", "externalId": "123" }] }] } </pre> </div>

Comments	This example shows how you can import multiple comments for an issue.	<div style="border: 1px dashed gray; padding: 10px;"> <div style="text-align: center; background-color: #f0f0f0; border: 1px solid #ccc; margin-bottom: 5px;">Comment Example</div> <pre> { "projects": [{ "name": "Sample data", "key": "SAM", "issues": [{ "status": "Open", "reporter": "admin", "summary": "Parent case", "externalId": "1", "comments": [{ "body": "This is a comment from admin 5 days ago", "author": "admin", "created": "2012-08-31T17:59:02.161+0100" }, { "body": "This is a comment from admin 1 day ago", "author": "admin", "created": "2012-08-31T17:59:02.161+0100" }] }] }] } </pre> </div>
----------	---	---

Worklogs	This example shows the syntax to import worklog detail.	<div style="border: 1px dashed blue; padding: 10px;"> <div style="text-align: center; background-color: #f0f0f0; border: 1px solid #ccc; margin-bottom: 5px;">Worklog Example</div> <pre style="font-family: monospace; border: 1px solid #ccc; padding: 5px;"> "worklogs": [{ "author": "admin", "comment": "Worklog", "startDate": "2012-08-31T17:59:02.161+0100", "timeSpent": "PT1M" }, { "author": "admin", "startDate": "2012-08-31T17:59:02.161+0100", "timeSpent": "PT3H" }] </pre> </div>
Component	Components can be specified in a JSON file in two ways, by providing a name, or by providing an object. This example shows both. The JIRA Importers plugin will always create a new component with "Default Assignee" switched to "Project Default", as you are unable to specify a "Default Assignee".	<div style="border: 1px dashed blue; padding: 10px;"> <div style="text-align: center; background-color: #f0f0f0; border: 1px solid #ccc; margin-bottom: 5px;">Component Example</div> <pre style="font-family: monospace; border: 1px solid #ccc; padding: 5px;"> "components": ["Component", //Component specified only by name { // Component specified by object "name": "SomeName", "lead": "admin", "description": "Some description" }], </pre> </div>

<p>Issues with Time Tracking</p>	<p>Time Tracking detail can be imported with an issue. This example shows you an issue with Time Tracking detail. The "originalEstimate", "timeSpent", and "estimate" values must be in Period format (Format ISO_8601 - Durations). The "startDate" value accepts both the DateTime and Period format.</p> <p>Please ensure Time Tracking is enabled in JIRA before you start your import, otherwise the data will be ignored by the JIRA Importers plugin during the import.</p>	<div style="border: 1px dashed blue; padding: 10px;"> <p style="text-align: center;">Issues with Time Tracking</p> <pre> "issues": [{ "summary" : "My Example Time Tracking issue", "externalId": "1", "originalEstimate": "P1W3D", "timeSpent": "PT4H", "estimate": "P2D", "worklogs": [{ "author": "admin", "comment": "Worklog", "startDate": "P-1D", //can be a Period or DateTime "timeSpent": "PT1M" }, { "author": "admin", "startDate": "2014-01-14T17:00:00.000+0100", "timeSpent": "PT3H" }] }] </pre> </div>
----------------------------------	--	--

Dates can be represented in [SimpleDateFormat](#) "yyyy-MM-dd'T'HH:mm:ss.SSSZ" (example output: "2012-08-31T15:59:02.161+0100") or you can use relative dates like "P-1D" (which means one day ago).

Running the JSON File Import Wizard

Before you begin, please [back up](#) your JIRA data.

1. Log in to JIRA as a user with the **JIRA Administrators** global permission.
2. Choose 
 - > **System**. Select **Import & Export > External System Import** to open the Import external projects page.
3. Select **JSON** to open the **JSON File import** page.
4. Choose your JSON file.
5. Click the **Begin Import** button when you are ready to begin importing your JSON file into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.

Note: If you experience problems with the import (or you are just curious), click the **download a detailed**

log link to view detailed information about the JSON file import process. This information can also be useful if you encounter any errors with your import.

Congratulations! You have successfully imported your JSON projects into JIRA! If you have any questions or encounter any errors, please contact [Atlassian support](#).

Importing data from Mantis

The [JIRA Importers plugin](#), which is bundled with JIRA, allows you to import data from **Mantis** by connecting to a live Mantis database.

i Our main website highlights some top reasons why people [migrate from Mantis to JIRA](#). Version 4.2 or later of the JIRA Importers plugin is compatible with Mantis versions 1.1.8 to 1.2.16. The JIRA Importers plugin requires that your Mantis database is MySQL, PostgreSQL or Microsoft SQL Server. We have also received reports that the JIRA Importers plugin works with Oracle and DB2 databases. However, we have not tested this plugin against these databases.

Note that the JIRA importer plugin that imports Mantis to JIRA does not yet support newer versions of MySQL connector (7.0 and later). That might cause importing Mantis to JIRA with the importer plugin to fail. We recommend using the `mysql-connector-java-5.1.46` driver version instead.

The Mantis import process consists of simply running the Mantis Import Wizard ([below](#)).

- You can choose to map individual fields and field values during the import process, some of which are mandatory.
- At the end of the Mantis Import Wizard, you will be given the option of creating a Mantis configuration file, which contains the settings you configured while running through the Mantis Import Wizard. This is useful if you need to test your Mantis import on a test JIRA server first before performing the import on a production system.

On this page:

- [Running the Mantis Import Wizard](#)
- [Tips for importing Mantis data into JIRA fields](#)

Running the Mantis Import Wizard

Before you begin, please [back up](#) your JIRA data.

1. Log in to JIRA as a user with the **JIRA Administrators** [global permission](#).
2. Choose



> **System**. Select **Import & Export > External System Import** to open the Import external projects page.

3. Select **Mantis** to open the **Mantis Import Wizard: Setup** page.
4. On the **Mantis Import Wizard: Setup** page, complete the following fields/options:

Mantis URL	Specify the URL of your Mantis site. This is the URL you would normally use to access Mantis through a web browser.
Specify credentials	Select this checkbox if you want to import Mantis issues into JIRA, which require user credentials on your Mantis site to access them. Selecting this checkbox reveals/hides the Mantis Login and Mantis Password fields, into which you should specify these user credentials.
Database Type	Select the type of database that your Mantis installation uses: <ul style="list-style-type: none"> • PostgreSQL • Microsoft SQL Server • MySQL
Hostname	Specify the hostname or IP address of the server running your Mantis site's database server.

Port	Specify the TCP/IP port that the Mantis site's database server is listening on.  This field is automatically populated with the default port value based on the Database Type you choose above.
Database	Specify the name of your Mantis database (into which Mantis saves its data). The database name, username and user password can usually be found in the Mantis file <code>config_inc.php</code> . (Typically, the default username is "root" and the default password is empty). See also http://www.mantisbt.org/manual/manual.configuration.database.php
Username	Specify the database user that Mantis uses to connect to its database.
Password	Specify the password of the database user (above) that Mantis uses to connect to its database.
Use an existing configuration file	Leave this checkbox cleared if you do not have a configuration file or if you want to create a new configuration file. Configuration files specify a mapping between fields in Mantis and those in JIRA. Note: <ul style="list-style-type: none"> • If you select this option, you will be asked to specify an Existing Configuration File. • If you do not select this option, then at the end of the Mantis Import Wizard, JIRA will create a configuration file which you can use for subsequent Mantis imports (for re-use at this step of the Mantis Import Wizard).
JDBC connection parameters (in expanded Advanced tab)	The Mantis Import Wizard will construct a JDBC-based database URL from the Mantis database server details you specify above. JIRA uses this URL to connect to and import issues from Mantis. If you need to specify any additional connection parameters to your Mantis database, specify them here. If you chose MySQL (above), the Mantis Import Wizard will add several additional connection parameters by default.

5. Click the **Next** button to proceed to the **Set up project mappings** step of the Mantis Import Wizard.
6. On the **Set up project mappings** page, select which Mantis projects you wish to import into JIRA.
 -  All Mantis projects are selected by default, so clear the checkboxes under **Import** of the Mantis projects you *do not* wish to import into JIRA.
 For Mantis projects you wish to import into JIRA, click in **Select a project** and then do either of the following:
 - Start typing the name (or key) of a project that already exists in JIRA or use the drop-down menu to select an existing JIRA project.
 - Select **Create New** from the drop-down menu and in the resulting **Add A New Project** dialog box, type the following:
 - a. A new project **Name**
 - b. A new project **Key**
 -  This will be used as the prefix for all issue IDs in your JIRA project.
 - c. The **Project Lead**.
7. Click the **Next** button to proceed to the **Set up custom fields** step of the Mantis Import Wizard.
 -  This step will almost always appear because at least one Mantis field is not likely match an existing JIRA field.
8. On the **Set up custom fields** page, for each **External field** in Mantis which the Mantis Import Wizard cannot match to an existing JIRA field, you can choose to either:
 - have the Mantis Import Wizard automatically create new **custom fields in JIRA** based on the names of Mantis's fields. This is the default option - whereby the names of the JIRA custom fields to be automatically created appear in the **JIRA field** drop-down lists.
 - create your own custom fields in JIRA to map data from Mantis's fields. To do this, choose **Other** from the **JIRA field** drop-down list and specify the name of your custom field in the new field appearing immediately below **Other**.
9. Click the **Next** button to proceed to the **Set up field mappings** step of the Mantis Import Wizard.
10. On the **Set up field mappings** page, if there **External fields** in Mantis whose values you wish to

modify *before* they are imported into JIRA, select the **Map field value** checkboxes next to the appropriate fields.

i Please note that it is mandatory to map Mantis's **status** (i.e. **Status**) field to specific JIRA **Status** field values as the JIRA **Status** field is an integral part of [JIRA workflows](#).

- Other **External fields** in Mantis which are likely to appear on the **Set up field mappings** page are:

External field in Mantis	Not choosing the 'Map field value' checkbox
username	The Mantis Import Wizard will automatically map Mantis usernames to JIRA usernames (lowercase).
priority	The Mantis Import Wizard will automatically create missing values in JIRA and will ensure that the issues are migrated with the correct priority (e.g. "Normal" in Mantis to newly-created "Normal" in JIRA).
severity	The Mantis Import Wizard will not map values for this field.
resolution	The importer will create corresponding Resolutions in JIRA instead of using the existing ones.

- Select the appropriate JIRA **Workflow Scheme** in that will be used by the Mantis issues you will import into your JIRA project.
 - i** If you are importing your Mantis issues into an existing JIRA project, ensure that you choose the JIRA workflow scheme used by that existing JIRA project.
11. Click the **Next** button to proceed to the **Set up value mappings** step of the Mantis Import Wizard.
 12. On the **Set up value mappings** page, specify JIRA field values for each Mantis field value (as detected by the Mantis Import Wizard).
 - i** Any fields whose **Map field value** checkboxes were selected in the previous step of the Mantis Import Wizard will be presented on this page, including the mandatory **status** Mantis field.
 13. Click the **Next** button to proceed to the **Set up links** step of the Mantis Import Wizard.
 14. On the **Set up links** page, specify the JIRA link type for each Mantis link type (as detected by the Mantis Import Wizard). To learn more about JIRA link types, please see [Configuring issue linking](#).
 15. Click the **Begin Import** button when you are ready to begin importing your Mantis data into JIRA. The importer will display updates as the import progresses, then a success message when the import is complete.
 - i Note:**
 - If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the Mantis Import Wizard process.
 - If you need to import data from another Mantis product/project or site with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a Mantis configuration file, which you can use at the [first step](#) of the Mantis Import Wizard.

Congratulations, you have successfully imported your Mantis projects into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing Mantis data into JIRA fields

During the import process, the following data is copied from the Mantis database into JIRA:

In Mantis	In JIRA	Import Notes
Project Sub Project	Project	Mantis data is imported on a per-project basis. You can either specify an existing JIRA project as the target, or the importer will automatically create a project(s) for you at time of import. See Defining a project for more information about JIRA projects.

Category	Component	You can choose to have the importer automatically create your Mantis components in JIRA, or choose to have bugs imported into no component in JIRA.
Version	Fix Version	Versions are imported from Mantis (if you choose). After importing, you can manually set appropriate versions to the Released state in JIRA if you wish.
Bug	Issue	Every Mantis bug becomes a JIRA issue of type 'Bug'.
ID	Bug Import ID	Each imported issue will be given a new JIRA ID, and the old Mantis ID will be saved into a JIRA custom field called 'Bug Import ID'. This custom field is searchable, so you can search for JIRA issues by their old Mantis ID. If you don't need this custom field, delete it or 'hide' it (as described in Specifying field behavior).
Summary	Summary	
Description	Description	Within text, Mantis links (e.g. #1234) are converted to JIRA links (e.g. TST-123).
Comments	Comments	Within text, Mantis links (e.g. #1234) are converted to JIRA links (e.g. TST-123).
Attachments	Attachments	Attachments are extracted from the Mantis database and saved to disk. To specify the location on disk, see Configuring file attachments .
Priority	Priority (or a custom field)	You can choose to map one of either the Mantis Priority field or the Mantis Severity field (see below) to the built-in JIRA Priority field, and the other to a custom field. (Alternatively, you can choose to map both the Mantis Priority field and the Mantis Severity field to JIRA custom fields.) When importing into the JIRA Priority field, you can configure mapping of specific Mantis values to specific JIRA values.
Severity	Priority (or a custom field)	You can choose to map one of either the Mantis Priority field (see above) or the Mantis Severity field to the built-in JIRA Priority field, and the other to a custom field. (Alternatively, you can choose to map both the Mantis Priority field and the Mantis Severity field to JIRA custom fields.) When importing into the JIRA Priority field, you can configure mapping of specific Mantis values to specific JIRA values.
Status	Status	You can configure mapping of specific Mantis values to specific JIRA values, provided you create your workflows in JIRA before running the importer. <ul style="list-style-type: none"> • The JIRA 'Status' field is integral to JIRA workflow. • To create a JIRA workflow, please see Working with workflows. • To create a JIRA workflow scheme (which you can then associate with appropriate projects and Issue Types), please see Managing your workflows.
Resolution	Resolution	You can configure mapping of specific Mantis values to specific JIRA values.
Relationships	Links	You can configure mapping of specific Mantis relationship types to JIRA link types. <ul style="list-style-type: none"> • In JIRA, you can configure different types of links (please see Configuring issue linking).
CC List	Watchers	

User	User	<p>You can choose to have the importer automatically create JIRA users for any Mantis users who do not already exist in JIRA.</p> <ul style="list-style-type: none"> • Users who interacted with the Mantis system will be created as active accounts in JIRA. Other users will be imported into a special group called "mantis-import-unused-users" and will be deactivated. • Passwords from Mantis are not imported (as they are hashed in the database). Users from Mantis will need to get their passwords emailed to them the first time they log into JIRA. • Users with no real name stored in Mantis will get the portion of their email address (login name) before the "@" character as their Full Name in JIRA. • 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. • If you have a user-limited license (e.g. personal license), and the number of required users is larger than the limit, then the import will be stopped. A page will be displayed showing a list of users that can't be created.
Other fields	Custom fields	<p>If your Mantis system contains any custom fields, you can choose to map them to specific JIRA custom field(s). If your custom fields don't yet exist in JIRA, the importer can automatically create them for you.</p>

Moving or archiving individual projects

Over time, your organization's requirements may change. You may need to:

- [Archive](#) a completed or obsolete project.
- [Split](#) a large JIRA instance into several JIRA instances, with particular projects in each.
- [Restore](#) a single project from a backup file into a JIRA instance.
- [Restore](#) an entire JIRA instance, from a backup into a new empty JIRA instance.

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:

- [Online archiving](#)
 - ['Hiding' a project](#)
 - [Making a project 'Read-Only'](#)
 - [Accessing an archived online project](#)
- [Offline archiving](#)
 - [Archiving a project offline](#)
 - [Accessing an archived offline project](#)
 - [Restoring a deleted project](#)
- [Archiving issues](#)

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 no-one 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 no one will be able to modify them.

1. Create a new [permission scheme](#). Grant the **'Browse Project'** permission 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](#)).
3. To prevent workflow transitions from happening you will need to update the workflow and add a condition to each transition. The conditions should check that a user has the Edit Issues permission.

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.

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.

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.
 -  Please note that deleting the Project will result in all the attachments also getting deleted from the [JIRA Home Directory](#). Please ensure that the attachments are copied to the test instance before deleting the project.

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. Note that the JIRA version and database type must be consistent with when the archive was created.

Archiving issues

Archiving issues is also possible. The basic method would be to filter for issues that you want to archive then bulk move them into a separate project which can then be archived by using one of the methods above.

Splitting JIRA applications

Occasionally, an organization may need to split its existing JIRA application instance into two separate instances. For example, there might be a requirement to have some particular projects in one instance, and other projects in a second instance.

Note

This process requires two separate server licenses.

1. Back up your [database](#), using your database backup procedures, and verify the backup.
2. Back up your attachments directory and verify the backup.
3. Install the needed JIRA applications (e.g. JIRA Software) on your new server.
 - ⚠ **Please Note:**
 - The JIRA application version number on your new server must be the same as (or higher than) the version number on your existing server.
 - Do not use the same [JIRA home directory](#) for the two JIRA application instances. Specify a new JIRA application home directory for the JIRA application on your new server.
 - Do not connect the two JIRA application instances to the same external database instance.
4. Create an XML backup from your existing JIRA server application, as described in [Backing up data](#).
5. Import the XML backup 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. See [Configuring file attachments](#) for more information.
7. At this point you should have two JIRA application instances with the same users, projects, issues and attachments. Log in 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 application.
9. Generate new Server ID for the newly installed JIRA application, as described in the article [Changing Server ID](#). This step is needed if you plan to create [Application Links](#) between the two instances.

Exporting issues

If you already have a JIRA Cloud site and want to move to JIRA Server, you can create a backup of your JIRA Cloud data that you can then import into a server installation. Note that the Atlassian Cloud service takes backups for your instance every 24 hours for purposes of application recovery (not for rolling back application data).

You can backup and export the following data from your JIRA Cloud site:

- Issues
- Users and user group settings
- Issue attachments, user avatars, and project logos (if selected)

On this page:

- [How to create a backup](#)
- [How to structure the export file](#)
- [How to import backup data into JIRA Cloud](#)
- [How to import backup data into JIRA Server](#)

How to create a backup

You can generate a new backup every 24 hours from the time the previous backup has finished. Note that JIRA only stores one backup file at a time, and any existing existing backup will be overwritten by a new one. To generate a backup:

1. Choose  **> System.**
2. In the Import and Export section, click **Backup manager.**
3. Check the additional files option if you want to include issue attachments, user avatars, and project logos in the export.
4. Select the type of backup:
 - If you will restore the data to a cloud instance, select **Create Backup for Cloud**
 - If you will to restore the data to a server instance, select **Create Backup for Server**

After the backup is complete, click the file link to download the backup.

▼ Troubleshooting: If you can't find the Backup Manager...

In some cloud sites where the Backup Manager menu does not appear, you can use one of the following workarounds to access it:

`https://<domain_name>.atlassian.net/plugins/servlet/ondemandbackupmanager/admin`

`https://<account_name>.jira.com/plugins/servlet/ondemandbackupmanager/admin`

How to structure the export file

Once you have generated and unzipped a backup file, you should have an output similar to the following:

```
JIRA-backup-20161021
  activeobjects.xml
  entities.xml
  data
    attachments
    avatars
  logos
```

Note that database data is stored in both `activeobjects.xml` and `entities.xml`. Issue attachments, user avatars, and project logos are stored in corresponding directories. If you don't want to import attachments, avatars, or logos to your new JIRA Cloud site, you can remove the corresponding directory and then zip the modified directory tree before importing.

How to import backup data into JIRA Cloud

1. Structure your export file as mentioned [above](#).
2. Follow the instructions on [importing issues](#).
3. After the import, log into your new JIRA Cloud site with the same admin account you used to log into your original site. Cross-application links (links to a source file page from a JIRA issue) will point back to the corresponding source location after the import.

How to import backup data into JIRA Server

1. Structure your export file as mentioned [above](#).
2. Follow the instructions on the [Migrating from JIRA Cloud to JIRA Server](#) page.
3. After the import, log into your JIRA Server instance with the `sysadmin` username and corresponding password. Make sure to change your password after you log in. Cross-application links (links to a source file page from a JIRA issue) will point back to the corresponding source location after the

import.

Note that the *sysadmin* user is created automatically during the backup process. This user is created to allow you to log in with the necessary JIRA System Administrator permission (a server-specific permission not available in the cloud) after restoring data in JIRA.

Importing issues from JIRA server applications

You can import issues into JIRA cloud applications from existing JIRA server applications. After the import, there are a few tasks you need to do.

On this page:

- [Importing issues from JIRA applications](#)
- [Before you begin](#)
- [Procedure](#)
- [After the import](#)

Importing issues from JIRA applications

This import procedure overwrites all the existing data and configuration in JIRA cloud applications and cross-application settings.

Before you begin

- If you're migrating to JIRA Software Cloud, and you're using JIRA Server 6.4 (and earlier) and JIRA Agile 6.7 (and earlier), you must first upgrade to JIRA Software Server 7.0 or above.
-  This import procedure overwrites all the existing data and configuration in JIRA cloud applications and cross-application settings. For example, issues and their attachments, look and feel configuration, *and* users and group memberships. Please be aware that if you choose to import data with this procedure after you have used JIRA cloud applications for a while, you will lose all the data input in the interim.
- This process is recommended for those 3rd party systems which require direct database access to perform data import (e.g. Bugzilla and Mantis). In this case, exposing such connection to JIRA cloud applications would be either insecure (for both sides) or impractical (due to performance issues - very chatty remote connection), so this procedure is used instead.
- Restrictions:
 - Character encoding — JIRA cloud applications use UTF-8 encoding. If your JIRA instance uses other character encoding methods, you cannot import data to JIRA cloud applications.
 - *Third-party issue trackers only:* If you do not have a JIRA server application, download the same version or an earlier version from the [JIRA Downloads Archive](#) and obtain an evaluation license for it from my.atlassian.com.

Procedure

1. Get the data

Third-party issue trackers only. Import data from your existing third-party issue trackers to a JIRA server application. Follow the [instructions here](#).

1. Log in as an administrator. Then at the top right of the screen, choose  **> System**.
2. Then choose **Restore system** in the **Import and Export** section. The JIRA Import Wizard appears. Read the prerequisites to view the supported formats for the data to

- be imported.
- In your JIRA Server application, create an XML backup of the issue data with the JIRA XML backup utility and then compress the attachments. For instructions, refer to the [Backing Up Data](#) page.
 - Optional:* If you want to import avatars or logos, back them up as well. To do this, compress the `/data/avatars` and `/data/logos` directories individually. Make sure that the `/avatars` and `/logos` directories are at the top level.

Supported file formats for the backup data:

- Issues: XML, Zip containing XML file (`.zip`), GZipped XML file (`.xml.gz`), BZip2 XML file (`.xml.bz2`)

From JIRA 4.4 onward, it is recommended that you use the Zip that JIRA generates during backup. If you do not use it, your data might not be completely restored.

- Attachments, avatars and logos: `.zip`, `.tar.gz`/`.tgz`, `.tar.bz2`

2. Import the data

- Upload the files to the cloud by [using WebDAV](#). (For backup files larger than 4GB, refer to [Uploading Large files to WebDAV](#))
- Log in as an administrator. Then at the top right of the screen, choose  **> System**.
- Then choose **Restore system** in the **Import and Export** section.
- Click the **Next** button and follow the instructions on the wizard to finish the import process.

The wizard checks URLs in the specified XML backup, and will let you choose whether or not to update URLs to point to the new JIRA cloud application.

After the import

Granting application access to new users

The import process does not honor the default application access settings and does not give access to any applications to new users. You must grant application access to these users for them to be able to log in.

For information on how to assign application access, see [Managing application access](#).

Setting permissions

JIRA application permissions

In your old JIRA applications, if you have made changes to the default **'JIRA Administrators'** global permissions, for example you added a group called *managers* to the **'JIRA Administrators'** global permission, you must configure the JIRA global permission settings in your JIRA cloud applications after the import.

This is because the import process does not import the settings of the **'JIRA System Administrators'** and **'JIRA Administrators'** global permissions. The other global permission settings such as **'Browse Users'** are imported.

- 'JIRA System Administrators'**: As JIRA cloud applications are hosted products, you do not have the **JIRA System Administrators** permission. Hence, there is no configuration required for this permission.
- JIRA Administrators'** global permission: configure this permission by adding groups and users to them as needed.

Permissions for other applications

Application permissions are managed in each application individually. If your site has other cloud applications, e.g. Confluence or Bamboo, refer to [Managing application access](#) for information on how to configure permissions for these applications.

Migrating data with 3rd party add-ons

Whether you're scaling your organization, simplifying maintenance for hardware and licensing, or taking an extra step to validate all changes coming into JIRA, we want to ensure there's a clear path for you to migrate data. That's why we teamed up with two of our Top Vendors, **Adaptavist** and **Botron**, who each offer an add-on to help you transfer projects, configuration, and the desired accompanied data from one JIRA instance to another. Why two vendors? While both ensure the same goal, they each offer a different approach to get there. Read more to choose the one that works best for your team.



as in JIRA

ange to your workflows or just a new custom field, take through staging to production to make sure it's all

ing JIRA configuration

Botron: Promoting JIRA configuration



A instances

o much of JIRA, but too many instances might overwhelming. When moving to a single JIRA, don't leave

lating JIRA instances

Botron: Consolidating JIRA instances



objects

ving out of a suitcase, you can at least make sure it's Move them freely between one JIRA and another.

g JIRA projects

RA projects

Migrating data with Adaptavist

An add-on for JIRA prepared by Adaptavist helps you transfer your configuration, projects, and all associated data from one JIRA to another. Read about possible scenarios and choose the one you need.

Promoting JIRA configuration

[Learn more](#) about moving your changes from development through staging to production to make sure they're all checked up. By doing so, you protect your production environment from any mistakes or unevaluated changes.

Consolidating JIRA instances

[Learn more](#) about consolidating multiple JIRA instances into a single one.

Migrating JIRA projects

[Learn more](#) about migrating your projects, along with all relevant data, from one JIRA instance to another.

Promoting configuration changes from staging to production

Overview

- Overview
- Preparing
- Developing and validating changes in staging
- Exporting configuration for changed projects from staging
- Importing project configuration into production
- Verifying results of promotion
- Troubleshooting

What do we mean by "promoting configuration changes from staging to production"?

Let us explain with an example. Imagine you are the JIRA administrator in charge of a JIRA instance with a few hundred users. The users working in two of the biggest projects want to implement changes to how those projects are managed in JIRA. The JIRA team knows that implementing those changes needs at least three days of work, and also that they should be reviewed and validated by the users before putting them in production. Some of the user requirements are not absolutely clear and you know that until users see the implementation they will not be able to decide if that new implementation is what they actually need. A preliminary analysis of the requirements has shown that they imply changes to project workflows, screens, custom fields, and permission schemes. New groups will also be created so that permissions can be granted only to those users that need them.

Starting to implement these changes on production means JIRA users would have to put up with three days of partially implemented changes. What's more, the changes won't be final until the users validate them, so it is possible they have to be modified and redone a few times. This would mean massive disruption to everyday work for users in the affected projects. So you quickly realize that those changes must be implemented first in a separate development instance (which we will call "staging" in this guide.)

This decision brings up new questions:

- How to migrate the changes implemented in staging to production without manually redoing them?
- How to make sure the tests and validations in staging really represent the final result when the changes are moved into production?

This guide intends to answer these questions, offering a recommended process for implementing, reviewing, and promoting configuration changes between JIRA instances. This process is based on the use of the plugin called [Project Configurator for JIRA](#).

A more elaborate process

In some cases, when the rate of changes applied into production is very high with frequent configuration updates for different projects, the process described in this guide might not guarantee a representative test of the new configuration impact on production. In these cases, it would be practical to have a variation of the process that uses an additional JIRA instance.

In this variation, when changes are ready to be applied to production, a clone of production is created. Let's call this clone "pre-production". Changes are imported into pre-production and the new configuration is validated there, eventually fixing anything if necessary. Then, the new configuration for the changed projects is exported from pre-production and imported into production.

Obviously, the import into pre-production does not require the same caution as into a real production instance, so a previous backup and locking up the instance would not be needed.

What is included in the concept of "configuration"?

The best way to explain it is pointing to the [list of entities that Project Configurator for JIRA will move in a configuration export/import](#).

When you export the configuration for a group of projects, all configuration objects that are used by those projects will be exported. For more information, see [Selection of objects to export](#).

Limitations

Technical limitations are described [here](#). Note that the limitations in the [items that must coincide in both JIRA instances](#) section would not be an issue if the advice in this guide is followed. Starting with the staging instance as a clone of production would guarantee that those aspects of both instances are the same.

Cloud and Server instances

The process explained in this guide is valid only for JIRA Server instances. If you want to use it for promoting changes into a JIRA Cloud instance, you could only do it *indirectly* by following these steps:

1. [Clone a JIRA Cloud instance into a Server instance](#) that will be used as a staging machine (for development and test.) This would replace the [first step](#) explained in the guide.
2. Follow the rest of steps as explained in the guide, up to and including [exporting the configuration changes from staging](#).
3. Lock the JIRA Cloud instance to users.
4. Clone it to a JIRA Server instance. This will act as the "production" instance in the next steps of the process: [importing configuration](#) and [verifying results](#).
5. When verification shows that promotion results are correct at the "production" Server instance, [move it to the Cloud](#), replacing the previous Cloud instance.
6. Open the new Cloud instance to users and resume normal work.

Preparing

Get and install Project Configurator for JIRA

Project Configurator for JIRA is [available at the Atlassian Marketplace](#) as a Paid-via-Atlassian Plugin 2 add-on. It can be installed and licensed in all [the usual ways](#), either from the Universal Plugin Manager inside JIRA, or from [its page](#) at the Marketplace.

The plugin must be available in all instances where it will be used for exporting or importing.

Start a staging instance as a clone of production

You should start with a staging instance that is a copy of your production machine. This will ensure that configuration changes applied in staging cause the same effects they will have when applied to production. In other words, this will make tests and validation on staging representative of what will happen later when the configuration changes are moved into production.

For more information on how to clone a JIRA instance, see [Establishing staging server environments](#).

Synchronize staging with production

If the staging instance was cloned from production some time ago, it is a good idea to "refresh it" with the latest configuration changes from production. Remember – as the staging instance is a closer reflection of production, the tests performed in staging will be more representative of the effects of the new configuration when it is finally promoted to production.

If the JIRA Install directory at production hasn't been changed (for example, installing new add-ons or upgrading existing ones) since it was last cloned to staging, you don't need to repeat the whole cloning process. Just make a backup of production and restore it at staging. See [Backing up data](#) and [Restoring data](#).

Agree with users on promotion windows

It is a good idea to agree previously with users when they will be available to review and validate changes at the staging instance.

As a safety measure, we will take a backup of the production JIRA before promoting configuration changes to it. This implies that the contents of that instance must be frozen after the backup so that, if the need arises to restore the backup, there won't be any last-minute changes that are lost. So, it will be very convenient to find in advance candidate time slots where the following operations can be performed on production:

- Locking the instance

- Creating a backup
- Promoting configuration changes from staging
- Validating these changes
- Unlocking the instance

Very likely, qualified users will have to be a part of this decision.

Let us know what you think

[Feedback](#)

Developing and validating changes in staging

Change the configuration in staging

Now, it's time to change the required items on staging – custom fields, workflows, schemes, etc. in order to satisfy the requirements that have been identified.

Validate those changes internally, analyze impact on other projects

Review and validate the new configuration. Depending on the complexity and scope of proposed changes, it may be important to check if:

- The workflows are complete, with all required states and transitions.
- Issues have fields to hold all required information.
- Every issue has all its required fields filled, either at creation or at some point in its workflow.
- Everybody is granted permissions in line with tasks and responsibilities, but not more.
- All screens are verified, as they will be seen by end users.
- All main use cases are verified – create, transition, and edit typical issues, use filters, dashboards, or reports to obtain the required information.

It is especially important that attention is paid to the impact of these changes on other projects. This impact is produced when you modify a configuration object (e.g. a workflow scheme) that is shared with several projects. Most often, JIRA will show you where a configuration object is used. Following the example, the next image displays the workflow schemes administration page, where you can see which projects use every active workflow scheme:

Name	Projects	Issue Type	Workflow	Actions
classic	• Project Configurator Plugin	Unassigned Types	classic default workflow	Edit Copy

If you detect that a configuration object is being changed and it is used by other projects, you should analyze the impact of the changes on those other projects, eventually performing some of the reviews and tests mentioned above. The most complex situation would be that you detect that the proposed changes are incompatible with other projects when the impact on them is not acceptable. In this case, your best option is to copy the configuration object that is to be modified, so that one of the copies can be changed, while the other (used by other projects) remains untouched.

Note: Changes to custom fields with impacts on other projects can be ignored if the **Smart custom field contexts** option is enabled during the promotion of changes to production.

Validate changes with users

When all changes are implemented in staging they can be shown to the users and reviewed jointly with them. The goal is getting the users to view and approve the new configuration, or gather feedback about desired changes.

Note that in some cases this user acceptance will be mandatory, according to the organization procedures.

Knowing what has changed

Perhaps at some point during the development, you may be asking yourself what has changed in the configuration since you started from the production clone. There are two ways to get a practical answer to this question:

1. [Export the configuration](#) from the staging instance, and [run a simulated import](#) into production. This will print all differences between the configuration for the exported projects in staging and their *current* configuration in production.
2. Before starting developments in staging, [export the existing configuration](#) for the projects and save the resulting XML configuration file. When you want to check what has changed as a result of the new developments, export the new configuration into another XML file and compare it to the one you obtained before the development started. Any diff tool for text files available in your environment can be used for the comparison and the output XML file is designed so that it can be used to track configuration changes. This method has the advantage that you can store XML files of different stages of the development and then use them as a "version control" of the whole stream of configuration changes that are being made in the project. You could compare and view configuration changes in different moments in time and even restore the configuration in the staging instance to one of those past configurations ([importing the configuration](#) of that XML file into staging.)

Let us know what you think

[Feedback](#)

Exporting configuration for changed projects from staging

When all the configuration changes are ready in staging, you can promote them to production. The first step is exporting them from staging.

Starting the export

1. In the staging instance, open a JIRA session as a user with the system administrator permission.
2. In the top-right corner, open the Administration menu, and click **Add-ons**.
3. In the menu on the left, in the PROJECT CONFIGURATOR section, click **Export selected projects**.

ISSUE COLLECTORS

[Issue Collectors](#)

PROJECT CONFIGURATOR

[Import project configuration](#)

[Import complete project](#)

[Export all projects](#)

[Export selected projects](#)

MONITORING

4. Select the projects that you want to export (to select more than one, use the `CTRL` key in Windows, or `COMMAND` key in macOS.)
5. Select **Configuration only** below the list of projects, and click **Next**.

The screenshot shows the 'Project export' wizard in the JIRA Administration console. The 'Add-ons' tab is selected, and the 'Project export' section is active. The wizard is in the 'Select projects' step, with a progress bar showing three steps: 'Select projects', 'Export options', and 'See results'. A list of projects is displayed, with 'Temporary PB' selected. Below the list, the 'Export' options are 'Configuration only' (selected) and 'Configuration and data'. A 'Next' button is located at the bottom of the wizard.

Export options

On the next page, you can fine tune some export options. The meaning of each option is explained on [this page](#). Let's review them and see which are usually the best choice.

- **Filtering custom fields** – In most cases, you will export only the custom fields that are used by the exported projects. Exporting all custom fields would add unnecessary complexity to the process. This option is kept mostly for historical reasons and backward compatibility.
- **User export options** and **Group export options** – The choice here depends to a great extent on the management model in place for users and groups, and their part in the changes that have been performed in staging.
 - Full export – Choose if users and groups are managed in the JIRA internal directory, and have been created or changed in staging.
 - Do not export – Choose if users and groups are managed in an external directory (Active Directory, any other kind of LDAP, etc.), or have *not* been changed in staging.
 - Do not export and Ignore invalid users/groups – These might be useful when you have a large number of users and groups, and some of them might be inconsistent, e.g. there are references to users and groups that do not exist any longer, or have invalid email addresses.

Bear also in mind that if the production instance has several directories for users and groups, any new user/group will be created in the first writable directory (see [here](#) for more details.)

Exporting filters, dashboards, and Agile boards

If the changes developed in staging are centered on a small group of projects, you can export only filters, dashboards, and Agile boards that are related to these projects. In such a case:

- For filters and dashboards, select the **Shared with exported projects** option.
- For Scrum and Kanban boards, select the **Associated to exported projects** option.

The criteria to decide when one of these objects is shared or associated with a project are these:

Filters and dashboards – They are shared with the project or any of its roles.

Scrum and Kanban boards – The board appears under the project title at the project navigation menu. This happens when the main filter for the board includes a clause like "project=XXX" that restricts its issues to that project.

Launching the export

Finally, click **Export project configuration**, and you will navigate to a page with a progress bar that represents the progress of the export task. When the export is completed, your browser will download an XML file with the configuration for the selected projects. Store the downloaded XML file at a known location.

Let us know what you think

[Feedback](#)

Importing project configuration into production

The next step in the promotion process is importing the new configuration into production.

Before you begin

Importing a configuration involves a large number of changes to a JIRA instance, so it is a good practice to adopt some safety measures, just in case something goes wrong or not according to expectations.

The most recommended protection is to have a valid backup of the production database from just before the import. This usually requires that the production instance is closed to users. The reason for that is the fact that user operations performed after the backup would be lost if the backup had to be restored.

Close the production instance to users and back it up, either with the JIRA XML backup tool, or by using native database tools.

Import the project configuration

1. In the production instance, open a JIRA session as a user with the system administrator permission.
2. In the top-right corner, open the Administration menu, and click **Add-ons**.
3. In the menu on the left, in the PROJECT CONFIGURATOR section, click **Import project configuration**. A page will open, where you can specify which configuration file you want to load, and choose some import options.

The screenshot shows the 'Import project configuration' page in JIRA. The top navigation bar includes 'Applications', 'Projects', 'Issues', 'Add-ons', 'User management', and 'System'. The left sidebar has sections for 'ATLASSIAN MARKETPLACE', 'JIRA SUITE UTILITIES', and 'PROJECT CONFIGURATOR'. Under 'PROJECT CONFIGURATOR', 'Import project configuration' is selected. The main content area is titled 'Import project configuration'. It features a 'Project Configuration File' field with a 'Choose File' button and the filename 'config-dump-...eme.xml.txt'. Below this are several checkboxes for import options:

- Apply changes? When ticked, configuration changes will be applied to JIRA
- Create other projects? When ticked, load will create other projects needed by custom field configuration contexts.
- Smart custom field contexts When ticked, load will change only those custom field configuration contexts related to projects being imported
- Try to publish drafts Try to publish automatically workflow drafts and workflow scheme drafts created during the import
- Continue on errors found in dashboards and filters When ticked, load will not stop when an error is found importing a dashboard or filter

At the bottom, there is a 'Do not load:' section with a dropdown menu listing:

- Projects (changes)
- Project specific
- Versions
- Components
- Role members
- Global
- Users
- Groups

A note below the dropdown states 'Selected object types will not be created or modified in the load'. An 'Import project configuration' button is at the bottom right.

This page shows a form with several elements:

- A notice about warnings before importing configurations (only if you have never launched an import before)

- A button that lets you select an XML file to be imported
 - Several other [import options](#), which you can select to enable
 - A list of object types, which you can select to be ignored during the import
4. Click **Choose File**, and select the XML file you exported from staging. Make sure that the **Apply changes** radio button is cleared. This will allow you to run a simulation without loading the changes.
 5. Select some import options.

You can read about each import option [here](#), but let's first see how they are used in most cases:

- **Create other projects** – Usually, this option is enabled if you choose to disable the **Smart custom field context** option.
- **Smart custom field contexts** – Enable this option if the production instance has a large number of custom fields and projects that use them, and you don't want the new configuration to impact other projects. If you'd rather have the custom fields configured exactly the same as in staging, then disable this option.
- **Try to publish drafts** – Most users enable this option so that the add-on automatically publishes new versions of workflows and workflow schemes.
- **Continue on errors found in dashboards and filters** – Enable this option if you're importing a large number of filters and dashboards that are not controlled by the admin/development teams (they were authored by the users.) Otherwise, you can disable it.

6. Click **Import project configuration** at the bottom of the page. This will launch a simulated configuration load. You will then see a page with results.

Administration Back to project:

Applications Projects Issues Add-ons User management System

Project configuration import info Back to import project configu

Simulated load finished without errors

Configuration import info

```
INFO 13:10:09,339 Parsing XML file...
INFO 13:10:09,340 Starting to apply project configuration...
WARN 13:10:09,340 Informative messages only. No changes will be actually applied!
INFO 13:10:09,340 Creating project APWMLTS Another Project With a Weird Issue Type Scheme
INFO 13:10:09,342 Creating project CWLTS Contains a Weird Issue Type Scheme
INFO 13:10:09,346 [Status In Progress] Modifying statusCategory to: indeterminate
INFO 13:10:09,350 [Resolution Done] Making it a non default resolution
INFO 13:10:09,351 Creating issue type Experimental
INFO 13:10:09,351 Creating issue type Fault
INFO 13:10:09,352 [Issue type scheme JIRA Service Desk Issue Type Scheme for Project CITBIZ] Modifying included issue types to: [Incident, Service Request, Change, Problem, Fault, Experim
INFO 13:10:11,780 [Project APWMLTS] [Actual members for project role Administrators] Setting member users to: [admin]
INFO 13:10:11,781 [Project APWMLTS] [Actual members for project role Administrators] Setting member groups to: []
```

This page shows a simulation of all operations that would be performed on the production instance after importing the configuration file. Since it's only a simulation, none of these changes have actually been applied. Review this information, and check if the changes are aligned with your expectations.

If everything is fine, you can launch the actual import. Go back to the **Import project configuration** page, again select the XML file, and enable the **Apply changes** option. Finally, click **Import project configuration** at the bottom of the page. You will obtain a similar results page with the trace of all changes that have been applied to the production instance.

Let us know what you think

[Feedback](#)

Verifying results of promotion

Check results of promoting changes into production

After completing the import, you can perform a quick review of the results. First of all, you can examine the import trace that Project Configurator created during the real import. Check if there are any errors that might have interrupted the import, or errors regarding the import of filters or dashboards that wouldn't stop

the import, but might have an impact on a specific filter or dashboard.

You can also run some quick tests. All the advice about tests in staging given on [this page](#) applies also to the production instance. In fact, you could use a subset or all of these tests now.

Open the production instance to users

Finally, if these tests show that the new configuration is working as expected, the only thing left to do is opening production instance, so that every user can log in and start the work.

Let us know what you think

[Feedback](#)

Troubleshooting

If you encounter any problems while exporting or importing your configuration changes, check the following resources:

- [Export – contents of an error page explained](#)
- [Export – most frequent errors](#)
- [Import – contents of an error page explained](#)
- [Import – most frequent errors](#)
- [Issue tracker – issues with their analysis and solutions](#)
- [Support channels](#)

Let us know what you think

[Feedback](#)

Migrating projects to another JIRA instance

Let's assume you have two instances of JIRA – a source, and a target. In the source instance, you have several projects and users working on these projects. You'd like to transfer the projects to the target instance along with the configuration, issues, and attachments. Ideally, after the migration, your users wouldn't even notice the change. Here are a couple of possible scenarios:

- **Expanding a small project** – the source instance is operated by a group or department within a bigger organization. A project is started there as an experiment. After some time, however, this project has expanded to other groups, and you want to move it to the target, corporate instance of JIRA.
- **Merging JIRA instances** – a company acquires another company. Both have their own JIRA instances and decide to consolidate them into a single instance. Merging one JIRA with another requires migrating all projects.
- **Balancing the use of licenses** – a company runs two instances of JIRA, one with 500 user licenses, and another with 10,000 user licenses. If, in the bigger instance, only 8,000 users are actively working, you can move some projects from the smaller instance to improve the balance.

Project Configurator for JIRA

This guide describes how to migrate your projects with [Project Configurator for JIRA](#). It offers an export function that packs your configuration, issue data, and attachments into a single ZIP archive. You can then transfer it to the target server, and import your projects from the ZIP archive.

Preparing for migrating projects

Versions

We recommend that JIRA versions on the source and target instances are the same. If that's not the case, you should upgrade one of the instances.

However, in some cases, you can work around this limitation.

Migrating between different JIRA versions

- If the JIRA version on the target instance is earlier than on the source instance, you'll need to upgrade the target instance.
- If the JIRA version on the source instance is earlier than on the target instance, you have two options:
 - Upgrade the source instance.

- Migrate by using a staging server, following these steps:
 1. Clone the target instance to a staging server.
 2. Take an XML backup of the source instance.
 3. Restore the backup on the staging server. This will erase all content on the staging server, and upgrade the migrated data to a later version.
 4. Use Project Configurator for JIRA to migrate your projects from the staging server to the target instance using the procedure described in this guide.

Disk space requirements

When migrating, you'll export your projects into a ZIP archive that will contain the following information:

- Configuration of the exported projects
- Issues in those projects (including attachments, comments, worklogs, history, and so on)

Before the ZIP archive is created, its components are assembled in a temporary directory, which requires even more space than the final ZIP archive. We recommend that the size of the temporary directory is around 4 times the size of the final ZIP archive.

▼ How do I calculate the size of the ZIP archive?

You can use formulas to calculate the approximate size of the ZIP archive. To do this, you'll need the following information:

- The size of an XML backup of your database.
- The total number of projects in your instance, and the number of projects you want to migrate.
- The size of attachments for the migrated projects. Attachments for each project are stored in separate directories in `<JIRA-home-directory>/data/attachments`, each directory is named after the project's first key. This path is the default directory, you can check it in JIRA by going to



> System > Attachments.

1. Use the following formula:
 $size\ of\ the\ database * migrated\ projects / all\ projects * 1.2$

For example, if the size of your DB is 500 MB, and you want to migrate 5 projects out of 20:
 $500\ MB * 5/20 * 1.20 = 150\ MB$

2. Next, check the size of attachments for the migrated projects, and use another formula.
 $size\ of\ the\ attachments / 4$

For example, if the size of your attachments is 300 MB:
 $300\ MB / 4 = 75\ MB$

The estimated size for the final ZIP archive, using the above examples, would be 225 MB. The disk space for the temporary directory must be around 1 GB, because it must be 4 times the size of the ZIP archive.

Provide the same disk space both on the source and the target instance, because the ZIP archive will be uncompressed on the target instance before being imported to JIRA.

Licenses

Make sure that you have enough licenses for your users in the target instance. If you don't, you'll need to provide more, or deactivate some users after the migration. For more information, see [Create, edit, or remove a user](#).

User management

User management during the migration depends on your specific environment, however, you should consider the following rules:

- Users are mapped from the source to the target instance by usernames (field `Username` in JIRA). If a single person has two different usernames in both instances, you should make them consistent:
 - Rename one of the users, or
 - If the user is managed in an external directory with LDAP, which has some attribute that is the same for both usernames, you can [configure it to act as 'username'](#).
- Users managed in external directories can be migrated with the following methods:
 - Tools specific to external directories (e.g. replicating users across LDAP instances). In this case,

you should migrate your users before migrating your projects. They might be referenced in some parts of the projects' configuration (e.g. permissions).

- Project Configurator for JIRA. For more information and restrictions, see [Specific information for some object types](#).
- A combination of both.

These considerations also apply to group management.

Matching properties of both instances

Make sure the following properties are the same for both instances:

- timezone
- locale (especially the rules that define how numbers and dates are formatted into strings)
- the maximum size for text fields (the size in the target instance can be larger than the size in the source instance)

For more information about setting the size limit for text fields, [click here](#).

Names of configuration objects

The migration process will treat configuration objects with the same names in both instances as representing the same thing. It's mostly the case, but sometimes two objects with the same name might be completely different. The configuration coming from the source will overwrite the configuration in the target because it's assumed that the source represents the newer configuration you want to implement. To avoid losing some items:

- Review the names of [globally available configuration objects](#) in both instances
- If there are coincidences, check if those name actually represent the same object.
- If needed, rename an object in one of the instances to avoid overwriting it with wrong data.

Project Configurator offers some features that might help you with this issue:

- [Import conflict detection](#) – it produces a summary report of all configuration objects in the target instance that will be mapped to objects coming from the source instance. The report also shows where the duplicated objects are used in the target instance.
- ["Used by" report](#) – it shows where the configuration objects are used. This is useful if you need to rename, change, or delete any of these objects.

Plugins defining custom field types

Any plugin that defines custom field types in the source instance must also be installed in the target instance with the same version.

Plugins defining workflow extensions (conditions, validators, post-functions)

Any plugin that defines workflow extensions in the source instance must also be installed in the target instance, either in the same or a newer version.

- Workflows from the source instance will be migrated to the target instance. If the required plugins are not there, the workflows might not work.
- If you don't install the plugins, the import might fail because the "create transition" action of the corresponding workflow is run for every issue imported into the target instance.

Migrating Agile boards

Migration of Agile boards is supported, but if you're still on JIRA 6, make sure you have JIRA Agile 6.7.7, or later. For earlier versions, you won't be able to migrate the boards.

Migrating sprint and ranking data

Migrating sprint and ranking data is supported only for JIRA Software. The data from JIRA Agile will be ignored.

Language and locale

The source and the target instance must be installed with the same language and locale. JIRA Software creates its fields with different names depending on the language settings. This happens even if one of the instances

uses English US, while the other English UK (e.g. Epic Color and Epic Colour). For more information, see [this issue](#).

Duplicate and obsolete rank fields

In some cases, an upgrade of JIRA Software creates fields of type Rank, which are called Rank (Obsolete). If the projects you want to migrate use these fields, you'll probably encounter errors during the migration. These custom fields are usually locked, which means they must be created or configured by JIRA Software (Project Configurator will not create them in your target instance). As a result, your projects in the source and target instance will be using different sets of fields, which breaks the migration.

To avoid these issues, complete the following steps:

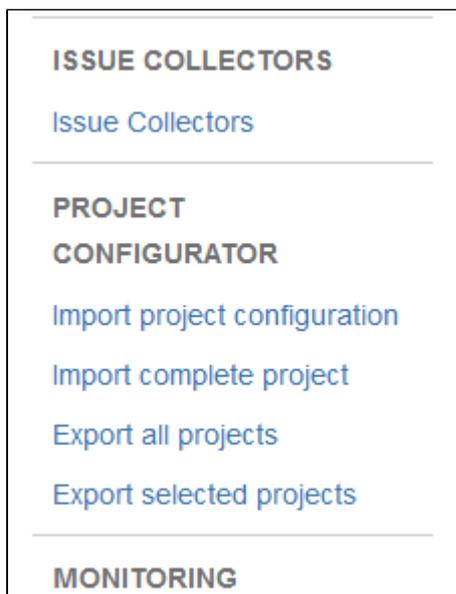
1. Verify that JIRA Software is installed on both the source and the target.
2. If there are duplicate or obsolete Rank fields in the source instance, remove these fields and the related rank values. For more information, see [JIRA KB article](#) and [JSW-13098](#).

Exporting projects from the source instance

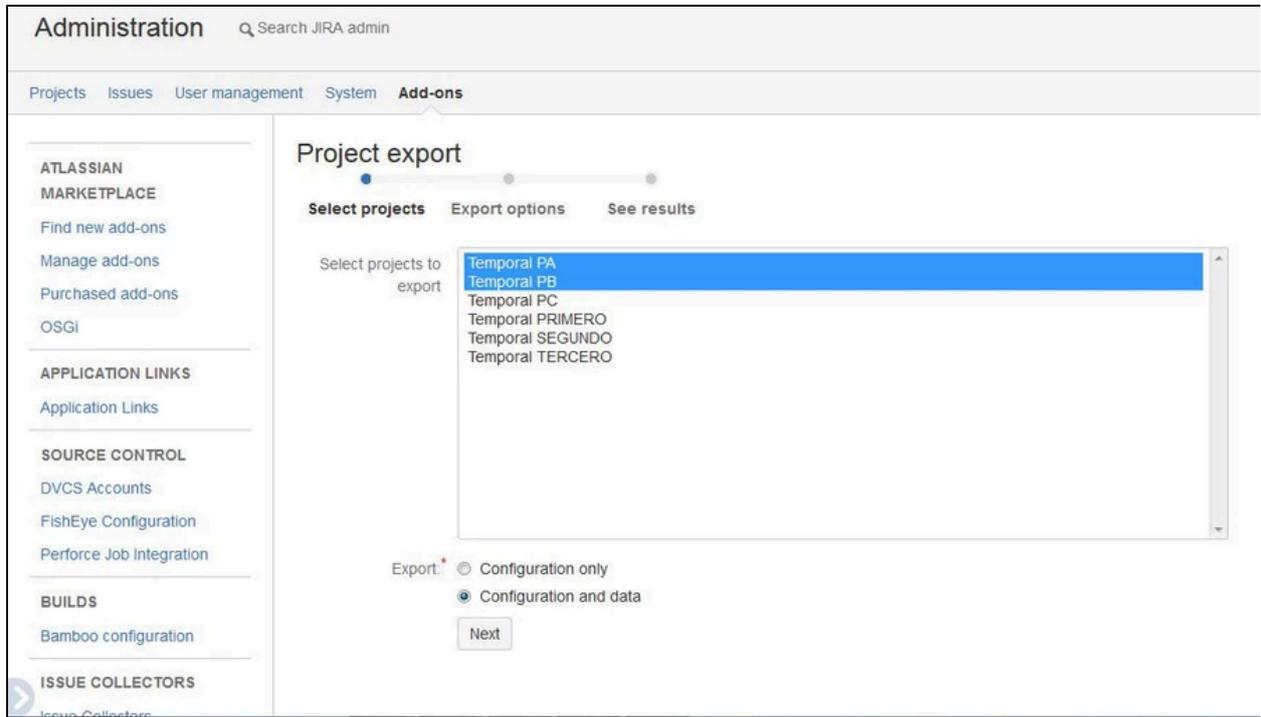
Export your projects from the source instance.

Starting the export

1. Log in to the source JIRA as an administrator.
2. Go to the add-on administration page.
3. In the menu on the left, in the PROJECT CONFIGURATOR section, click **Export selected projects**.



4. Select the projects that you want to export (to select more than one, use the CTRL key in Windows, or COMMAND key in macOS.)
5. Select **Configuration and data** below the list of projects, and click **Next**.



Export options

Next, you will choose export options.

When migrating projects, the most important options are related to exporting custom fields and users.

- **Filtering custom fields** – Select this option to migrate only the custom fields that are used by your exported projects. It'll save you time.
- **User/group export options** – Choose a way to export your user data:
 - **Full export:** This is the default option. If a username is anywhere in the exported configuration, the add-on will try to find and add it to the exported items. If the user is not valid, the export will halt with an error.
 - **Ignore invalid users:** If the username is not valid, the user will not be added to the exported items. The username will still appear as component or project lead, and will be a part of different schemes in the exported file. In these cases, however, the export will not halt. Currently, the add-on can detect an invalid username in two cases:
 - When it can't find a user with that name
 - When the user has an invalid email address (it does not conform to the pattern "X@Y" where X and Y are non-empty strings)
 - **Do not export:** No users will be exported.

For more information about the export options, see [Selecting export options](#).

Exporting filters, dashboards and Agile boards

As for filters and dashboards, it's enough if you select one of the following options:

- **Shared with exported projects** – export filters/dashboards that are used by your exported projects.
- **With all users or with exported projects** – export filters/dashboards that are used by your exported projects, or all users.

For Scrum and Kanban boards, you'll probably export only those boards that are associated with the exported projects (Associated to exported projects).

Export summary

When the export is complete, you will be redirected to a summary of the export that shows the location of the exported ZIP archive. You will need it in the next step.

Export summary

```

Launching export...OK
Exporting configuration...OK
Exporting project data...OK
Exporting attachments...OK
Exported attachments:
    - Project PA: 0 attachments
    - Project PB: 0 attachments
Compressing files...OK

Export file available at D:\nuevo\jira-project-config-plugin\target\jira\home
\export\projectconfigurator\project-dump_BP8Q-WXN6-SKX3-NB5M_2_PROJECTS.zip
  
```

Running a test migration

Before you migrate your projects to the original target instance, you can establish a staging environment, which will be a copy of your JIRA, and test the migration.

Before you begin

Create a staging environment for testing the migration.

Importing

1. In your test instance, go to `<JIRA-home-directory>/import`, and create a directory called `project configurator`.
2. Copy the exported ZIP archive to the `projectconfigurator` directory.
3. Log in to JIRA as an administrator, go to the add-on page, and click **Import complete project**.
4. In the Project File field, enter the name of the exported ZIP archive. Make sure to include the `.zip` extension.
5. Select **Run first a simulated configuration import**.
6. Click **Import complete project**.

The following image shows the relevant options for importing a complete project.

Import options

The default values are sufficient for most environments. However, if you'd like to check other import options, take a look at the following definitions:

- **Run first a simulated configuration import** – this option is recommended, and gives you a chance to review changes to the configuration before applying them.
- **Smart custom field contexts** – enable this option if some custom fields from the source instance have the same type and name as different custom fields already existing in the target. For more information, see [Smart custom field contexts](#).
- **Continue on errors found in dashboards and filters** – often, users create and maintain filters and dashboards without any supervision or help from the administrators. If that's the case for your team, enable this option, so that the migration process skips any inconsistencies or errors in the configuration of filters and dashboards

Summary

After your projects are imported to the test instance, you'll see a summary of the import.

Next, a page will be displayed showing the configuration changes that will be applied to the test instance. The content of this page is similar to the simulated import. The difference is that here you must verify and accept the proposed changes.

✓
Simulated load finished without errors

⚠
Before continuing...

Please review changes to be made to configuration before clicking "Next" to continue

Configuration import info

```

INFO 14:35:04,105 Parsing XML file...
INFO 14:35:04,109 Starting to apply project configuration...
WARN 14:35:04,109 Informative messages only. No changes will be actually applied!
INFO 14:35:04,110 Creating project PA Temporal PA
INFO 14:35:04,110 Creating project PB Temporal PB

```

Drafts for workflows and workflow schemes

In most cases, Project Configurator automatically creates and publishes the required drafts. This happens in the background, and you won't notice it unless you examine the import trace.

In some cases, however, the add-on will not be able to publish the drafts, either because your input is required to map issues to the new status, or the structure of a new workflow is not compatible with the old one. The add-on will stop, and you'll need to complete extra steps to continue. For more information, see [Optional stop for publishing drafts](#).

Validating the test migration

When the test migration is complete, you'll need to review the results and traces before you proceed to migrate your projects to the target instance. You should also check what changes, if any, are needed, and apply them to your source instance.

Import results and import trace

When reviewing the import trace, keep in mind that:

- Data import is carried out in a sequence for each of the migrated projects.
- Warnings – they do not stop the import but might cause some pieces of information to be skipped (e.g.

- values from some custom fields).
- Errors – these will stop the data import for a project. If you're migrating several projects and some of them fail, retry the import. Project Configurator for JIRA will detect the already migrated projects and omit them. If you want to retry the import for a project that already has some data (any issue, version, or component), delete this project at the target instance, and only then retry the import. Otherwise, this project might be detected as the already migrated one.

Project data import results Back to import complete project page

✔ Import of data and attachments finished

Import results

```
PROJECT: Project DL Frontend
Time: 318
Number of created users: 0
Number of created issues: 3
Number of created attachments: 0
PROJECT: Security
Time: 29
Number of created users: 0
Number of created issues: 0
Number of created attachments: 0
```

Import trace

```
Starting data import for project with key: PDLF
INFO 12:27:02,053 Project Import: Parsing the backup file 'D:\other\support-and-debug\amps-standalone\target\container\tomcat7x\cargo-jira-home\temp\com.awnaba.projectconfigurator1475144786701\data\data.zip' to obtain a Backup Overview.
INFO 12:27:02,504 Project Import: Backup Overview was successfully extracted from 'D:\other\support-and-debug\amps-standalone\target\container\tomcat7x\cargo-jira-home\temp\com.awnaba.projectconfigurator1475144786701\data\data.zip'.
INFO 12:27:03,146 Project Import: Mapping the backed up data to data in the current system, and validating the mappings...
INFO 12:27:03,190 Project Import: No validation errors were found and the import can continue.
INFO 12:27:03,195 Starting project import for project 'PDLF'.
INFO 12:27:03,195 Creating missing users. Attempting to create 0 users.
INFO 12:27:03,198 Finished creating missing users. 0 users created.
INFO 12:27:03,205 Creating the issues.
INFO 12:27:03,237 Finished creating the issues.
INFO 12:27:03,315 Creating the issue-related data.
INFO 12:27:03,367 Finished creating the issue-related data.
INFO 12:27:03,371 Creating the change item data.
INFO 12:27:03,378 Finished creating the change item data.
INFO 12:27:03,380 Creating the attachments.
INFO 12:27:03,386 Finished creating the attachments.
```

Configuration changes

To review configuration changes, select **Click here to see configuration import trace**. It will show configuration changes that were applied to your test instance before the import.

Administration Q Search JIRA admin

Projects Issues User management System Add-ons

Configuration import trace Close

Configuration import trace

```
INFO 14:38:45,772 Parsing XML file...
INFO 14:38:45,780 Starting to apply project configuration...
INFO 14:38:45,780 Creating project PA Temporary PA
INFO 14:38:45,836 Creating project PB Temporary PB
WARN 14:38:47,894 [Field Configuration Default Field Configuration] [Field assignee] Cannot make field optional as this is not a requirable field
INFO 14:38:47,968 [Project PA] Associating to workflow scheme: classic
INFO 14:38:48,284 [Project PB] Associating to workflow scheme: classic
INFO 14:38:48,300 Load finished without errors
```

Validate results of test import

After finishing the test import, examine the test instance and validate the migration results. Some suggested checks:

After completing the test import, examine the test instance, and validate the migration results. We recommend to check the following items:

- The projects were migrated, and can be viewed in JIRA.
- The projects have the same number of versions and components as in the source instance.
- Role members are the same. If the number of projects is too big to check that, verify a sample of roles and projects.
- The number of issues and attachments is the same.
- Review a sample of migrated issues, focusing on the following items:
 - General issue data (including custom field values)
 - Sequence of comments
 - Change history
 - Worklogs
 - Attachments (number, name, and content)

User acceptance tests

Finally, select a group of users, so they can work on and test the migrated projects. They should be using the same account they had in the source instance. Check the following items:

- Can you view, and edit issues?
- Can you launch transitions?
- Do you have access to reports, filters, dashboards, or Agile boards?
- Can you access the same information you were using in the source instance?

Record any changes required for a successful migration

During the test migration, you might need to make some changes to the source and target instances, or even files created during the export process, so that the final migration completes successfully. It's also possible that the reviews and tests described on this page detect some other issues, which require changes in both instances. It's important that you record all these changes and fixes so that you can repeat them before the final migration.

Importing projects into the target instance

After you've successfully tested and validated the migration, you can migrate your projects to the target instance. The whole process will be just the same as the test migration.

Before you begin

Back up the target instance, so you can restore it in the case of any problems.

Migrating

1. Lock the source instance, so that your users can't make any changes.
2. [Export your projects](#) from the source instance.
3. [Import your projects](#) to the target instance.
4. [Validate the migration results](#).
 - If you're happy with the outcome, open the target instance to your users.
 - If something is not right, remove the migrated projects from the target instance, and retry the migration.

Troubleshooting the migration

If you encounter any problems with the migration, check the available resources and the list of known issues.

Resources

If you encounter any problems while exporting or importing your configuration changes, check the following resources:

- [Export – contents of an error page explained](#)
- [Export – most frequent errors](#)
- [Import – contents of an error page explained](#)

- [Import](#) – most frequent errors
- [Issue tracker](#) – issues with their analysis and solutions
- [Support channels](#)

Known issues

Custom field context applies only to some issue types

An import fails with the following message:

```
The custom field 'XXXX' in the backup project is used by issue types
'AAAA, BBBB' but the field with the same name in the current JIRA
instance is not available to those issue types in this project.
```

This is a [known issue](#). To work around it, make the custom fields available to all issue types:

1. In the target instance, [configure the custom fields](#) to be available to all issue types.
2. Rerun the import, but choose to ignore custom fields in the configuration import. Otherwise, the custom fields will be imported from the source, overwriting your changes in the target instance.
3. Optional: After a successful import, you can restore these custom fields to their original configuration (restricted to some issue types).

Validation errors with no extra information

The following warnings appear when a value of a text field exceeds the maximum text field size in the target instance. For more information about changing this limit, see [Preparing for migrating projects](#).

```
WARN 11:00:56,968 The attachment 'MMMMMMMMM.zip' does not exist at
'/tmp/com.awnaba.projectconfigurator1493567880435/data/attachments/XXXXXX
XXXXXX'. It will not be imported.

WARN 11:00:56,968 The attachment 'screenshot-1.png' does not exist at
'/tmp/com.awnaba.projectconfigurator1493567880435/data/attachments/YYYYYY
YYYYYYY'. It will not be imported.

WARN 11:00:56,968 The attachment 'screenshot-1.png' does not exist at
'/tmp/com.awnaba.projectconfigurator1493567880435/data/attachments/ZZZZZ
ZZZZZZZ'. It will not be imported.

INFO 11:00:56,969 Project Import: Validation errors were found. The
import cannot continue.
```

Problem with migrating values for the Time in Status custom field

The import trace shows the following warning:

```
WARNINGS: Unable to import custom field 'Time in Status'. The custom
field type does not support project imports.
```

Since it's only a warning, it will not stop the import, but the values for this custom field will not be imported. There's no need to worry about that because these values will be automatically recalculated.

Merging JIRA instances

Overview

Merging JIRA instances requires migrating all projects, issues, dashboards, etc. from one instance to another

without deleting or changing the existing content in the target instance. After the merge, users are expected to resume their work in the target instance, working with the same set of data.

Here are some reasons why you might want to merge your JIRA instances:

- A company purchases another company, both of them are using JIRA and want to merge their two instances into a single one.
- A company has several instances of JIRA in different departments and wants to consolidate all of them into a single, corporate JIRA instance.

Steps

With some differences explained on this page, **merging JIRA instances** is the same as moving all projects from the source instance to the target instance. You should use the steps described in [Migrating projects to another JIRA instance](#), and then proceed with the details described here.

Are all configuration objects migrated?

With some exceptions, where [export options](#) allow the user to control what objects will be included in the export, all configuration objects that are *directly or indirectly required by a project* are exported. Objects that are not used by any project, like inactive workflows, screens, or schemes are not transferred.

Exporting all projects

To export all projects from a JIRA instance:

1. Log in to the source JIRA as an administrator.
2. Go to the add-on administration page.
3. In the menu on the left, in the PROJECT CONFIGURATOR section, click **Export all projects**.



Best export options

These options will be most useful when exporting your projects:

Filtering custom fields – You can either export all custom fields or only those that are used by your projects. In the latter case, custom fields that are not used by any project will be omitted.

User export options and **Group export options** – see the discussion on user management [here](#).

Exporting filters, dashboards and Agile boards

When merging instances, you'll need to transfer all filters, dashboards, and boards.

- For filters, select **All filters (shared or private)**.
- For dashboards, select **All dashboards (shared or private)**.
- For Scrum and Kanban boards, select **All Scrum and Kanban board**.

What to do if the ZIP file is too big

If exporting or importing a ZIP file with all projects is taking too long, or if the ZIP file is too big, try splitting the process into smaller chunks. For example, if the source instance has 50 projects, migrate them in 5 batches,

each containing 10 projects. You can expect the import of the first batch to be more complex than the rest because it also needs to apply all the configuration. You might also want to postpone the import of filters, dashboards, and Agile boards until the last batch when all projects are already available in the target instance.

Shut down the source JIRA instance

To prevent users from logging in to the source instance and creating new data there after the migration, you should shut down this instance, or at least leave it in the read-only mode. Keeping it in the read-only mode might be useful for JIRA admins who could use it in case any data inconsistency for the merged projects is detected after the merge.

Migrating data with Botron

An add-on for JIRA prepared by Botron Software helps you transfer your configuration, projects, and all associated data from one JIRA to another. Read about possible scenarios and choose the one you need.

Promoting JIRA configuration

[Learn more](#) about moving your changes from development through staging to production to make sure they're all checked up. By doing so, you protect your production environment from any mistakes or unevaluated changes.

Consolidating JIRA instances

[Learn more](#) about consolidating multiple JIRA instances into a single one.

Migrating JIRA projects

[Learn more](#) about migrating your projects, along with all relevant data, from one JIRA instance to another.

Promoting JIRA configuration from development to production

Overview

This document describes best practices for promoting JIRA configuration from the development to production environment. You can use Botron's Configuration Manager add-on to effectively test your desired changes prior to rolling them out in the production environment.

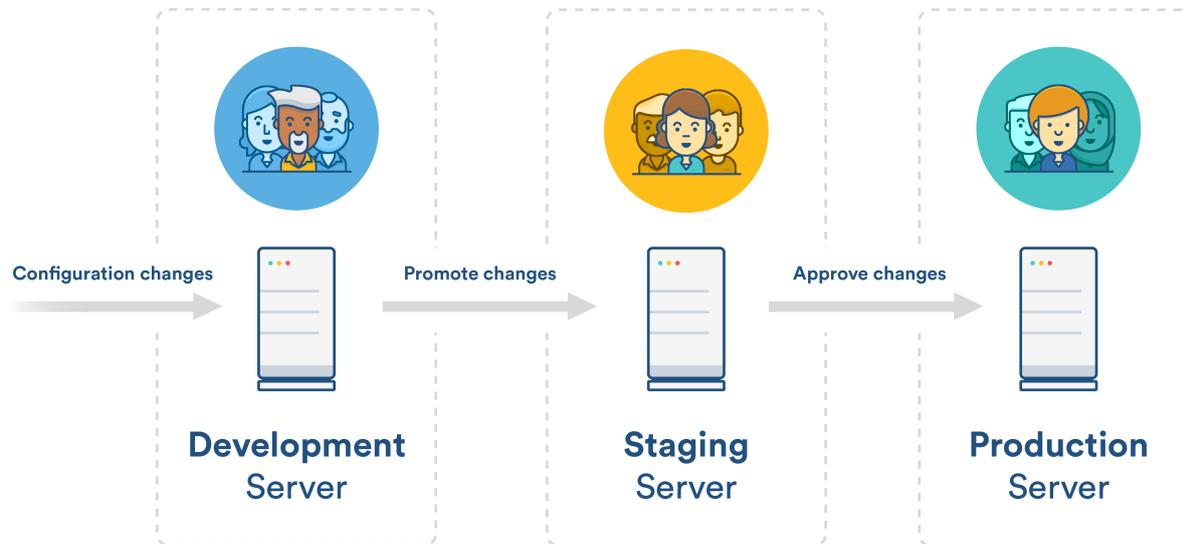
- [Architecture Strategy Recommendations](#)
- [Planning](#)
- [Stages of three-tier architecture strategy](#)
- [Moving add-on data](#)
- [Limitations & workarounds](#)
- [Common issues](#)
- [Frequently Asked Questions](#)
- [Need help?](#)

Definitions

For this document, we'll assume the following definitions:

- **Development** – A free-for-all one or many environments where users can play with cutting-edge or risky changes.
- **Staging** – A pre-production environment, where the systems administration team can establish exact procedures prior to rollout. The staging should be a clone or close replica of the production environment.
- **Production** – Your live instance, expecting minimal downtime and well-tested changes.
- **Configuration snapshots** are created using the [Configuration Manager for JIRA](#) add-on and represent the state of your JIRA [configuration objects](#) and their relations to each other at a given point in time. There are two types of configuration snapshots:
 - **Project snapshot** – Contains the configuration of a number of selected projects (with their schemes, workflows, fields, etc.)
 - **System snapshot** – Contains the entire configuration of a single JIRA instance (projects, workflows, schemes, screens, etc.)

Architecture Strategy Recommendations



If JIRA is a critical system, we recommend a 3-tier architecture strategy consisting of **development**, **staging**, and **production** environments.

If JIRA is not a critical system, you can use a 2-tier strategy with **development** and **production**.

Planning

Prepare environments (hardware)

Staging should be as close as possible to the hardware used for the production server. Although the development can be any type of hardware, the general guideline is that it is as close as possible to production too.

Keep the staging in sync

The staging environment is used as final rehearsal prior to introducing changes in production. To ensure that the rehearsal is accurate and to eliminate potential surprises during the actual deployment, the staging should be an identical replica of the production environment.

Depending on the type of hardware used for production, there are several options for keeping the staging in sync:

- **Physical hardware** – Keeping staging in sync can be accomplished by either cloning the storage used for the database and filesystem, or simply following the JIRA [Backup/Restore](#) procedure.
- **Virtual machine** – Creating a snapshot of the production, and creating a new virtual machine from this snapshot.
- **Configuration Manager** – A unique feature allows the staging server configuration to be [restored](#) with the production.

Configuration Manager for JIRA

Botron's [Configuration Manager for JIRA](#) add-on needs to be [installed](#) on each JIRA server.

Tracking

It is recommended that every change to JIRA setup and configuration is tracked via **change request ticket**. The change request ticket should follow the chosen promotional path and include the initial user request, as well as additional information as it progresses through the lifecycle – configuration snapshot, change and

impact analysis, duration, etc.

Disruptive changes

A recommended IT practice is that all non-emergent fixes should be introduced in a defined cadence, e.g. each Friday afternoon from 1 to 3 PM. The changes introduced can be segmented into two groups – **disruptive** and **non-disruptive**. The former can be introduced only in defined maintenance windows when the user access is limited.

Communication strategy

A communication strategy is designed to inform JIRA users about the changes that will be introduced – this information includes the exact date of the changes roll-out, as well as a comprehensive summary. A link to the change request ticket might also be included.

The communication strategy can be conducted via email, notification banners, or other standard means of internal communications. A recommended practice is to inform the users at least 5 times:

- 1 week prior to the changes being introduced
- At the beginning of the business day
- 1 hour before
- Notification right before
- After the changes are introduced, a message indicating success/failure.

System health

The system health of all 3 servers should be assessed, and any [Integrity Check](#) errors should be resolved.

Stages of three-tier architecture strategy

In this section, we'll walk you through the process of rolling out changes for JIRA production environment with a 3-tier architecture strategy. The illustrated processes require the installation of the [Configuration Manager for JIRA](#) add-on, which enables you to [create and deploy configuration snapshots](#) between the different environments. The add-on should be installed on each environment. For more information regarding licensing – visit the [FAQ](#) section.

First stage (development)

Overview of the first stage:

- **Environment:** Development environment
- **Goals:** Develop new project configuration; allow user acceptance testing; create **configuration snapshot** that can be **promoted** to the **staging** environment.
- **Users:** Business users or administrators. Open to anyone with expertise in JIRA configuration.
- **Tools required:** [Configuration Manager for JIRA](#)

The first stage of this process is conducted in the **development environment**. It typically involves the following **users**: business users or administrators, anyone with an expertise in JIRA configuration.

The **steps** of this stage are as follows:

1. **Develop new project configuration** – a project configuration can contain several configuration elements, e.g. workflows, custom fields, screens, etc.
2. Once the new configuration is in place, a **user acceptance testing** is performed to ensure proper configuration.
3. [Create configuration snapshot](#) that can be **promoted** to the **staging** environment.

Create Configuration Snapshot

Progress: Select → Filters → Boards → Dashboards → **Preview** → Create

Preview Snapshot

Details

Name:	Type:	Description:
Project	Project (HSP)	N/A.

▼ Filters (3 of 5)

Name	Owner	
My Approvals	Administrator	Exclude
My Milestones	Administrator	Exclude
My Tasks	Administrator	Exclude

◀ < 1 > ▶

▼ Boards (2 of 2)

Name	Owner	
TSP board	admin	Exclude
SOM board	admin	Exclude

◀ < 1 > ▶

▼ Dashboards (2 of 2)

Name	Owner	
Dashboard 1	Administrator	Exclude
System Dashboard		Exclude

◀ < 1 > ▶

Back **Create** Cancel

- Download the configuration snapshot** (if your JIRA configurations are being tracked via tickets, the snapshot should be attached to the appropriate tickets.)

Configuration Manager for JIRA will include **only** configuration objects referenced directly or indirectly by the project. If you want to add custom fields to your configuration snapshot, certain conditions must be met. For more information on adding custom fields to your configuration snapshot, [click here](#).

Second stage (staging)

A high-level overview of the second stage:

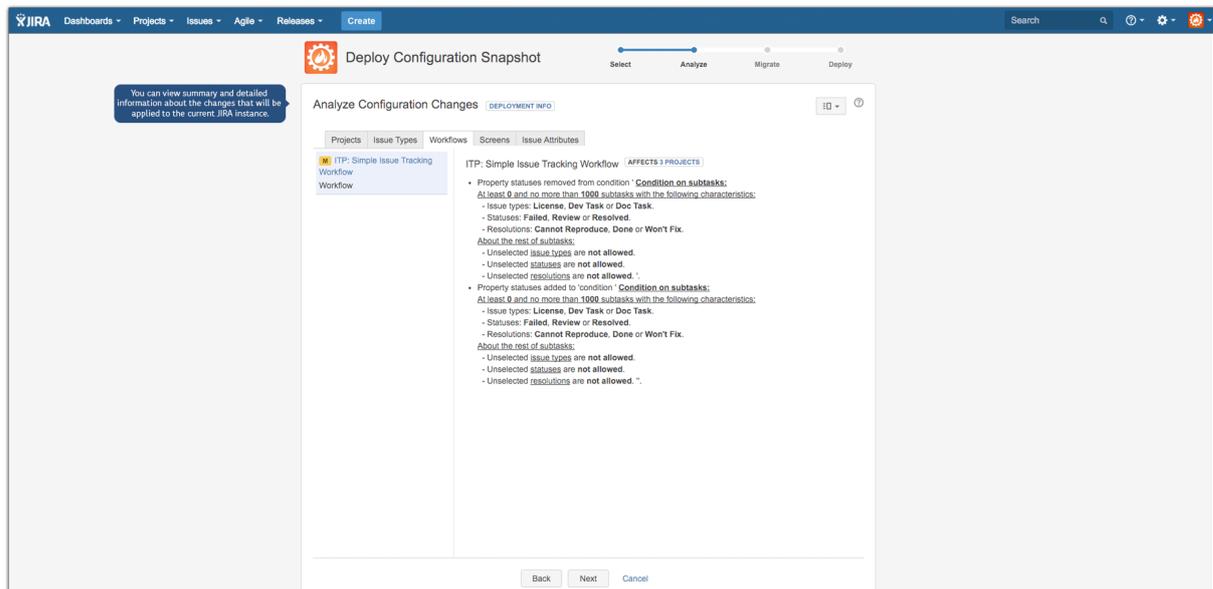
- **Environment:** Staging environment
- **Goals:** Validate proposed configuration changes; assess changes and impact; prepare communication plan; estimate the required downtime, and clarify production rollout procedure.
- **Users:** JIRA System administrators
- **Tools required:** [Configuration Manager for JIRA](#)

The second stage of this process is conducted in the **staging environment** and involves JIRA system administrators who perform a comprehensive validation to ensure that the introduced changes won't impact other projects. There is a difference between **change** and **impact**. For example, a **change** in the name of a

workflow status (from Done to Accepted) might interfere with the company policy or break JQL filters used for reporting. A typical example of an **impact** is when a change to the notification scheme is introduced – that change will likely impact other projects in production.

The **steps** of this stage are as follows:

1. Deploy configuration snapshot and validate proposed configuration changes.



2. It is crucially important to validate the proposed new configuration for **changes and impact**. If the result is negative – go back to Development.
3. Communicate any changes to your users prior to officially introducing them.
4. Estimate the required downtime and clarify production roll-out procedure by measuring the required deployment time and updating your ticket.

Third stage (production)

A high-level overview of the third stage:

- **Environment:** Production environment
- **Goal:** Roll-out configuration changes to production
- **Users:** JIRA System administrators
- **Tools required:** Configuration Manager for JIRA

The third and final stage of this process includes rolling out your configuration changes to production. After ensuring that the results of the change and impact analysis of the previous stage don't show any conflicts or undesired impact, move forward with the deployment to the production environment.

The **steps** of this stage are as follows:

1. Execute the communication plan required.
2. Create a configuration snapshot for backup.
3. Limit user access – changes to production servers are done during **maintenance windows** when the user access is limited (this is not a strong requirement but rather a recommended best practice.)
4. Deploy configuration snapshot.
5. Review audit logs for any warnings.

Administration Search JIRA admin

Projects Add-ons User Management Issues System **Configuration Manager** Audit Log

Snapshots
Deploy

Integrity Check
Audit
Get Started

Configuration Integrity Check Run Integrity Check

Failure
180 configuration elements were checked and 40 errors were found. See details in the table below.

Unknown errors were detected
Send **support zip** to support@atlassian.com, so we can assist you in resolving this issue.

Type	Description
>	Unknown Error while checking issue Type Scheme Default Issue Type Scheme
>	Unknown Error while checking issue Type Scheme Plane Issue Type Scheme
>	Missing object Role Developers refers to the missing user yoda
>	Unknown Error while checking screen Scheme Broken Screen Scheme
>	Unknown Error while checking screen Scheme More Broken Scheme
>	Unknown Error while checking issue Type Screen Scheme Simple issue type screen scheme
>	Missing object Workflow Sales Workflow refers to the missing group Force
∨	Missing object Workflow Sales Workflow refers to the missing custom field customfield_10201

Location:

```

graph LR
    A[Sales Workflow (WORKFLOW)] --> B[Open (1) (STEP)]
    B --> C[Close (31) (TRANSITION)]
    C --> D[User Is In Group Custom Field (CONDITION)]
    D --> E[customfield_1020 (CUSTOM FIELD)]
  
```

Solution:
Modify workflow **Sales Workflow** to point to an existing custom field or remove the reference to the missing custom field.

>	Missing object Workflow Sales Workflow refers to the missing role 10110
>	Missing object Workflow Sales Workflow refers to the missing custom field customfield_10100
>	Missing object Workflow Sales Workflow refers to the missing role 10110
>	Missing object Workflow Sales Workflow refers to the missing role 10110
>	Missing object Project Sample1 refers to the missing user yoda
>	Missing object Project Sample1 refers to the missing group Force
>	Missing object Project Sample1 refers to the missing user yoda
>	Unknown Error while checking project Sample1 : java.lang.NullPointerException
>	Missing object Project Sample2 refers to the missing user yoda

Bug tracking and project tracking for software development powered by Atlassian JIRA (v6.2.3#6260-sha1:63ef1d6) · About JIRA · Report a problem

Atlassian

6. Declare **SUCCESS/FAILURE** – based on the deployment result. If it is successful and the access to the users was limited, communicate and open the system for all users.
7. In case of **FAILURE** – restore the snapshot on the production server, and start again. If the staging server is identical to the production, failures at this point aren't possible.

How to automate the promotion of JIRA project configuration from **test** through **staging** to **productio**

n JIRA environment with Configuration Manager for JIRA

Moving add-on data

Migrating the add-on data is one of the most common data portability issues we've encountered. In this section, we will explore various options for effectively moving the add-on data.

The data created and stored by each add-on can be categorized based on its usage and storage mechanism.

- **User-created data.** For example – the hierarchical structures created by the Structure add-on include data created and consumed by users. It is not related anyhow to the project, system, or even add-on configuration. This type of data is not handled by Configuration Manager for JIRA and the add-on should provide the export/import functionality.
- **Configuration data.** The add-on configuration could contribute to workflow post functions/conditions/validators, custom fields, dashboard gadgets, and other JIRA objects, or could be private for the add-on and stored in the database (e.g. using [Active Objects](#).) Configuration Manager supports several [add-ons](#) out of the box, other add-ons can be easily made compatible if the add-on vendor implements SPI provided by Configuration Manager.

Limitations & workarounds

There are certain **system limitations** that need to be considered during a configuration roll-out, including:

- **Add-ons data** – Certain add-on custom field configuration, add-on workflow property configuration, and any other add-on data outside of JIRA configuration objects might not be included.
- **JIRA Service Desk** – JIRA Service Desk-specific configuration is not supported. It will be added in future releases of Configuration Manager.
- **Differences in configuration objects in 6.x to 7.x** – This should be considered when the snapshot is created on a different major version of JIRA than the target. 7.x has removed some permission types (e.g. the USE global permission is replaced by Application roles) and has introduced various other configuration objects, specifically permissions for permission schemes (e.g. manage sprints), security level type (application role), visibility permission (logged in users.)

Suggested **workarounds** regarding limitations and other known issues include:

- **Scripts** – API level scripts can be developed to migrate any data that is not handled by Configuration Manager for JIRA. Great choice for scripts development is using the [ScriptRunner](#) add-on.
- **Manual** – If the amount of configuration is not extensive it can be manually added to the target JIRA.
- **Professional Services** – Botron's team of solution architects can develop a custom solution that migrates any type of data.

Common issues

We have identified several common issues during deployments, which could negatively impact the data portability process. Some of the most common issues include:

- **Integrity Check** errors which prevent the snapshot creation/deployment. To preserve the integrity of the target/production server, Configuration Manager [Integrity check](#) is executed every time a snapshot is created or deployed. All critical errors reported should be resolved in order to continue. Detailed information can be obtained [here](#).
- **Differences in JIRA versions** – differences in the source-target JIRA versions, specifically major versions.
- **Application permissions/licenses** – for 7.x, e.g. deploying a software project when the user performing the deployment doesn't have permissions to create software projects; or the JIRA Software application isn't installed/licensed.
- **Inconsistencies in user directories** – if test/stage/production environments don't have the same user directory, user creation may be required if a new project is deployed and the project's lead doesn't exist in the target's user directories.
- **Proxy/firewalls** – this may be problematic for large instances when creating a snapshot or before the [Analysis](#) phase when deploying, otherwise there shouldn't be any issues with the deployment itself. This is worked around by increasing any timeout limits on the proxy servers and disabling the firewall

blocking rules.

Frequently Asked Questions

1. Is my entire configuration going to be rolled out to production automatically using Configuration Manager for JIRA?
Yes. The entire configuration, captured in the configuration snapshot, will be rolled out. The supported configuration objects types which are included in the snapshots are listed [here](#).
2. Can I roll out project configuration changes from a newer server instance to an older server instance?
It is strongly recommended that your production and staging server are on the same version. Configuration Manager allows deployment between different versions and warns the user about that. Snapshots created on newer versions may not work on old ones, as they might include new configuration objects which do not exist in the older versions – for example in JIRA 7.x there are new permissions which do not exist in 6.4.x.
3. How long does it take to deploy the configuration snapshot?
Project configuration snapshots take in most cases from a few seconds to 1 minute. Large system snapshots which include hundreds of projects, thousands of filters, and hundreds of Agile boards could take more than 1 hour to deploy. An accurate estimate of the deployment time should be obtained during the staging.
4. What if an error occurs during deployment? Does that break the target/production instance?
All changes deployed by Configuration Manager are executed in a single transaction. If an error occurs, the whole [transaction](#) is rolled back so the only time the server configuration is modified is when all the changes are deployed successfully.
5. Can I automate the snapshot creation and deployment?
Yes. Configuration Manager provides a public [REST API](#) for this purpose.
6. I can't create configuration snapshot due to Integrity check errors. Why is there such a limitation? How can I resolve this problem?
The [Integrity check errors](#) could indicate a serious problem. Other not critical errors are marked as warnings and they don't block the creation or deployment of snapshots. By not allowing a deployment of configuration with critical errors, Configuration Manager protects the production system from being modified with something that is not working.
7. Can Configuration Manager handle custom-built add-ons?
Yes. Configuration Manager can be extended to handle any custom add-ons. Contact Botron's [service s team](#) for more information.

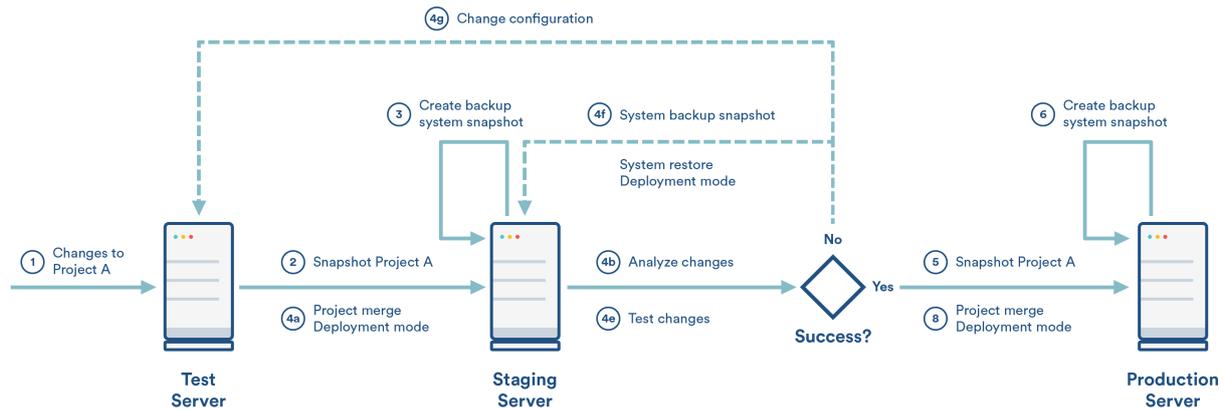
Need help?

If you cannot resolve the problem yourself, you can contact us for assistance via the following communication channels:

- Send email to support@botronsoft.com
- For training or services specific information, contact services@botronsoft.com

Three stages in detail

The procedure of staging any changes before they are made on the **production** server is a recommended IT best practice. The following article describes how to set up such environments: [Establishing staging server environments for JIRA applications](#).



Steps

This use case has two major phases: promotion from test to staging and promotion from staging to production.

Test to staging

1. **Test:** Create all required changes to project A.
2. **Test:** Create a snapshot for project A.
3. **Staging:** (optional) Create a system snapshot for backup.
4. **Staging:** Deploy the snapshot.
 - a. Use the [Project Merge](#) mode.
 - b. Review the change and impact analysis in the [Analyze](#) step of the deployment. Now that **staging** is a close or identical replica of the **production**, during this analysis, you will see how the deployed configuration will impact the **production** server.
 - c. Proceed with the deploy (the change and impact analysis shows that everything is OK.)
 - d. Review the [Audit log](#) for any warnings.
 - e. Test the deployed changes – the actual testing depends on the changes and may include creating issues, exercising issues workflow, etc. Declare **SUCCESS/FAILURE** – if the deployment and testing are successful proceed with promotion to **production**.
 - f. In case of failure, locate the error and make changes in the **test** environment to address it.

Note – **do not** make changes directly to the **staging** server – it needs to remain the same as **production**.

- g. In case of failure, restore at the **staging** server the system snapshot created at step 3 above.

Staging to production

5. **Staging:** Create a snapshot of project A.
6. **Production:**(optional) Create a system snapshot for backup.
7. **Production:** (optional) [Limit the user access](#) to JIRA. It is recommended that all changes to production servers are done during **maintenance windows** when the user access is limited.
8. **Production:** Deploy the snapshot.
 - a. Use the [Project Merge](#) mode.
 - b. Review the change and impact analysis in the [Analyze](#) step of the deployment.
 - c. Now that the changes were staged at the **staging** instance, the change and impact analysis shown at this step should be identical with the one on staging. Those changes were tested and you can proceed with confidence.
 - d. If the change and impact analysis is different than the one on **staging**, then both environments are different and this should be corrected.
 - e. Proceed with the deployment (the change and impact analysis shows that everything is OK.)
 - f. Review the [Audit log](#) for any warnings.

- g. Declare **SUCCESS/FAILURE** – based on the deployment result. If it is successful and the user access was limited, open the system for all users.
- h. Only in case of failure – restore at the **production** server the system snapshot created at step 6 above. Restart the process from the beginning. If the **staging** server is identical to the **production**, failures at this point aren't possible.

Note – **do not** make changes directly to the **production** server

Automation

Automation of the above steps can be accomplished by using the public [REST API](#).

Let us know what you think

[Feedback](#)

Migrating JIRA projects

Overview

This document describes best practices for moving projects between JIRA server instances. The described migration processes are performed with the help of the [Configuration Manager for JIRA](#) add-on. In addition to moving JIRA projects, the described approach can be used for consolidating efforts such as merging multiple JIRA instances.

- [Architecture Strategy Recommendations](#)
- [Planning](#)
- [Stages](#)
- [Limitations & Workarounds](#)
- [Common issues](#)
- [Frequently Asked Questions](#)
- [Need Help?](#)

Definitions

For this document, we'll assume the following definitions:

- **Development** – A free-for-all one or many environments where users can play with cutting-edge or risky changes.
- **Staging** – A pre-production environment, where the systems administration team can establish exact procedures prior to rollout. The staging should be a clone or close replica of the production environment.
- **Production 1 (Source)**: Your live instance where the project(s) originate, expecting minimal downtime and well-tested changes.
- **Production 2 (Target)**: Your live instance where the project(s) should be moved, expecting minimal downtime and well-tested changes.
- **Configuration snapshots** are created using the [Configuration Manager for JIRA](#) add-on and represent the state of your JIRA [configuration objects](#) and their relations to each other at a given point in time. There are two types of configuration snapshots:
 - **Project snapshot** – Contains the configuration of a number of selected projects (with their schemes, workflows, fields, etc.)
 - **System snapshot** – Contains the entire configuration of a single JIRA instance (projects, workflows, schemes, screens, etc.)

Architecture Strategy Recommendations

Data

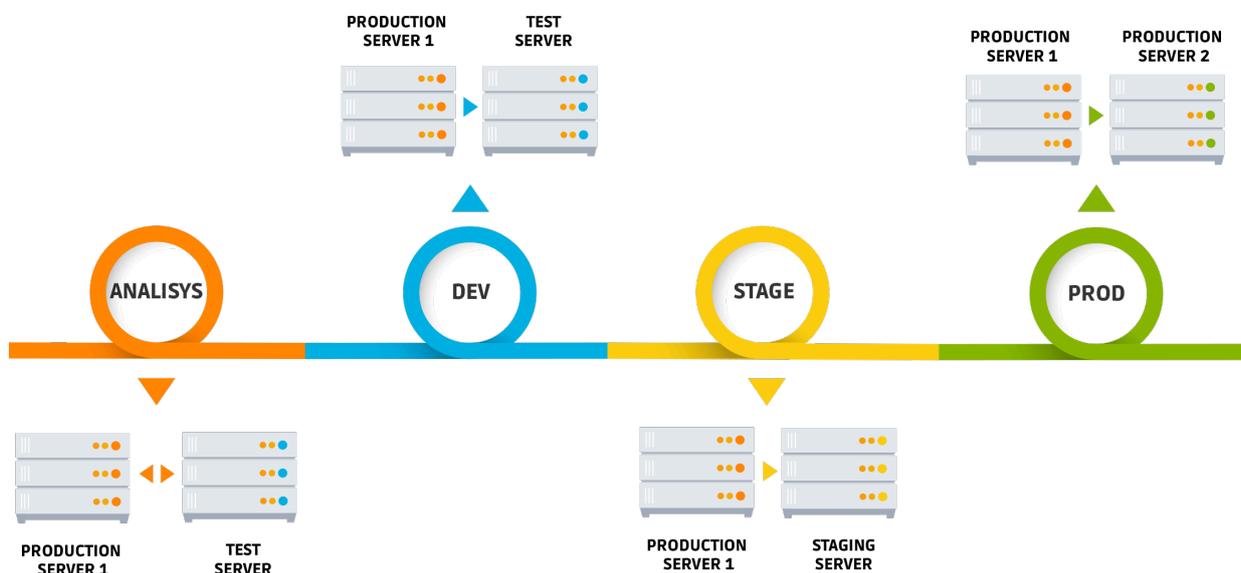
This guide presents how to run a transparent, end-to-end migration with no data loss. The majority of the [data objects](#) are handled out-of-the-box by the Configuration Manager for JIRA. All exceptions and ways to handle

them are listed in the [Limitations](#) sections of this guide.

Process

The moving of projects between **Source** (Production 1) and **Target** (Production 2) instances will result in changes in your **Target** (Production 2) instance. We recommend that you follow the [Test -> Staging -> Production](#) procedure. Before the migration, a significant analysis should be performed. These activities are grouped under the Analysis phase in this guide. The major difference between the process of moving projects and rolling-out configuration changes is that in the first case scenario all user-generated data needs to be transferred, which requires a significant up-front analysis. The process of moving multiple projects between JIRA instances is comprised of four major stages:

- Analysis - during this stage an analysis of the conflicts and gaps between the project configurations is performed; see [Application Migration Specification](#).
- Development - during this stage **Migration Procedure** document is developed and tested on the test server.
- Staging - during this stage the Migration Deployment procedure is staged and tested on the staging server.
- Production - during this stage the migration deployment procedure is executed in the official production environment.



Planning

The following planning steps described in the [Test-Staging-Production](#) procedure use case should be completed before the migration:

- Prepare the staging environments & keep them synchronized
- Install Configuration Manager for JIRA
- Track changes with change request tickets
- Plan disruptive & non-disruptive changes during maintenance windows
- Prepare communication strategy

Additionally, you should include the following planning steps:

- Prepare backup of both production servers
- Assess the condition of both, source and target, instances and fix any detected issues. System health assessment should be performed with [JIRA Integrity checker](#) and Configuration Manager for JIRA [Integrity Check](#).

Stages

Stage One: Analysis

Overview of the first stage:

JIRA Instances: Production 1 (or a clone), Test Server (clone of Production 2)

Goals: Prepare Application Migration Specification (AMS) document

Users: JIRA System Administrator, AD admin, DBA.

Tools Required: [Configuration Manager for JIRA](#)

Below is the list of recommended analysis. Note that any changes that need to be made should be included in the AMS document:

1. Add-ons and Versions

Both Prod 1 and Test Server meet the following requirements:

- Running the same JIRA version
- Have the same set of add-ons with matching versions

2. User Management Normalization

If both production servers are using the same external user directory, there should be a full match between the users and groups on both servers. If the match is not full, an analysis should be performed to identify gaps and conflicts.

Example

- Gap:** On Prod1 Jane Smith has username *jsmith*, while on Prod 2 server her username is *jane.smith*
- Conflict:** Both servers have user John Smith with username *jsmith*, but the usernames correspond to two different people

Similar issues apply to user groups.

Typically, the list of conflict and potential matches should be prepared and then the business users should provide information how to resolve it.

Warning

If the user management is not normalized, a wide range of negative effects will take place. All fields where users or groups are used - Assignee, Reporter, etc. might contain wrong user or group. All configuration objects with users or groups could end up improperly configured - permissions, notifications, workflow customization with users and user group fields, filter and dashboard ownership, Agile board administrators, JQL filters with users or user group fields, issue security schemes etc.

How to resolve gaps and conflicts

The general approach for gaps and conflicts is to rename the username either on the Prod 1 or Prod 2 servers to ensure consistency. The renaming depends on the solution used for user management Crowd, AD, LDAP.

3. "As is" strategy

In most cases, the issues and project configuration are moved to the new production server "as is" without any modifications. Any modifications will need to be transformed to ensure consistency on both JIRA instances. This process will ultimately add significant complexity and time to the migration and should be included in the AMS document. For this guide, an "As Is" strategy is used. For more information on the migration process with transformations, contact Botron's service team.

Example

The Bug issue type might be part of different workflows on the two server instances. When you move a project with that issue, users have to choose between two options. In the "As is" case, the various projects use different workflows for the same issue type. In the *transformat ion* case, the two projects will use the same workflow, thus all the source data will be transformed to match the new workflow.

4. Configuration conflicts and gaps

When migrating projects their names or keys might already exist in the Prod 2 server, this conflict should be resolved by renaming them on the Prod 1 server. The same can happen for custom fields, resolution, priorities, workflow names and schemes. The full list of conflicts and the mappings of their resolutions should be included in the AMS document. The reverse scenario is possible too when two custom fields on Prod 1 and Prod 2 have different names but the same semantics. In this case, they ought to be merged during the migration.

Tips

1. This analysis of these configuration conflicts and gaps can be done with the Configuration Manager for JIRA change and impact analysis feature. Prior to the migration, the Configuration Manager for JIRA enables you to perform a change and impact analysis and provides you with comprehensive details regarding the impact of the changes that will be applied to your JIRA instance. By doing this analysis, you will be able to see the projects that will be affected by the change, identify any conflicts and gaps and resolve them. The list of introduced changes and their corresponding conflicts and gaps are grouped in tabs in the same order following that of a JIRA menu.
2. This analysis is a great opportunity to optimize the usage of custom fields and reuse them between the two system instead of creating new ones. This will greatly improve the performance of the server.

5. Default Schemes

Any default schemes used on the Prod 1 server by the migrated projects should be changed. To change the default schemes:

- a. Copy the used default scheme.
- b. Rename the newly created scheme.
- c. Associate the project(s) with the new scheme.

6. Quality Strategy

A quality strategy should be developed and agreed upon by all stakeholders. The recommended quality strategy consists of test plans for each of the phases, and a post-migration remediation plan.

7. Test Plans

Development phase a set of tests should be executed to verify the following:

1. Filters - number of issues, issue ordering, column layout
2. Dashboards - layout (columns, rows), gadgets & gadget content
3. Agile
 - a. Board views (i.e. backlog, active sprints)
 - b. Issues, ranking, epics, versions, estimates, time tracking data aggregations
 - c. Quick filters
 - d. Sprints ordering & content
 - e. Reports
 - f. Project <-> Board associations
4. Issues
 - a. Agile data - sprints, epic links
 - b. Comments
 - c. History
 - d. Field values - 3-rd party custom fields, time tracking and estimates, other
 - e. Issue links
 - f. Confluence links
 - g. Bitbucket links (branches, commits)
5. Security
 - a. Projects - role memberships, access and operations for different roles
 - b. Boards, Filters, and Dashboards - ownership and operations
6. Operations - transition through workflow, create/deleted/edit issues, comments ,etc.

Staging phase - the tests from the development phase are executed again and a User Acceptance Test (UAT) is performed by the users of the projects that is being migrated.

Production phase, a subset of the development phase tests are executed plus a limited UAT test. Typically, there is a time limit on how long can these tests be performed in production. This time limit is aligned with the duration of the maintenance window during which the production phase takes place.

Post Production Remediation Plan - this plan covers the procedures designed to handle any post-migration issues that occurred during the production. It includes a SLA for response and resolution.

Stage Two: Development

Overview of the second stage:

JIRA Instances: Clone of Production 1, Test Server (clone of Production 2)

Goals: Prepare Migration Procedure document

Users: JIRA System Administrator, AD admin, DBA.

Tools Required: Configuration Manager for JIRA

Each of the migration tasks should be properly documented and put in an ordered list. It should include any input data and enough details so that it can be repeated consistently.

The recommended approach for documenting is a structure of tasks and steps to accomplish each task.

1. For each of the data or configuration changes included in the AMS document, execute the required steps.
2. **Create project snapshot** that includes the projects that are being migrated from **Prod 1 server**. Include all related to the projects Agile boards, filters, dashboards and issues

Issues support is still in Beta mode. We're making our best to provide official support for this with Configuration Manager 5.0.

Create Configuration Snapshot

Select Filters Boards Dashboards Preview Create

Preview Snapshot

Details

Name:	Type:	Description:
asd	Project (HSP)	N/A

▼ Filters (3 of 5)

Name	Owner	
My Approvals	Administrator	Exclude
My Milestones	Administrator	Exclude
My Tasks	Administrator	Exclude

« < 1 > »

▼ Boards (2 of 2)

Name	Owner	
TSP board	admin	Exclude
SOM board	admin	Exclude

« < 1 > »

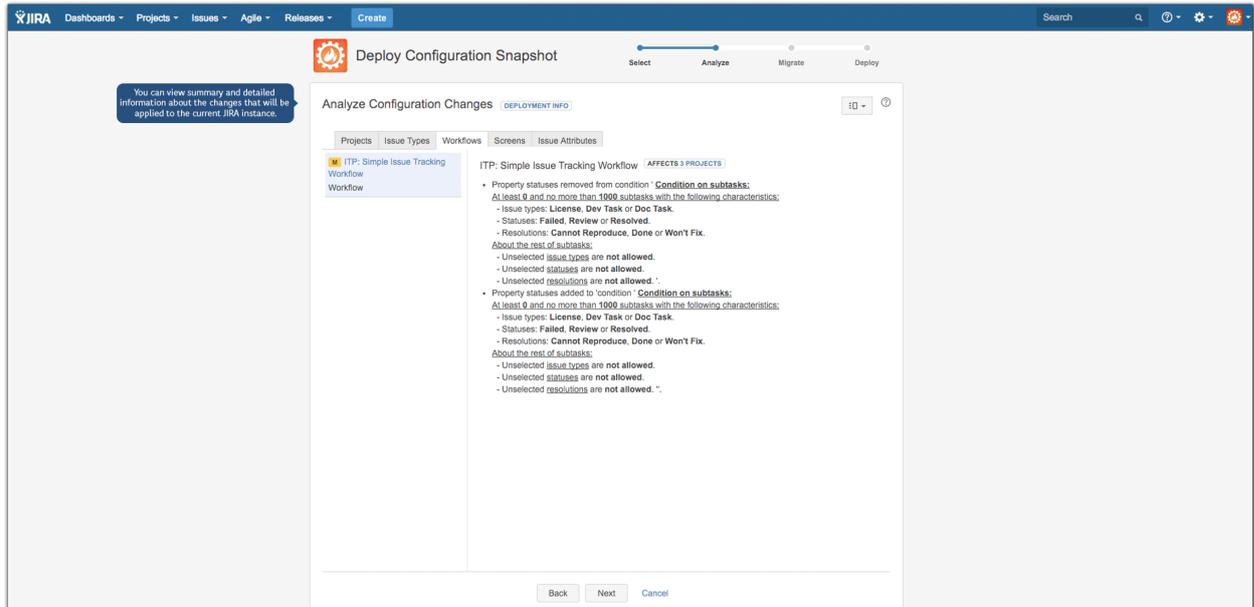
▼ Dashboards (2 of 2)

Name	Owner	
Dashboard 1	Administrator	Exclude
System Dashboard		Exclude

« < 1 > »

Back Create Cancel

3. **Download the configuration snapshot.** If your JIRA configurations are being tracked via tickets, the snapshot should be attached to the appropriate tickets.
4. **Deploy configuration snapshot to Test Server** and validate proposed configuration changes.



5. It is crucially important to make sure the proposed configuration **changes and impact** match the AMS.
 - a. If they don't match AMS, the changes made in step 2 need to be corrected. Restore your backup on the clone of Production 1 and the Test Server and resume from step 1.
 - b. If they match AMS, continue.
6. Update the ticket with the required deployment time for the snapshot deployment.
7. Execute the test plan for test stage
 - a. If the tests fail, they have to be resolved. Restore your backup on the clone of Production 1 and the Test Server and resume from step 1.
 - b. If the tests pass, attach the Migration procedure to the migration ticket and proceed to Stage 3.

The steps described in this stage should be repeated several times to ensure that everything runs smoothly and both the proposed changes and the associated tests are successful.

Stage Three: Staging/QA

Overview of the third stage:

JIRA Instance: Production 1 (or a clone), Staging Server (close replica of Production 2)

Goals: This is the grand rehearsal of the migration and has two major goals:

1. Verify the migration procedure and conduct an UAT.
2. Get an accurate time estimate for the duration of the migration.

Users: JIRA System Administrator, AD admin, DBA, business users to conduct UAT.

Tools Required: Configuration Manager for JIRA

The **steps** of this stage are as follows:

1. Execute each of the tasks from the migration procedure without any modifications.
 2. Estimate the required downtime.
 3. Perform development stage tests.
 4. Perform the UAT tests.
- If both the development phase test and the UAT are successful, declare successful staging and proceed to production.
 - In the case of failure, depending on the type of issue(s), either change the deployment procedure and repeat the staging or go back to development phase to correct the issue(s).

Stage Four: Production Deployment

JIRA instance: Production 1, Production 2

Goals: Finish the migration in the official Production 2

Users: JIRA System Administrator, AD admin, DBA.

Tools Required: [Configuration Manager for JIRA](#)

The fourth and final stage of the migration process is, for the most part, identical to the staging stage. The major difference being that you'd be working with the actual Production 1 and 2 servers, not their clones.

The **steps** of this stage are as follows:

1. Create full backups of both production servers.
2. Restrict the user access to Production 2 server.
3. Execute each of the tasks from the migration procedure as they are captured without any modifications.
4. Execute development stage tests.
5. Execute the UAT tests.
6. If both the development phase test and the UAT are successful, declare successful migration and open the access for the users.
7. If not successful, depending on the type of issues found there are two options:
 - a. Apply changes/fixes to Production 2 server and rerun the UAT tests.
 - b. Restore the backups and reschedule the production deployment until the issues are resolved.

Limitations & Workarounds

There are certain **system limitations** that need to be considered:

- **Add-ons data:** Certain add-on custom field configuration, add-on workflow property configuration any other add-on data outside of JIRA configuration objects might not be included. A custom solution needs to be developed.
- **JIRA Service Desk:** JIRA Service Desk-specific configuration is not supported OOTB. It will be added in future releases of Configuration Manager.

Suggested **workarounds** regarding limitations and other known issues include:

- **Scripts** - API level scripts can be developed to migrate any data not handled by Configuration Manager for JIRA. Great choice for scripts development is using the [ScriptRunner](#) add-on.
- **Manual** - If the amount of configuration is not extensive it can be manually added to the target JIRA.
- **Professional Services** - Botron's team of solution architects can develop a custom solution that migrates any type of data.

Common issues

We have identified several common issues during deployments, which could negatively impact the data portability process. Some of the more common issues include:

- **Integrity Check** errors which prevent the snapshot creation/deployment. In order to preserve the integrity of the target/production server. Configuration Manager [Integrity check](#) is executed every time a snapshot is created or deployed. All critical errors reported should be resolved in order to continue. Details information can be obtained [here](#).
- **Differences in JIRA versions:** differences in source-target JIRA versions, specifically major versions.
- **Application permissions/licenses:** for 7.x, e.g. deploying a software project when the user performing the deployment doesn't have permissions to create software projects; or the JIRA Software application isn't installed/licensed
- **Proxy/firewalls:** this may be problematic for large instances when creating a snapshot or before the [Analysis](#) phase when deploying, otherwise there shouldn't be any issues with the deployment itself. This is worked around by increasing any timeout limits on the proxy servers and disabling the firewall blocking rules.

Frequently Asked Questions

1. Is my entire configuration and all the issues are going to be moved to production automatically using Configuration Manager for JIRA?
Yes. The entire configuration, captured in the configuration snapshot, will be rolled-out. All the issues to the selected projects as well. Some add-ons configuration and data which are not

supported by Configuration Manager for JIRA won't be moved. The supported objects are listed [here](#).

2. Can I move projects to a server with a newer server instance to an older server instance?
It is strongly recommended that both productions servers are on the same version. The opposite adds unnecessary complexity.
3. How long does it take to move multiple projects?
Depending on the user management normalization, the amount of data and the other analysis. It can take anywhere from several hours to few weeks.
4. Can custom add-ons data be moved using Configuration Manager?
Yes. Configuration Manager can be extended to handle any custom add-ons. Contract Botron's [services](#) team for more information.

Need Help?

If you cannot resolve the problem yourself, you can contact us for assistance via the following communication channels

- Send email to support@botronsoft.com
- For training or services specific information contact at services@botronsoft.com

Consolidating multiple JIRA instances

The process of consolidating multiple JIRA instances is exactly the same as the migration process described in Migrating JIRA projects.

For the complete documentation on Migrating JIRA projects, see [here](#).

While the end result is different, the approach, tools, planning and overall staging procedure is the same. To consolidate multiple JIRA server into one, you need to use the Configuration Manager for JIRA to merge system and project level configuration data, user-generated data, 3rd party add-on configuration and other related information. To ensure consistency in your merger, a comprehensive analysis, testing and staging of the consolidation efforts should be performed prior to rolling out the changes of your final merger. Any errors, gaps, conflicts and merge-specific issues that you identify should be resolved during the analysis phase of the consolidation process.

If you face any challenges during your consolidation efforts or cannot resolve the problem on your own, feel free contact us for assistance through the following communication channels:

- Send email to support@botronsoft.com
- For training or services specific information contact at services@botronsoft.com

Configuring JIRA application emails

To get the most out of your JIRA applications, you can configure them to send email notifications when an event occurs, and to perform tasks on the receipt of an email.

Search the topics in 'Configuring JIRA application emails':

Sending emails

Your JIRA applications can send emails when an event occurs, such as an issue is created or completed, or when the issue is updated. These are called email notifications.

[Learn more](#) about configuring the email notifications and customizing the content.

Receiving emails

Your JIRA applications can be configured to perform tasks when they receive emails. You can choose to allow your applications to create new issues when an email is received, or update an existing issue with a comment.

[Learn more](#) about configuring the incoming mail options and creating mail handlers.

Configuring email notifications

JIRA can send email notifications to users when significant events occur (e.g. creation of an issue; completion of an issue).

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

On this page:

- [Email notifications](#)
- [Configuring a project's email address](#)
- [Email recipients](#)
- [Email HTML formatting](#)
- [Troubleshooting email notifications](#)

Email notifications

Enabling email notifications

It is possible to customize your [email content](#). The [email address](#) from which notifications are sent can also be configured for each project. See [Configuring an SMTP mail server to send notifications](#) and [Creating a notification scheme](#) for more information.

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.

Configuring a project's email address

It is possible to configure a project's email address, which is the email address that notifications are sent from – i.e. the 'sender address'. This will also serve as the reply address for responses, which can work in conjunction with [Creating issues and comments from email](#).

By setting the **Sender 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 **From address** specified in the **SMTP Mail Server** configuration is used as the default **Sender Address** for all projects.

The 'Sender Address' for a project can be configured as follows:

1. Choose



> **Projects**, and select the relevant project. The 'Project Summary' page (see [Defining a project](#)) for your selected project is shown.

2. At the lower-right section of the 'Project Summary' page, locate the **Notifications** section and click the 'pen' icon to the right of the **Email** address.



Notifications

JIRA can notify the appropriate people of particular events in your project, e.g. "Issue Commented". You can choose specific people, groups, or roles to receive notifications.

Scheme: [Angry Nerds Scheme](#)

Email:  jira@jdog.atlassian.net

3. In the resulting **Project Email Address** dialog box, enter a valid email address in the **Sender**

Address field, and click **Update** to complete the process. This email address will now be used as the 'sender' address in all email notifications sent by this project.

Note: You can reinstate the default email address (as specified in the **SMTP Mail Server** configuration) by re-editing the **Sender Address** field (in the **Project Email Address** dialog box) but leaving it blank.

 You cannot specify a project's email address until an **SMTP Mail Server** has been previously configured. See [Configuring an SMTP mail server to send notifications](#) for more information.

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. the **Project Lead** and current reporter) for one event notification, the user will only receive the first encountered email notification. JIRA will log the fact that this user was on multiple recipient lists.

JIRA's default setting is to not notify users of their own changes. This can be changed on a per user basis via their profile preferences.

Email HTML formatting

Each JIRA user can specify in their profile preferences, whether to send outgoing emails in either text or HTML format. **JIRA Administrators** can specify a default email format by choosing the **cog icon**



at top right of the screen, then **User Management > User Preferences**.

The HTML email format can accommodate internationalized words in the 'Issue Details' section. However, due to Internet Security Settings, which prevent images from being automatically downloaded, the HTML email messages 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 email clients, it is possible to do this in two different ways:

1. Per email message:

- **Mozilla Thunderbird** — by clicking on the 'Show Remote Content' button above the email.
- **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 email.
- **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 email.

2. Configuring the email 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.

Troubleshooting email notifications

Using the JIRA admin helper

The JIRA admin helper can help you diagnose why a user isn't receiving email notifications when they should be, or why a user is receiving email notifications when they shouldn't be. This tool is only available to JIRA administrators.

To diagnose why a user is or is not receiving notifications for an issue:

1. View the issue in JIRA.

2. Click **Admin > Notification Helper**.
3. Enter the username of the user.
4. Click **Submit**.

Tip: You can also access the Notifications Helper via the cog menu for each issue in the issue navigator, or by selecting the **cog icon**



at top right of the screen, then **Add-ons**. Select **Admin Helper > Notification Helper** to open the following page.

The screenshot shows the JIRA Administration interface. At the top, there is a search bar for 'JIRA admin' and a navigation menu with tabs for 'Projects', 'Add-ons', 'User Management', 'Issues', and 'System'. The 'Add-ons' tab is active. On the left, there is a sidebar menu with categories: 'ATLASSIAN MARKETPLACE' (with links for 'Find New Add-ons' and 'Manage Add-ons'), 'CLOUD CONNECTORS' (with links for 'Connections', 'Mappings', 'Mapping Schemes', 'Remote Issue Links', 'Diagnostics', 'Support', and 'Licensing'). The main content area is titled 'Notification Helper' and contains the following form:

Notification Helper
Find out why users receive, or do not receive notifications for this issue

User:
Begin typing to find a user

Issue:
Begin typing to find an issue

Notification Event:
Begin typing to find a notification event or press down to see all

Configuring an SMTP mail server to send notifications

To enable JIRA to send [notifications](#) about various events, you need to first configure an SMTP mail server in JIRA .

Note: For all of the following procedures, you must be logged in as a user with the **JIRA System Administrators** [global permission](#). **JIRA Administrators** can enable or disable outgoing mail, but not configure SMTP mail servers.

On this page:

- Define or edit the SMTP mail server
- Specify a host name or JNDI location for your SMTP mail server
- Configuring a JNDI location
- Troubleshooting

Define or edit the SMTP mail server

1. Choose



> **System**.

2. Select **Mail > Outgoing Mail** to open the SMTP Mail Server page.

i If no SMTP mail server has been defined, then a **Configure new SMTP mail server** button will be shown on the page. If one has already been defined, then the SMTP mail server's details will be shown on the page, along with a set of operation links at the right.

3. Click either the **Configure new SMTP mail server** button to define a new SMTP mail server, or the **Edit** link at the right to edit the existing SMTP mail server, which will open the **Add/Update SMTP Mail Server** page.

4. Complete the top section of this page as follows:

Name	Specify an arbitrary name to identify this SMTP mail server configuration.
Description	(Optional) Specify an arbitrary description that describes the SMTP mail server. This description appears below the Name of the SMTP mail server on the SMTP Mail Server configuration page.
From address	Specify the email address used in the 'sender address' (or 'from') field of notification messages sent by JIRA, unless overridden in a project configuration . Only specify an email address for this field (e.g. <code>jira@example-company.com</code>). JIRA will use this value to construct the full 'from' header based on the current user ("Joe Bloggs (JIRA) <jira@example-company.com>"). To change the 'from' header, go to Administration > System > General Configuration and (under Settings), edit the Email from field.
Email prefix	Specify 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 email notifications from JIRA based on this prefix.

Screenshot: Add (or Update) SMTP Mail Server

Add SMTP Mail Server ?

Use this page to add a new SMTP mail server. This server will be used to send all outgoing mail from JIRA.

Name *
The name of this server within JIRA.

Description

From address *
The default address this server will use to send emails from.

Email prefix *
This prefix will be prepended to all outgoing email subjects.

Server Details
Enter *either* the host name of your SMTP server or the JNDI location of a `javax.mail.Session` object to use.

SMTP Host

Service Provider

Protocol

Host Name *
The SMTP host name of your mail server.

SMTP Port
Optional - SMTP port number to use. Leave blank for default (defaults: SMTP - 25, SMTPS - 465).

Timeout
Timeout in milliseconds - 0 or negative values indicate infinite timeout. Leave blank for default (10000 mSecs).

TLS
Optional - the mail server requires the use of TLS security.

Username
Optional - if you use authenticated SMTP to send email, enter your username.

Password
Optional - as above, enter your password if you use authenticated SMTP.

or

JNDI Location

JNDI Location
The JNDI location of a `javax.mail.Session` object, setup by your application server.

Specify a host name or JNDI location for your SMTP mail server

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

- specify the **SMTP host** details of your SMTP mail server;
- or:**
- specify the **JNDI location** of a `javax.mail.Session` object — that is, use JNDI to look up an SMTP 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:** for instance, you could switch to RSET instead of NOOP for testing connections by setting the `mail.smtp.userSet` property.
 - **Centralized management:** mail details are configured in the same place as database details and may be configured through your application server administration tools.

Specify the SMTP host details

Most people configure JIRA's SMTP mail server by specifying the SMTP host details of this mail server directly in JIRA.

1. In the **SMTP host** section of the **Add/Update SMTP Mail Server** page ([above](#)), complete the following form fields:

Service Provider (not available when updating an existing SMTP mail server)	Choose between using your own SMTP mail server (i.e. Custom), or either Gmail (i.e. Google Apps Mail / Gmail) or Yahoo! (i.e. Yahoo! Mail Plus) as the service provider for your SMTP mail server. If you choose either Gmail or Yahoo! options and then switch back to Custom , some of the key fields in this section will automatically be populated with the relevant SMTP mail server settings for these service providers.
Protocol	Choose between whether your SMTP mail server is a standard (i.e. SMTP) or a secure (i.e. SECURE_SMTP) one.
Host Name	Specify the hostname or IP address of your SMTP mail server. Eg. <code>smtp.yourcompany.com</code>
SMTP Port	<i>(Optional)</i> The SMTP port number, usually 25 for SMTP or 465 for SMTPS, either of which are assumed if this field is left blank.
Timeout	<i>(Optional)</i> Specify the timeout period in milliseconds, which is treated as 10000 if this field is left blank. Specifying 0 or a negative value here will result in JIRA waiting indefinitely for the SMTP server to respond.
TLS	<i>(Optional)</i> Select this checkbox if your SMTP host uses the Transport Layer Security (TLS) protocol.
Username	<i>(Optional)</i> If your SMTP host requires authentication, specify the username of these authentication credentials here. (Most company servers require authentication to relay mail to non-local users.)
Password	<i>(Optional)</i> Again, if your SMTP host requires authentication, specify the password associated with the username you specified above. When editing an existing SMTP mail server, select the Change Password checkbox to access and change this field.

Please 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`
 - The SMTP must support the multipart content type. Without this mails will not be able to send.
2. *(Optional)* Click the **Test Connection** button to check that JIRA can communicate with the SMTP mail server you just configured.

3. Click the **Add** (or **Update**) button to save JIRA's SMTP mail server configuration.

Specify a 'JNDI Location'

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

In the **JNDI Location** section of the **Add/Update SMTP Mail Server** page (above), specify the location of a `javax.mail.Session` object to use when sending email, in the **JNDI Location** field. This will begin with the prefix `java:comp/env/`

Configuring a JNDI location

The **JNDI Location** that you specify in JIRA will depend on JIRA's application server and configuration. JNDI locations are typically configured in the application server that runs JIRA. Hence, JIRA will need to be restarted after configuring a JNDI location for that configuration to be available in JIRA.

For example, in Tomcat 6 (the application server bundled with 'recommended' distributions of JIRA), your **JNDI Location** would be `java:comp/env/mail/JiraMailServer` and you would add the following section to the `conf/server.xml` of your [JIRA application installation directory](#), 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 do not 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>
```

If you happen to be running JIRA on an application server other than Apache Tomcat (which is [not a supported JIRA configuration](#)), a similar methodology for configuring a JNDI location to your SMTP mail server should apply to that application server.

If you have problems connecting, add a `mail.debug="true"` parameter to the `<Resource/>` element (above), which will let you see SMTP-level 'debugging' details when testing the connection.

Move the JavaMail Classes

You will also need to ensure that the JavaMail classes (typically in JAR library files) are present in your application server's classpath and that these do not conflict with JIRA's JAR library files. This is necessary because the application server itself (not JIRA) is establishing the SMTP connection and as such, the application server can not see the JAR library files in JIRA's classloader.

Some operating systems may bundle the JavaMail classes with application servers (e.g. **Tomcat in Red Hat Enterprise Linux**). This may conflict with JIRA's copy of the JavaMail classes, resulting in errors like:

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

or:

```
java.lang.IllegalArgumentException: Mail server at location
[java:comp/env/mail/JiraMailServer] is not
of required type javax.mail.Session.
```

Lighter application servers such as Apache Tomcat (including the one incorporated into the 'recommended' distributions of JIRA), do not always come with JavaMail.

To prevent any conflicts, check your application server's `lib/` directory:

- If the application server already contains `mail-1.4.1.jar` and `activation-1.1.1.jar`, then just **remove** `mail-1.4.1.jar` and `activation-1.1.1.jar` from the `<jira-application-dir>/WEB-INF/lib/` subdirectory of the [JIRA application installation directory](#).
- If the application server does not contain `mail-1.4.1.jar` and `activation-1.1.1.jar`, then **move** the `mail-1.4.1.jar` and `activation-1.1.1.jar` from the `<jira-application-dir>/WEB-INF/lib/` subdirectory of the [JIRA application installation directory](#) into the `lib/` subdirectory of the JIRA installation directory (for 'recommended' distributions of JIRA) or the `lib/` subdirectory of the application server running JIRA.

SMTP over SSL

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

Firstly, you will need to **import the SMTP server certificate** into a Java keystore. The process is described on the [Configuring an SSL connection to Active Directory](#) page.

 **Important Note:** Without importing the certificate, JIRA will not be able to communicate with your mail server.

Secondly, edit your mail server connection properties and specify `starttls` and `SSLSocketFactory`. From `{ $JIRA_INSTALL }/conf/server.xml` (this example uses Gmail's server):

```
<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"
/>
```

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.

Common Problems

- If JIRA does not appear to be creating or sending emails or creating issues and comments from email, your JIRA installation could be experiencing **OutOfMemory errors**. Please check your log files for OutOfMemory errors. If there are OutOfMemory errors, please restart JIRA and [investigate the errors](#).
- If you find some incoming emails simply disappear, check that you have not accidentally **started a second copy of JIRA** (eg. in a staging environment) which is downloading and deleting email messages. See the [Restoring data](#) page for flags you should set to prevent mail being processed.
- If you receive 'Mail Relay' errors, make sure you have specified the **Username** and **Password** in the **SMTP Host** section of JIRA's **SMTP Mail Server** configuration page.

Getting Help

If you cannot resolve a problem yourself, please [create a support case](#) in the 'JIRA' project and we will assist.

Customizing email content

JIRA generates emails in reaction to events using a templating engine. The templating engine is Apache'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.

Please Note:

- To change the columns in your filter subscriptions, you don't need to customize the mail templates.
- There's a feature request to improve this at [JRA-7266](#), which you can vote on to improve its chances of being implemented.
- Bear in mind that the next time you upgrade JIRA – or need a new installation for any reason – you will have to manually copy any changes you have made to Velocity templates (as well as JSPs) into the new installation of JIRA. If the Velocity templates and/or JSPs have changed in the newer version, you will have to manually port your customizations into them (as opposed to copying these files directly over from your old JIRA installation to your upgraded one).

Customizations to Velocity templates or other JIRA files are not included in the scope of [Atlassian Support](#).

Email template locations

1. Open up your JIRA distribution, and navigate to the following paths:
 - The `WEB-INF/classes/templates/email/` of the `<jira-application-dir>` in your [JIRA installation directory](#).
 - The `jira/src/etc/java/templates/email/` in your extracted JIRA source directory.

2. Under this directory there are three directories: `html`, `text` and `subject`. The `html` subdirectory contains the templates used to create emails in html, while the `text` directory the plain text mail outs. The `subject` directory contains the templates used to generate the subject of the emails. The templates are named after the event that will trigger the email.
3. Bring the template up in your favorite text editor. Referring to the [JIRA template documentation](#) (particularly [Velocity Context for Email Templates](#)) and [Velocity Users Guide](#), make the customizations you want.
4. Restart JIRA.

For new email templates:

1. Create your new `mytemplate.vm` files in the `html`, `text` and `subject` directories, based on the existing files in those directories
2. Add the templates to `atlassian-jira/WEB-INF/classes/email-template-id-mappings.xml` to make them valid choices for when you are adding a new event.

Note that since JIRA 4.1 each new template has to have a corresponding file in the `subject` directory.

Advanced customization

The `Issue` object is passed into the `vm` templates. Notice some of its implementation in `/includes/summary-topleft.vm`. As an example, calling `$issue.getProject()` would allow you to determine the project an issue comes from, and even create logic to show different information for emails from different projects.

Deploying Velocity templates without restarting JIRA

In a *development* instance, you can play with picking up velocity file changes without a restart. From `<jira-install>/atlassian-jira/WEB-INF/classes/velocity.properties`:

1. Change `class.resource.loader.cache` from `true` to `false`
2. Remove the comment sign (`#`) from `#velocimacro.library.autoreload=true`

Making this change in production will eventually lead to JIRA not serving pages along with the 'ran out of parsers' error in the log file.

See also [Adding Custom Fields to Email](#).

Creating issues and comments from email

Admins can configure JIRA to receive and process emails. JIRA can receive emails from licensed users to create issues or add comments and attachments to existing issues automatically.

If you're looking for a help desk solution, it may be more practical to use JIRA Service Desk, rather than setting up JIRA Core or JIRA Software for this purpose.

JIRA Service Desk uses a built-in processor to receive and process issue requests from emails. Issues created in JIRA Service Desk don't require the sender to have a license to create, view, comment, add attachments, or transition issues. [Read more about receiving email requests with JIRA Service Desk](#).

[Learn how to download and set up JIRA Service Desk for your instance](#).

On this page:

- [Configuring issue or comment creation from email](#)
- [Mail handlers](#)
- [Issue/comment creation](#)
- [Handy tips with mail handlers](#)
- [Best practices \(pre-processing JIRA email messages\)](#)
- [Troubleshooting](#)

Configuring issue or comment creation from email

Issues and comments in JIRA can be generated either from:

- email messages sent to an account on a POP or IMAP mail server, or
- messages written to the file system generated by an external mail service.

Note that for all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Step one: Configure a mail server/service

POP or IMAP email messages

To set up issue and comment creation from email, you will need to create a mail account for a POP or IMAP mail server that JIRA can access – typically, one mail account for each JIRA project. For example, for the 'ABC' project, you might establish an account `abc-issues@example.com`

JIRA will periodically scan for new email messages received by your mail account (via a service) and appropriately create issues or comments for any emails it finds (via a mail handler).

JIRA's mail handlers can also *optionally* create new user accounts for senders not previously seen. See the [Create a new issue or add a comment to an existing issue](#) section for more details.

 Note that this is not possible if you are using [External User Management](#).

Once you have created a mail account on a POP or IMAP mail server, [configure JIRA to receive email from that mail server account](#).

 **Tip:** You can configure JIRA's mail servers so that recipients of email notifications can simply reply to these messages and have the body of their replies added as comments to the relevant issue. To do this, simply set the **From address** in [JIRA's SMTP mail server](#) to match that of the POP or IMAP mail server's account being monitored. In most cases, this means having JIRA's SMTP and POP or IMAP mail servers use the same mail account. See [below](#) for details on how to configure JIRA to handle these emailed replies.

File system messages

To set up issue and comment creation from messages written to the file system by an external mail service, your external mail service must be able to write these messages within the `import/mail` subdirectory of the [JIRA home directory](#).

External mail services are very much like the POP or IMAP services above, except that instead of email messages being read from a mail account, they are read from a directory on the disk. External mail services are useful because they overcome the potential security risks associated with anonymous mail accounts. Instead you can simply configure your external mail service to dump incoming email messages within the JIRA Home Directory's `import/mail` subdirectory, which is scanned periodically.

Please also be aware that JIRA expects only one message per file, so your external mail service should be configured to generate such output.

 Note — how JIRA handles messages on a mail server/service:

- For mail accounts, JIRA scans email messages received by your mail account's 'Inbox' folder. However, for IMAP mail servers, you can specify a different folder within your mail account.
- If JIRA successfully processes a message, JIRA deletes the message from your mail account (on a POP) or file system (i.e. for file system messages). On an IMAP mail server processed messages are not deleted but marked as read.
- If JIRA does not successfully process a message, the message will remain either in your mail account or on the file system.

Step two: Configure a mail handler

Once you have configured JIRA to receive messages from a mail server/service, you configure JIRA to handle these messages through a 'mail handler'.

1. Choose



> **System.**

2. Select **Mail > Incoming Mail** to open the Incoming Mail page.
3. Click the **Add incoming mail handler** button (or the **Edit** link next to an existing mail handler) in the **Mail Handlers** section to open the **Mail Handler** dialog box.

The screenshot shows the 'Incoming Mail' configuration page in JIRA. The 'Mail Handlers' section is active, and the 'Mail Handler' dialog box is open. The dialog box contains the following fields:

- Name:** A text input field.
- Server:** A dropdown menu with 'Local Files' selected.
- Delay:** A text input field with '1' entered. Below it, the text reads 'Delay between running time, in minutes.'
- Handler:** A dropdown menu with 'Create a new issue or add a comm...' selected. A help icon (?) is visible to the right.
- Folder Name:** A text input field.

At the bottom of the dialog box, there is a note: 'This service will retrieve data from (jira.home)/import/mail'. Below the note are 'Next' and 'Cancel' buttons.

4. Specify a **Name** that describes what your mail handler will do — for example, 'Create issues or comments from Example Company's IMAP mail server'.
5. Select the mail **Server** that you configured in step one ([above](#)). This is either a POP or IMAP mail server or the **Local Files** option for an external mail service that writes messages to the file system.
6. Specify the **Delay** (in minutes) between the mail handler's running time. This effectively defines the frequency with which JIRA scans the **Server** that you specified in the previous step.
7. Choose the type of mail **Handler** from dropdown list. For more information, refer to the [Mail Handlers](#) section below.
8. If you chose either an IMAP mail server or the **Local Files** option in the **Server** field, then a **Folder Name** field appears below the **Handler** dropdown list:
 - For an IMAP mail server, if you want mail handler to scan for new messages from a folder other than the 'Inbox' in your mail account, specify the name of that folder here.
 - For the **Local Files** option, if your file messages are being written to a subdirectory within the `import/mail` subdirectory of the JIRA home directory, specify the subdirectory structure (within `import/mail`) here.
9. Click **Next** to continue with specifying the remaining options specific to mail **Handler** you selected above. For more information, see the [Mail Handlers](#) section below.
10. (*Optional*) Click the **Test** button to test your mail handler. If you are using Local Files as the server, copy a saved email that contains a "Subject: " line to the configured directory. JIRA will remove this file after it is parsed, or log a message about why an issue could not be created. You may have to specify the project, issuetype and reporterusername properties as a minimum configuration. A sample email file might look like this:


```
To: jira@example.com
From: some-jira-user@example.com
Subject: (TEST-123) issue summary title here
Body of the email goes here
```
11. Click the **Add / Save** button to save your mail handler.

Note — the relationship between JIRA mail handlers and services:

- A JIRA mail handler is part of a [JIRA service](#). Hence, when you create a mail handler, its service will appear as an entry on the Services page.
- Be aware that editing mail handlers can only be performed through the Mail Handlers page (described [above](#)).
- On the Mail Handlers page, clicking the Delete link associated with a mail handler removes that handler. Since a mail handler is part of a service, then if you delete a mail handler's service on the Services page, its associated handler will also be removed from the Mail Handlers page.

Mail handlers

JIRA provides the following default mail handlers:

- Create a new issue or add a comment to an existing issue
- Add a comment from the non quoted email body
- Add a comment with the entire email body
- Create a new issue from each email message
- Add a comment before a specified marker or separator in the email body

For more information about how these mail handlers create issues and comments in JIRA, refer to [Issue/comment creation](#) (below).

Also refer to the [Handy tips with mail handlers](#) (below) for tips on tweaking mail handlers to allow JIRA to handle the following types of email messages:

- Email sent from people without a JIRA user account.

Create a new issue or add a comment to an existing issue

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.

To configure a 'Create a new issue or add a comment to an existing issue' mail handler:

1. If you have not already done so, begin configuring your mail handler ([above](#)).
2. On the **Create a new issue or add a comment to an existing issue** dialog box, complete the following fields/options:

Project	Specify the project key of the default project to which new issues are created by this handler — for example, JRA. Note: <ul style="list-style-type: none"> • This field is only relevant for issue creation, not for issue commenting. • If an email message contains an issue key in its subject line and that issue key exists in your JIRA installation, the handler will add the email message content as a comment on the issue, regardless of which project the issue is in.
Issue Type	Choose the default issue type for new issues.
Strip Quotes	Select this checkbox to remove quoted text from from an email message's body (e.g. from previous email replies) before the body's content is added to the JIRA issue's comment.

<p>Catch Email Address</p>	<p>If specified, only email messages whose To:, Cc:, Bcc: lines contain the recipient specified in this field will be processed — for example, <code>issues@mycompany.com</code></p> <p>Upon specifying an address here, all email messages whose To:, Cc:, Bcc: lines contain addresses other than the Catch Email Address are ignored. This is useful if you have multiple aliases for the same mail account (e.g. <code>foo-support@example-co.com</code> and <code>bar-support@example-co.com</code> aliases for <code>support@example-co.com</code>) for multiple mail services (e.g. each one to create issues in separate JIRA projects).</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Note: in practice, this option is rarely useful and should not be confused with the more common Default Reporter. You can only specify one catch email address and one issue type per mail handler.</p> <p>In addition, there is a known bug in JIRA 7.0.0 and JIRA 7.0.1, which means that multiple email handlers that are used to create issues in different projects when an email is sent to multiple aliases will not process the email correctly. This has been fixed in JIRA 7.0.2. For more information, see</p> <p>JRASERVER-41831 - Duplicate issues creation fails - Creating multiple issues by one Email CLOSED</p> </div>
<p>Bulk</p>	<p>This option only affects 'bulk' email messages whose header has either its Precedence: field set to bulk or its Auto-Submitted field not set to no. Such messages would typically be sent by an automated service. When such an email message is received, the following action will be performed, based on the option you choose:</p> <ol style="list-style-type: none"> a. Ignore the email and do nothing. b. Forward the email (i.e. to the address set in the Forward Email text field). c. Delete the email permanently. <p>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).</p>
<p>Forward Email</p>	<p>If specified, then if this mail service is unable to handle an email message it receives, an email message indicating this problem will be forwarded to the email address specified in this field.Note: An SMTP mail server must be configured for this option to function correctly.</p>
<p>Create Users</p>	<p>Select this checkbox if you want JIRA to create new user accounts from any received email messages whose From: field contains an address that does not match one associated with an existing JIRA user account. This allows the creator of the email message to be notified of subsequent updates to the issue, which can be achieved by configuring the relevant project's notification scheme to notify the Reporter of updates.</p> <p>The username and email address of these newly created JIRA user accounts will be the email addresses specified in the From: fields of these received messages. The password for these new JIRA users is randomly generated and an email message is sent their addresses informing them about their new JIRA user account.</p> <p>Users created this way will be added to the default group/s of the default JIRA application (and therefore take up a license for this application). See the Managing groups documentation.Note: this option is not compatible with Default Reporter field option below and as such, choosing the Create Users option will hide the Default Reporter option.</p>

Default Reporter	<p>Specify the username of a default reporter, which will be used if the email address in the From: field of any received messages does not match the address associated with that of an existing JIRA user — for example, a JIRA username such as <code>email-led-reporter</code></p> <p>Note:</p> <ul style="list-style-type: none"> • This option is not available if the Create Users checkbox is selected. • Please ensure that the user specified in this field has the Create Issues project permission for the relevant Project (specified above) as well as the Create Comments project permission for the other relevant projects to which this mail handler should add comments. • When an issue is created and this option is specified, the email message's From: field address is appended in a brief message at the end of the issue's Description field, so that the sender can be identified.
Notify Users	<p>Clear this checkbox if you do not want JIRA to send out an email message notifying users whose accounts have been created by the Create Users option above.</p> <p>Note: this option only functions if the Create Users checkbox has been selected.</p>
CC Assignee	<p>Select this checkbox if you want JIRA to automatically assign the issue created to a JIRA user:</p> <ul style="list-style-type: none"> • Whose email address (registered with their JIRA account) matches the first matching address encountered in the To:, then Cc: and then Bcc: field of the email message received. • Who also has the Assignable User project permission for the relevant Project (specified above).
CC Watchers	<p>Select this checkbox if you want JIRA to automatically add JIRA users to the issue created, where those users' email addresses (registered with their JIRA accounts) match addresses encountered in the To:, Cc: or Bcc: fields of the email message received.</p> <p>Please note that when an issue is created, new JIRA users created by the Create Users option (above) <i>cannot also be added</i> to the issue's watchers list by this CC Watchers option. JIRA users must <i>already</i> exist in JIRA's userbase, and must have an email address.</p>

3. Test and save your mail handler (above).

Add a comment from the non quoted email body

This message handler creates a comment, but only uses the 'non quoted' lines of the body of the email message. A quoted line is any line that starts with a '>' or '|' symbol and such lines of text will not be added to the comment. The issue to which the comment is added is chosen from the first issue key found in the email subject. The author of the comment is taken from the address of the email message's **From:** field.

To configure an 'Add a comment from the non quoted email body' mail handler:

1. If you have not already done so, begin configuring your mail handler (above).
2. On the **Add a comment from the non quoted email body** dialog box, complete the following fields/options:

Catch Email Address	<p>If specified, only email messages whose To:, Cc:, Bcc: lines contain the recipient specified in this field will be processed — for example, <code>issues@mycompany.com</code></p> <p>Upon specifying an address here, all email messages whose To:, Cc:, Bcc: lines contain addresses other than the Catch Email Address are ignored. This is useful if you have multiple aliases for the same mail account (e.g. <code>foo-support@example-co.com</code> and <code>bar-support@example-co.com</code> aliases for <code>support@example-co.com</code>) for multiple mail services (e.g. each one to create issues in separate JIRA projects).</p> <p>Note: in practice, this option is rarely useful and should not be confused with the more common Default Reporter. You can only specify one catch email address and one issue type per mail handler.</p>
Bulk	<p>This option only affects 'bulk' email messages whose header has either its Precedence field set to bulk or its Auto-Submitted field not set to no. Such messages would typically be sent by an automated service. When such an email message is received, the following action will be performed, based on the option you choose:</p> <ol style="list-style-type: none"> Ignore the email and do nothing. Forward the email (i.e. to the address set in the Forward Email text field). Delete the email permanently.
Forward Email	<p>If specified, then if this mail service is unable to handle an email message it receives, an email message indicating this problem will be forwarded to the email address specified in this field.Note: An SMTP mail server must be configured for this option to function correctly.</p>
Create Users	<p>Select this checkbox if you want JIRA to create new user accounts from any received email messages whose From: field contains an address that does not match one associated with an existing JIRA user account. This allows the creator of the email message to be notified of subsequent updates to the issue, which can be achieved by configuring the relevant project's notification scheme to notify the Reporter of updates.</p> <p>The username and email address of these newly created JIRA user accounts will be the email address specified in the From: field 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.</p> <p>Users created this way will be added to the default group/s of the default JIRA application (and therefore take up a license for this application). See the Managing groups documentation.Note: this option is not compatible with Default Reporter field option below and as such, choosing the Create Users option will hide the Default Reporter option.</p>
Default Reporter	<p>Specify the username of a default reporter, which will be used if the email address in the From: field of any received messages does not match the address associated with that of an existing JIRA user — for example, a JIRA username such as <code>email-led-reporter</code></p> <p>Note:</p> <ul style="list-style-type: none"> This option is not available if the Create Users checkbox is selected. Please ensure that the user specified in this field has the Create Issues project permission for the relevant Project (specified above) as well as the Create Comments project permission for the other relevant projects to which this mail handler should add comments.
Notify Users	<p>Clear this checkbox if you do not want JIRA to send out an email message notifying users whose accounts have been created by the Create Users option above.</p> <p>Note: this option only functions if the Create Users checkbox has been selected.</p>

3. Test and save your mail handler ([above](#)).

Add a comment with the entire email body

This message handler creates a comment based on the entire body of the email message received. The issue to which the comment is added is chosen from the first issue key found in the email subject. The author of the comment is taken from the address of the email message's **From:** field.

To configure an 'Add a comment with the email body' mail handler:

1. If you have not already done so, begin configuring your mail handler ([above](#)).
2. On the **Add a comment with the entire email body** dialog box, complete the following fields/options:

Catch Email Address	<p>If specified, only email messages whose To:, Cc:, Bcc: lines contain the recipient specified in this field will be processed — for example, <code>issues@mycompany.com</code></p> <p>Upon specifying an address here, all email messages whose To:, Cc:, Bcc: lines contain addresses other than the Catch Email Address are ignored. This is useful if you have multiple aliases for the same mail account (e.g. <code>foo-support@example-co.com</code> and <code>bar-support@example-co.com</code> aliases for <code>support@example-co.com</code>) for multiple mail services (e.g. each one to create issues in separate JIRA projects).</p> <p>Note: in practice, this option is rarely useful and should not be confused with the more common Default Reporter. You can only specify one catch email address and one issue type per mail handler.</p>
Bulk	<p>This option only affects 'bulk' email messages whose header has either its Precedence: field set to bulk or its Auto-Submitted field not set to no. Such messages would typically be sent by an automated service. When such an email message is received, the following action will be performed, based on the option you choose:</p> <ol style="list-style-type: none"> Ignore the email and do nothing. Forward the email (i.e. to the address set in the Forward Email text field). Delete the email permanently.
Forward Email	<p>If specified, then if this mail service is unable to handle an email message it receives, an email message indicating this problem will be forwarded to the email address specified in this field.Note: An SMTP mail server must be configured for this option to function correctly.</p>
Create Users	<p>Select this checkbox if you want JIRA to create new user accounts from any received email messages whose From: field contains an address that does not match one associated with an existing JIRA user account. This allows the creator of the email message to be notified of subsequent updates to the issue, which can be achieved by configuring the relevant project's notification scheme to notify the Reporter of updates.</p> <p>The username and email address of these newly created JIRA user accounts will be the email address specified in the From: field 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.</p> <p>Users created this way will be added to the default group/s of the default JIRA application (and therefore take up a license for this application). See the Managing groups documentation.Note: this option is not compatible with Default Reporter field option below and as such, choosing the Create Users option will hide the Default Reporter option.</p>

Default Reporter	<p>Specify the username of a default reporter, which will be used if the email address in the From: field of any received messages does not match the address associated with that of an existing JIRA user — for example, a JIRA username such as <code>email-led-reporter</code></p> <p>Note:</p> <ul style="list-style-type: none"> • This option is not available if the Create Users checkbox is selected. • Please ensure that the user specified in this field has the Create Issues project permission for the relevant Project (specified above) as well as the Create Comments project permission for the other relevant projects to which this mail handler should add comments.
Notify Users	<p>Clear this checkbox if you do not want JIRA to send out an email message notifying users whose accounts have been created by the Create Users option above.</p> <p>Note: this option only functions if the Create Users checkbox has been selected.</p>

3. Test and save your mail handler ([above](#)).

Create a new issue from each email message

This message handler creates a new issue for each incoming message.

To configure an 'Create a new issue from each email message' mail handler:

1. If you have not already done so, begin configuring your mail handler ([above](#)).
2. On the **Create a new issue from each email message** dialog box, complete the following fields/options:

Project	<p>Specify the project key of the default project to which new issues are created by this handler — for example, <code>JRA</code>.</p> <p>Note:</p> <ul style="list-style-type: none"> • This field is only relevant for issue creation, not for issue commenting. • If an email message contains an issue key in its subject line and that issue key exists in your JIRA installation, the handler will add the email message content as a comment on the issue, regardless of which project the issue is in.
Issue Type	<p>Choose the default issue type for new issues.</p>
Catch Email Address	<p>If specified, only email messages whose To:, Cc:, Bcc: lines contain the recipient specified in this field will be processed — for example, <code>issues@mycompany.com</code></p> <p>Upon specifying an address here, all email messages whose To:, Cc:, Bcc: lines contain addresses other than the Catch Email Address are ignored. This is useful if you have multiple aliases for the same mail account (e.g. <code>foo-support@example-co.com</code> and <code>bar-support@example-co.com</code> aliases for <code>support@example-co.com</code>) for multiple mail services (e.g. each one to create issues in separate JIRA projects).</p> <p>Note: in practice, this option is rarely useful and should not be confused with the more common Default Reporter. You can only specify one catch email address and one issue type per mail handler.</p>

Bulk	<p>This option only affects 'bulk' email messages whose header has either its Precedence: field set to bulk or its Auto-Submitted field not set to no. Such messages would typically be sent by an automated service. When such an email message is received, the following action will be performed, based on the option you choose:</p> <ol style="list-style-type: none"> Ignore the email and do nothing. Forward the email (i.e. to the address set in the Forward Email text field). Delete the email permanently.
Forward Email	<p>If specified, then if this mail service is unable to handle an email message it receives, an email message indicating this problem will be forwarded to the email address specified in this field.Note: An SMTP mail server must be configured for this option to function correctly.</p>
Create Users	<p>Select this checkbox if you want JIRA to create new user accounts from any received email messages whose From: field contains an address that does not match one associated with an existing JIRA user account. This allows the creator of the email message to be notified of subsequent updates to the issue, which can be achieved by configuring the relevant project's notification scheme to notify the Reporter of updates.</p> <p>The username and email address of these newly created JIRA user accounts will be the email address specified in the From: field 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.</p> <p>Users created this way will be added to the default group/s of the default JIRA application (and therefore take up a license for this application). See the Managing groups documentation.Note: this option is not compatible with Default Reporter field option below and as such, choosing the Create Users option will hide the Default Reporter option.</p>
Default Reporter	<p>Specify the username of a default reporter, which will be used if the email address in the From: field of any received messages does not match the address associated with that of an existing JIRA user — for example, a JIRA username such as <input type="text" value="email-led-reporter"/></p> <p>Note:</p> <ul style="list-style-type: none"> This option is not available if the Create Users checkbox is selected. Please ensure that the user specified in this field has the Create Issues project permission for the relevant Project (specified above) as well as the Create Comments project permission for the other relevant projects to which this mail handler should add comments. When an issue is created and this option is specified, the email message's From: field address is appended in a brief message at the end of the issue's Description field, so that the sender can be identified.
Notify Users	<p>Clear this checkbox if you do not want JIRA to send out an email message notifying users whose accounts have been created by the Create Users option above.</p> <p>Note: this option only functions if the Create Users checkbox has been selected.</p>
CC Assignee	<p>Select this checkbox if you want JIRA to automatically assign the issue created to a JIRA user:</p> <ul style="list-style-type: none"> Whose email address (registered with their JIRA account) matches the first matching address encountered in the To: , then Cc: and then Bcc: field of the email message received. Who also has the Assignable User project permission for the relevant Project (specified above).

CC Watchers	<p>Select this checkbox if you want JIRA to automatically add JIRA users to the issue created, where those users' email addresses (registered with their JIRA accounts) match addresses encountered in the To:, Cc: or Bcc: fields of the email message received.</p> <p> Please note that when an issue is created, new JIRA users created by the Create Users option (above) <i>cannot also be added</i> to the issue's watchers list by this CC Watchers option. JIRA users must <i>already</i> exist in JIRA's userbase, and must have an email address.</p>
--------------------	--

3. Test and save your mail handler (above).

Add a comment before a specified marker or separator in the email body

This message handler creates a comment from the body of an email message - but ignores any part of the body past a marker or separator that matches a specified regular expression (regex).

For mail systems like Lotus Notes and Outlook, the core content of an email message is separated from other (e.g. replied or forwarded) content in the body by some predictable text string like '---- Original Message ----' or 'Extranet\n email.address/DOM/REG/CONT/CORP@CORPMAIL'. Hence, use this message handler, which can take any valid regex, to filter core from extraneous content from various different mail systems.

Also note that the issue to which the comment is added is chosen from the first issue key found in the email subject.

The **Add a comment before a specified marker or separator in the email body** mail handler has the following behavior with respect to received email messages:

- If the regex pattern (specified in the mail handler) is found, the text in the email message body before the first regex pattern match is used for the comment and the remainder of the body is discarded.
- If the regex pattern (specified in the mail handler) is not found, the entire text in the email message body is used for the comment.
- If no regex pattern is specified in the mail handler, the entire text in the email message body is used for the comment.
- If the regex expression specified in the mail handler is erroneous, the entire text in the email message body is used for the comment.

To configure an 'Add a comment before a specified marker or separator in the email body' mail handler:

1. If you have not already done so, begin configuring your mail handler (above).
2. On the **Add a comment before a specified marker or separator in the email body** dialog box, complete the following fields/options:

Split Regex	<p>Specify a regular expression matching the text that separates the content of the email message mail body from other (replied or forwarded) content in the body.</p> <p>Please Note:</p> <ul style="list-style-type: none"> • The regex must begin and end with a delimiter character, typically '/'. • Commas are not allowed in a regex, as they are used to separate each mail handler field/option when they are integrated into a JIRA service and there is not (as yet) an escape syntax. <p>For example:</p> <pre>/----\s*Original Message\s*----/</pre> <p>or</p> <pre>/_____*/</pre>
--------------------	--

Catch Email Address	<p>If specified, only email messages whose To:, Cc:, Bcc: lines contain the recipient specified in this field will be processed — for example, issues@mycompany.com</p> <p>Upon specifying an address here, all email messages whose To:, Cc:, Bcc: lines contain addresses other than the Catch Email Address are ignored. This is useful if you have multiple aliases for the same mail account (e.g. foo-support@example-co.com and bar-support@example-co.com aliases for support@example-co.com) for multiple mail services (e.g. each one to create issues in separate JIRA projects).</p> <p>Note: In practice, this option is rarely useful and should not be confused with the more common Default Reporter. You can only specify one catch email address and one issue type per mail handler.</p>
Bulk	<p>This option only affects 'bulk' email messages whose header has either its Precedence field set to bulk or its Auto-Submitted field not set to no. Such messages would typically be sent by an automated service. When such an email message is received, the following action will be performed, based on the option you choose:</p> <ol style="list-style-type: none"> Ignore the email and do nothing. Forward the email (i.e. to the address set in the Forward Email text field). Delete the email permanently.
Forward Email	<p>If specified, then if this mail service is unable to handle an email message it receives, an email message indicating this problem will be forwarded to the email address specified in this field.Note: An SMTP mail server must be configured for this option to function correctly.</p>
Create Users	<p>Select this checkbox if you want JIRA to create new user accounts from any received email messages whose From: field contains an address that does not match one associated with an existing JIRA user account. This allows the creator of the email message to be notified of subsequent updates to the issue, which can be achieved by configuring the relevant project's notification scheme to notify the Reporter of updates.</p> <p>The username and email address of these newly created JIRA user accounts will be the email address specified in the From: field 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.</p> <p>Users created this way will be added to the default group/s of the default JIRA application (and therefore take up a license for this application). See the Managing groups documentation.Note: this option is not compatible with Default Reporter field option below and as such, choosing the Create Users option will hide the Default Reporter option.</p>
Default Reporter	<p>Specify the username of a default reporter, which will be used if the email address in the From: field of any received messages does not match the address associated with that of an existing JIRA user — for example, a JIRA username such as <input type="text" value="email-led-reporter"/></p> <p>Note:</p> <ul style="list-style-type: none"> This option is not available if the Create Users checkbox is selected. Please ensure that the user specified in this field has the Create Issues project permission for the relevant Project (specified above) as well as the Create Comments project permission for the other relevant projects to which this mail handler should add comments.
Notify Users	<p>Clear this check box if you do not want JIRA to send out an email message notifying users whose accounts have been created by the Create Users option above.</p> <p>Note: this option only functions if the Create Users check box has been selected.</p>

3. Test and save your mail handler ([above](#)).

Custom mail handlers

You can design your own message handlers to better integrate your own processes into JIRA. Such custom mail handlers configured using the standard procedure [above](#).

For more information about creating custom mail handlers, see the [Message Handler Plugin Module](#) documentation.

Issue/comment creation

The following points describe how JIRA processes each incoming email message and determines how its content gets added as either a comment to an existing issue or a new issue altogether.

- The **subject** of an email message is examined for an existing issue key:
 - If an issue key is found in the **subject**, the content of the email message's **body** is processed and added as a comment to the issue with that issue key.
 - If an issue key is NOT found in the **subject**, the **in-reply-to header** is examined:
 - If the email message is found to be a reply to another email message from which an issue was previously created, the **body** is processed and added as a comment to that issue.
 - If the email message is NOT found to be a reply, a new issue is created.

For example, an email message to a mail account `foo@example-co.com` on a POP or IMAP mail server configured against a JIRA server will be processed as follows:

- Issue Creation:
 - The **subject** of the email message will become the issue summary.
 - ⚠ Since all issues require a summary, each email message intended for issue creation should include a **subject**.
 - The **body** of the email message will be the issue description.
 - A bug will be created for project 'JIRA' with the above information. (This is essentially based on the mail handler configuration [above](#)).
 - Any attachments to the email message will become attachments to the issue (assuming [attachments](#) have been enabled in JIRA).
 - ℹ To ensure compatibility with various operating systems, any of the following characters in the filename will be replaced with an underscore character: \, /, ", %, :, \$, ?, *, <, |, >.
 - If the incoming email is set to a high priority, the corresponding issue will be created with a higher priority than the default priority that is set in your JIRA system.
- 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).

Handy tips with mail handlers

To allow JIRA to handle email messages sent from people without a JIRA user account:

1. Create an 'anonymous'/'dummy' mail account on your mail server/service ([above](#)).
2. Create an equivalent 'anonymous'/'dummy' JIRA user account, whose **Email** field matches the mail account you created in the previous step.
3. When configuring your mail handler(s) ([above](#)) to handle messages from this mail account, set the **Default Reporter** to this 'anonymous'/'dummy' JIRA user account.

Best practices (pre-processing JIRA email messages)

For JIRA production servers, we recommend that setting up the following email message pre-processing:

- Since JIRA mail handlers remove successfully processed email messages from your mail server, ensure that your mail is sent to a backup folder so that a record of what mail JIRA processed is available.
- If your mail folder contains replies to JIRA's email notifications, set up rules that filter out auto-replies and bounces.

If you do not do this, there is a strong possibility of mail loops between JIRA and autoresponders like 'out of office' notifications. JIRA sets a 'Precedence:bulk' header (unless you have 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 (e.g. > 50 MB given 512 MB heap). As the un-handled mail is not 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 are sure your mail server will not 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 does not have to process (and possibly create issues from) spam.

Troubleshooting

JIRA's **Logging & Profiling** page has configuration options for Outgoing and Incoming mail. Whenever you create a new (or edit an existing) mail handler (above), a **Test** button is available to allow you to test your mail handler's configuration to ensure it works as expected. 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 in `catalina.out` (or standard output).

Common problems

- If JIRA does not appear to be creating sending emails or creating issues and comments from email, your JIRA instance could be experiencing **OutOfMemory errors**. Please check your log files for OutOfMemory errors. If there are OutOfMemory errors, please restart JIRA and [investigate the errors](#).
- If you find some incoming emails simply disappear, check that you have not accidentally **started a second copy of JIRA** (e.g. in a staging environment) which is downloading and deleting mails. See [Diagnosable email sending/receiving](#) for flags you should set to prevent mail being processed.
- If replies by email of JIRA's notifications list JIRA's SMTP server rather than the configured handler POP account (ie, in Outlook's 'Reply-to' functionality), the project needs to be configured to add a 'reply-to' header in outgoing notifications. This can be configured in the project view for that particular project in JIRA's Administration.
- If HTML/Rich Text formatting is not being process correctly by JIRA, this is an expected behavior. The email comment handler was designed to do plain text conversion.

Configuring JIRA applications to receive email from a POP or IMAP mail server

To enable JIRA to [create comments and issues from email](#), you need to first configure JIRA to receive email from a POP or IMAP mail server as described below.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Add or edit a POP or IMAP mail server

1. Choose



> **System.**

2. Select **Mail > Incoming Mail** to open the Incoming Mail page.
3. Click either the **Configure new POP / IMAP mail server** button to define a new POP / IMAP mail server, or the **Edit** link at the right of an existing POP / IMAP mail server configuration, which will open the **Add/Update POP / IMAP Mail Server** page.
4. Complete the fields on this page as follows:

Name	Specify a short, arbitrary name to identify your POP or IMAP mail server configuration. You could possibly just specify the email address of the POP / IMAP mail server.
Description	(Optional) Specify an arbitrary description that describes the POP or IMAP mail server configuration and/or what it is used for. For example, 'Email Issue Creation/Comments for <Project>'. This description appears below the Name of the POP / IMAP mail server on the POP / IMAP Mail Servers configuration page.
Service Provider (not available when updating an existing POP / IMAP mail server)	Choose between using your own POP / IMAP mail server (i.e. Custom), Gmail POP / IMAP (i.e. Google Apps Mail / Gmail [POP3 / IMAP]) or Yahoo! POP (i.e. Yahoo! MailPlus) as the service provider for your POP / IMAP mail server. i If you choose any of the Gmail or Yahoo! options and then switch back to Custom , some of the key fields in this section will automatically be populated with the relevant POP / IMAP mail server settings for these service providers.
Protocol	Choose between whether your POP / IMAP mail server is a standard (i.e. POP or IMAP) or a secure (i.e. SECURE_POP or SECURE_IMAP) one. ! JIRA Cloud does not support self-signed certificates.
Host Name	Specify the hostname or IP address of your POP / IMAP mail server. Eg. <code>pop.yourcompany.com</code> or <code>imap.yourcompany.com</code>
POP / IMAP port	(Optional) The port to use to retrieve mail from your POP / IMAP account. Leave blank for default. Defaults are: POP: 110; SECURE_POP: 995; IMAP: 143; SECURE_IMAP: 993.

Timeout	(Optional) Specify the timeout period in milliseconds, which is treated as 10000 if this field is left blank. Specifying 0 or a negative value here will result in JIRA waiting indefinitely for the POP / IMAP server to respond.
Username	The username used to authenticate your POP / IMAP account.
Password	The password for your POP / IMAP account.  When editing an existing POP / IMAP mail server, select the Change Password checkbox to access and change this field.

5. (Optional) Click the **Test Connection** button to check that JIRA can communicate with the POP / IMAP mail server you just configured.
6. Click the **Add** (or **Update**) button to save the POP / IMAP mail server configuration.

Screenshot: Add/Update POP / IMAP Mail Server

Add POP / IMAP Mail Server

Use this page to add a new POP / IMAP server for JIRA to retrieve mail from.

Name *
The name of this server within JIRA.

Description

Service Provider

Protocol

Host Name *
The host name of your POP / IMAP server.

POP / IMAP Port
Optional - The port to use to retrieve mail from your POP / IMAP account. Leave blank for default. (defaults: POP - 110, SECURE_POP - 995, IMAP - 143, SECURE_IMAP - 993)

Timeout
Timeout in milliseconds - 0 or negative values indicate infinite timeout. Leave blank for default (10000 mSecs).

Username *
The username used to authenticate your POP / IMAP account.

Password *
The password for your POP / IMAP account.

POP / IMAP over SSL

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

Firstly, you will need to **import the mail server certificate** into a Java keystore. The process is described on the [Connecting to SSL Services](#) page.

 **Important Note:** Without importing the certificate, JIRA will not be able to communicate with your mail server.

JIRA system administration

This section of the documentation contains all the information you need to keep your JIRA instance healthy and running smoothly. If you can't find the information you need, you can also check the [JIRA Knowledge Base](#) and [Atlassian Answers](#) for help, and of course you can contact our legendary [Support](#) team and create an issue if you're stuck.

Search the topics in 'JIRA system administration':

- | | |
|------------------------------------|--|
| System administration | Learn more about your JIRA installation, like where to view your audit logs, information on JIRA search indexing, and where to find your Support Entitlement Number (SEN). |
| Configuring global settings | Learn more about your how to configure the settings which apply to all your users, and default settings for your JIRA installation. |
| Server optimization | Learn more about how to configure your JIRA installation to best suit your hardware and optimize performance. |

System administration

The following section of the documentation contains details your JIRA installation, like where to view the audit logs, how to find your Support Entitlement Number, and search indexing. It also contains details on backing up your instance, how to add and remove licenses, and important information on your files and directories.

- [Finding your Server ID](#)
- [Increasing JIRA application memory](#)
- [Using the database integrity checker](#)
- [Precompiling JSP pages](#)
- [Logging and profiling](#)
- [Backing up data](#)
- [Restoring data](#)
- [Search indexing](#)
- [Using robots.txt to hide from search engines](#)
- [Licensing your JIRA applications](#)
- [Viewing your system information](#)
- [Live monitoring using the JMX interface](#)
- [Monitoring database connection usage](#)
- [Viewing JIRA application instrumentation statistics](#)
- [Generating a thread dump](#)
- [Finding your JIRA application Support Entitlement Number \(SEN\)](#)
- [Auditing in JIRA applications](#)
- [Important directories and files](#)
- [Integrating JIRA applications with a Web server](#)
- [Securing JIRA applications with Apache HTTP Server](#)
- [Changing JIRA application TCP ports](#)
- [Connecting to SSL services](#)
- [Running JIRA applications over SSL or HTTPS](#)
- [Configuring security in the external environment](#)
- [Data collection policy](#)
- [JIRA Admin Helper](#)
- [Raising support requests as an administrator](#)
- [Start and Stop JIRA applications](#)
- [Managing LexoRank](#)

Finding your Server ID

The **Server ID** is an identifier for your JIRA server. When creating a JIRA license on my.atlassian.com, you may be prompted to enter the Server ID. You can locate your Server ID on the **System info** page.

Finding your your JIRA Server ID

1. Log in as a user with the **JIRA System Administrators** global permission.
2. Choose



> **System**. Select **System info** on the left menu to open the System info page.

- The **Server ID** is displayed in the **JIRA Info** section of the page.

JIRA setup wizard

If you are installing JIRA for the first time, you can locate your Server ID on the **Specify your license key** screen in the JIRA setup wizard. You'll see this page if you choose to perform a custom install, or if your server is not connected to the Internet.

Increasing JIRA application memory

Java applications like JIRA Software and Confluence run in a "Java virtual machine" (JVM), instead of directly within an operating system. When started, the Java virtual machine is allocated a certain amount of memory, which it makes available to JIRA applications. By default, Java virtual machines are allocated 64 MB of memory, no matter how many gigabytes of memory your server may actually have available. 64 MB is inadequate for medium to large JIRA application installations, and so this needs to be increased. Seeing [OutOfMemoryErrors in the logs](#) is symptomatic of this.

Note:

- This page addresses how to increase Heap Space memory. Confirm that you're not receiving [Perm Gen](#) or [GC Overhead](#) errors.
- Make sure you do not to exceed **1024 MB** as a base configuration when installing JIRA in Windows 32 bit.
- For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

On this page:

- [Step 1: Diagnosis](#)
- [Step 2: Increase available memory](#)
- [Step 3: Verify your settings](#)

Step 1: Diagnosis

- ▾ [Expand to see diagnosis section](#)

Assess root cause

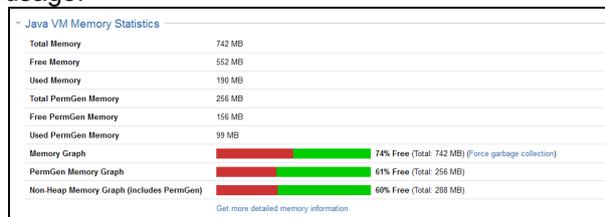
Often, there is a root cause for OutOfMemory Errors that may be better to address than just increasing memory. See [JIRA Crashes Due to 'OutOfMemoryError Java heap space'](#) for a discussion.

Determine JIRA application usage patterns

Choose



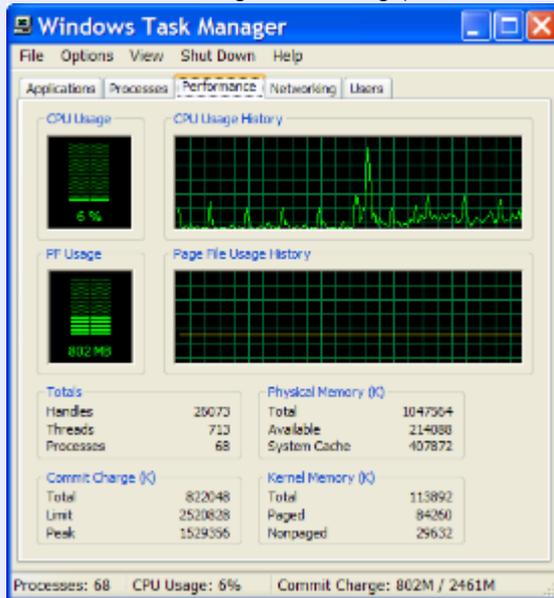
> **System.** Select **System support > System Info** to open the System Info page. Then, scroll down the page to view the Java VM Memory Statistics section, and look at the memory graph during times of peak usage:



i This server has been allocated a maximum of 768 MB and a minimum of 256 MB (typically defined in the `setenv` script which is executed by running the `start-jira` script). If you are trying to see whether your settings are being picked up by JIRA applications, this is where to look. Here, you can see that JIRA applications have reserved 742 MB, or which 190 MB is actually in use. If this JIRA application instance were running out of memory, it would have reserved the maximum available (768 MB), and would be using an amount close to this.

Determine available system memory**On Windows**

From the Close Programs Dialog (Press ctrl-alt-delete), select the Performance tab:



i The amount marked **Available** is the amount in kilobytes you have free to allocate to JIRA applications. On this server, we should allocate at most 214 MB.

On Linux

Run `cat /proc/meminfo` to view the memory usage.

Setting the `-Xmx` above the available amount on the server runs the risk of `OutOfMemoryErrors` due to lack of physical memory. If that occurs the system will use swap space, which greatly decreases performance.

Guidance

As a rule of thumb, if you have fewer than 5000 issues, JIRA applications should run well with the default 768 MB. Granting JIRA applications too much memory can impact performance negatively, so it is best to start with 768 MB, and make modest increases as necessary. As another data point, 40,000 works well with 768 MB to 1 GB.

Step 2: Increase available memory**Linux**

▼ Expand to see Linux instructions

To increase heap space memory in Linux installations:

1. In your `<JIRA application installation directory>/bin` (or `<Tomcat Installation Directory>/bin` for JIRA WAR installations), open the `setenv.sh` file.
2. Find the sections `JVM_MINIMUM_MEMORY=` and `JVM_MAXIMUM_MEMORY=`
3. See Diagnosis above and enter the appropriate values.

Windows (starting from .bat file)

▼ Expand to see Windows .bat file instructions

To configure system properties in Windows installations when starting from the .bat file:

1. In your <JIRA application installation directory>/bin (or <Tomcat Installation Directory>/bin for JIRA WAR installations), open the `setenv.bat` file.
2. Find the section `set JVM_MINIMUM_MEMORY=` and `set JVM_MAXIMUM_MEMORY=`
3. See [Diagnosis](#) above and enter the appropriate values.

Windows service

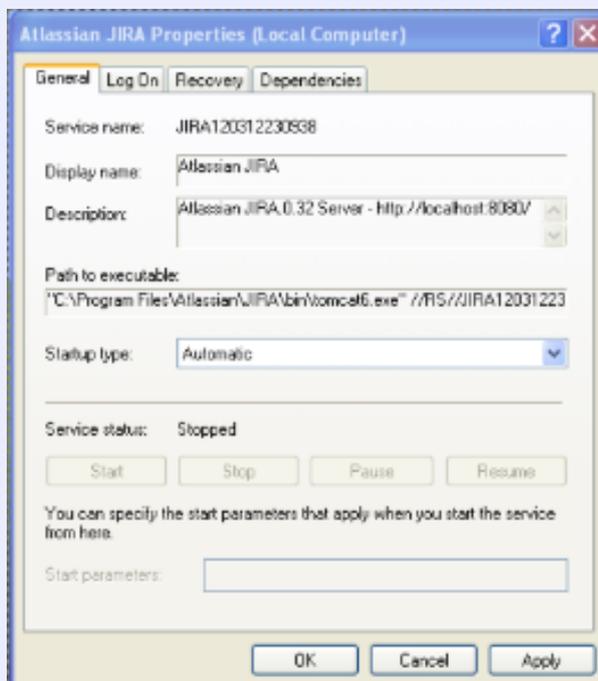
▼ [Expand to see Windows service instructions](#)

There are two ways to configure system properties when starting Running JIRA applications as a Windows service, either via command line or in the Windows registry.

Setting properties for Windows services via command line

To set properties for Windows services via command line

1. Identify the name of the service that JIRA applications are installed as in Windows (Control Panel > Administrative Tools > Services):



- i** In the above example, the **SERVICENAME** is: JIRA120312230938
2. Open the command window from Start > Run > type in 'cmd' > press 'Enter'
3. cd to the bin subdirectory of your JIRA application installation directory (or the bin sub directory of your Tomcat installation directory if your are running the JIRA WAR distribution).
For Example:

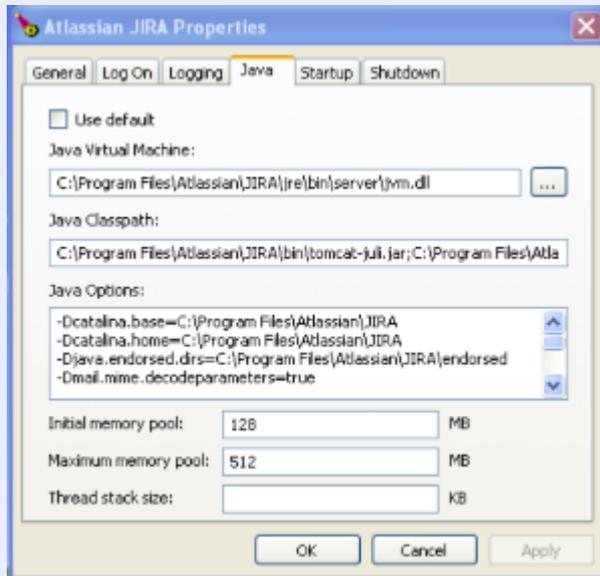
```
cd C:\Program Files\Atlassian\JIRA\bin
```

4. Run the following command:

```
tomcat8w //ES//%SERVICENAME%
```

For example: tomcat8w //ES//JIRA120312230938

5. Click on the Java tab to see the list of current start-up options:



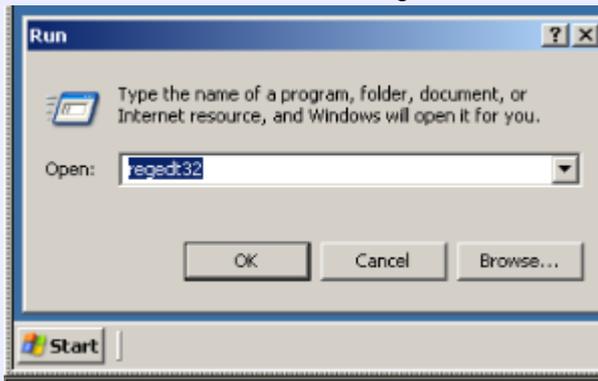
6. Set the maximum memory allocation here

Setting properties for Windows services via the Windows registry

In some versions of Windows, there is no option to add Java variables to the service. In these cases, you must add the properties by viewing the option list in the registry.

To set properties for Windows services via the Windows registry

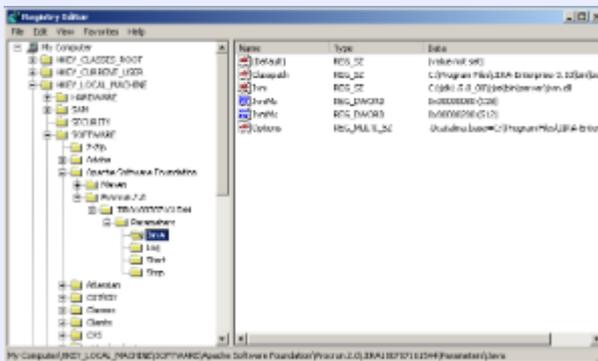
1. Go to Start > Run, and run "regedit32.exe".



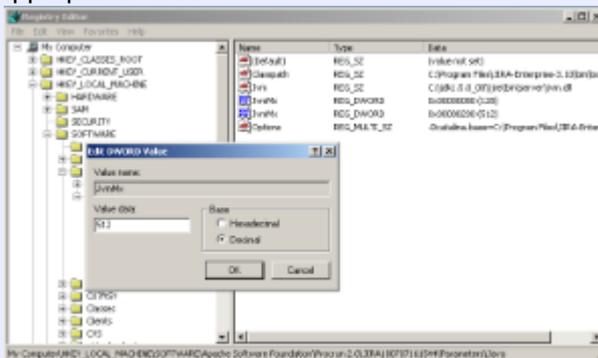
2. Find the Services entry:

32-bit: HKEY_LOCAL_MACHINE > SOFTWARE > Apache Software Foundation > Procrun 2.0 > JIRA

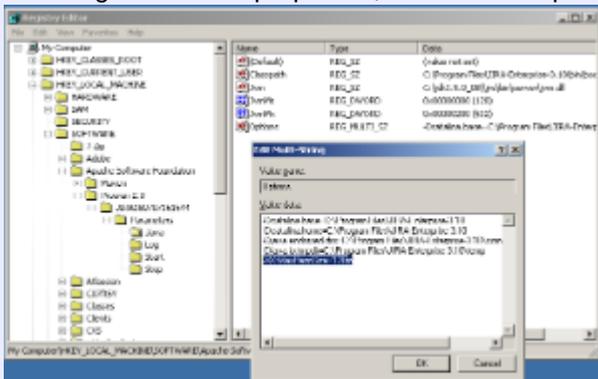
64-bit: HKEY_LOCAL_MACHINE > SOFTWARE > Wow6432Node > Apache Software Foundation > Procrun 2.0 > JIRA



3. To change existing properties, especially increasing Xmx memory, double-click the appropriate value.



4. To change additional properties, double-click options.



5. Modify the memory allocations here.

Step 3: Verify your settings

▼ Expand to see verification instructions

To verify what settings are in place, check the <JIRA application home directory>/logs/atlassian-jira.log or catalina.out file. A section in the startup appears like this:

```
JVM Input Arguments :
-Djava.util.logging.config.file=/usr/local/jira/conf/logging.properties -XX:MaxPermSize=256m -Xms256m -Xmx384m
-Djava.awt.headless=true -Datlassian.standalone=JIRA
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true
-Dmail.mime.decodeparameters=true
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djava.endorsed.dirs=/usr/local/jira/endorsed
-Dcatalina.base=/usr/local/jira -Dcatalina.home=/usr/local/jira
-Djava.io.tmpdir=/usr/local/jira/temp
```

 Look for Xmx (maximum) and Xms (minimum) settings.

This display is also available by [viewing your system information](#).

Using the database integrity checker

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.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Using the Integrity Checker

1. Choose



> **System**.

2. Select **System support > Integrity checker** to open the Integrity Checker page.

The integrity checker has a number of 'integrity checks' that look for common inconsistencies in JIRA's stored data.

Integrity Checker

Select one or more integrity checks from the list below to check for out of date information in the database.

Select All

Check Issue Relations

- Check Issue for Relation 'ParentProject'
- Check Issue for Relation 'RelatedOSWorkflowEntry'
- Check that all Issue Links are associated with valid issues

Check Search Request

- Check search request references a valid project

Check for Duplicate Permissions

- Check the permissions are not duplicated

Check Workflow Integrity

- Check workflow entry states are correct
- Check workflow current step entries
- Check JIRA issues with null status

Check Field Layout Scheme Integrity

- Check field layout schemes for references to deleted custom fields

Check for invalid filter subscriptions

- Check FilterSubscriptions for references to non-existent QuartzTriggers
- Check FilterSubscriptions for references to non-existent SearchRequests
- Check for existence of SimpleTriggers

3. Select one or more items whose data you would like to check the integrity of and click the **'Check'** button.
4. 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.

i Please Note: We **strongly** recommend taking a [backup](#) of your data before correcting any data inconsistencies.

5. 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.

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>

```

Logging and profiling

Logging

JIRA uses a powerful logging module called `log4j` for runtime logging.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Log file location

The logs are written to the `log` subdirectory of your **JIRA application home directory** (or elsewhere if you have configured a different location). You can view the location of the `atlassian-jira.log` in the **'File Paths'** section of the [system information](#) page.

- Security-related information (e.g. login, logout, session creation/destruction, security denials) is written to `atlassian-jira-security.log`.

Changing the location of the log

In the `log4j.properties` file (located in the **JIRA application installation directory**):

1. Change the following line:

```
log4j.appender.filelog=com.atlassian.jira.logging.JiraHomeAppender
```

...to this:

```
log4j.appender.filelog=org.apache.log4j.RollingFileAppender
```

2. Change the following line to point to the new location of the log file:

On this page:

- [Logging](#)
- [Profiling](#)

```
log4j.appender.filelog.File=atlassian-jira.log
```

Logging levels

There are five logging levels available in log4j: 'DEBUG', 'INFO', 'WARN', 'ERROR' and 'FATAL'. Each logging level provides more logging information than the level before it:

- 'DEBUG'
- 'INFO'
- 'WARN'
- 'ERROR'
- 'FATAL'

'DEBUG' provides the most verbose logging and 'FATAL' provides the least verbose logging. The default level is WARN, meaning warnings and errors are displayed. Sometimes it is useful to adjust this level to see more detail.

⚠ Please be aware: the 'DEBUG' setting may cause user passwords to be logged.

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. If you are unsure of which logging categories to adjust, the most helpful information generally comes from the `log4j.rootLogger` category and the `log4j<category>.com.atlassian` categories.

Temporarily changing the logging level

1. Choose



> **System.**

2. Select **System support > Logging & Profiling** to open the Logging page, which lists all defined log4j categories (as package names) and their current logging levels.
3. To change logging level of a category, click linked logging level associated with the relevant package name. To turn off logging of a category, click the '**OFF**' link associated with the relevant package name.

Permanently changing the logging level

1. Edit the `log4j.properties` file (located in the JIRA application installation directory).
2. Locate the section:

```
log4j.logger.com.atlassian = WARN, console, filelog
log4j.additivity.com.atlassian = false
```

and make your desired changes (e.g. change the WARN to DEBUG).

i 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.

3. Restart JIRA.

i Please note: If your application server configures logging itself, you may need to remove the `log4j.properties` file. You may also need to remove the entire `log4j.jar` file to get logging to work.

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 analyze 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.

Temporarily enabling profiling

1. Choose



> **System.**

2. Select **System support > Logging & Profiling** to open the Logging page, which lists all defined log4j categories (as package names) and their current logging levels.
3. Scroll to the '**Profiling**' section at the end of the page. This section will inform you whether profiling is currently turned 'ON' or 'OFF' and will provide you with 'Disable' or '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.

Permanently enabling profiling

1. In your JIRA installation directory, edit the `atlassian-jira/WEB-INF/web.xml` file.
2. Find the following entry:

```

<filter>
    <filter-name>profiling</filter-name>

    <filter-class>com.atlassian.jira.web.filters.JIRAProfilingFilter
</filter-class>
    <init-param>
        <!-- specify the which HTTP parameter to use to
turn the filter on or off -->
        <!-- 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. Profiling will be enabled when you restart JIRA.

Logging email protocol details

To assist in resolving email issues, it can be useful to know exactly what is passing over the wire between JIRA and SMTP, POP or IMAP servers. This page describes how to enable protocol-level logging.

To do this

Set **-Dmail.debug=true** and restart JIRA. Refer to [Setting properties and options on startup](#) for details on how to do this.

Output

In the logs, you should then see JavaMail initialize the first time a mail operation is run:

```

DEBUG: JavaMail version 1.3.2
DEBUG: java.io.FileNotFoundException:
/usr/local/jdk1.6.0/jre/lib/javamail.providers (No such file or
directory)
DEBUG: !anyLoaded
DEBUG: not loading resource: /META-INF/javamail.providers

```

```

DEBUG: successfully loaded resource:
/META-INF/javamail.default.providers
DEBUG: Tables of loaded providers
DEBUG: Providers Listed By Class Name:
{com.sun.mail.smtp.SMTPSSLTransport=javax.mail.Provider[TRANSPORT,smtps,
com.sun.mail.smtp.SMTPSSLTransport,Sun Microsystems, Inc],
com.sun.mail.smtp.SMTPTransport=javax.mail.Provider[TRANSPORT,smtp,com.s
un.mail.smtp.SMTPTransport,Sun Microsystems, Inc],
com.sun.mail.imap.IMAPSSLStore=javax.mail.Provider[STORE,imaps,com.sun.m
ail.imap.IMAPSSLStore,Sun Microsystems, Inc],
com.sun.mail.pop3.POP3SSLStore=javax.mail.Provider[STORE,pop3s,com.sun.m
ail.pop3.POP3SSLStore,Sun Microsystems, Inc],
com.sun.mail.imap.IMAPStore=javax.mail.Provider[STORE,imap,com.sun.mail.
imap.IMAPStore,Sun Microsystems, Inc],
com.sun.mail.pop3.POP3Store=javax.mail.Provider[STORE,pop3,com.sun.mail.
pop3.POP3Store,Sun Microsystems, Inc]}
DEBUG: Providers Listed By Protocol:
{imaps=javax.mail.Provider[STORE,imaps,com.sun.mail.imap.IMAPSSLStore,Su
n Microsystems, Inc],
imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun
Microsystems, Inc],
smtps=javax.mail.Provider[TRANSPORT,smtps,com.sun.mail.smtp.SMTPSSLTrans
port,Sun Microsystems, Inc],
pop3=javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun
Microsystems, Inc],
pop3s=javax.mail.Provider[STORE,pop3s,com.sun.mail.pop3.POP3SSLStore,Sun
Microsystems, Inc],
smtp=javax.mail.Provider[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,
Sun Microsystems, Inc]}
DEBUG: successfully loaded resource:
/META-INF/javamail.default.address.map
DEBUG: !anyLoaded
DEBUG: not loading resource: /META-INF/javamail.address.map
DEBUG: java.io.FileNotFoundException:
/usr/local/jdk1.6.0/jre/lib/javamail.address.map (No such file or
directory)
DEBUG: getProvider() returning
javax.mail.Provider[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun
Microsystems, Inc]
DEBUG POP3: connecting to host "localhost", port 110, isSSL false
S: +OK Dovecot ready.
C: USER pop-test
S: +OK
C: PASS pop-test
[Filter: profiling] Using parameter [jira_profile]
[Filter: profiling] defaulting to off [autostart=false]
[Filter: profiling] Turning filter off [jira_profile=off]
S: +OK Logged in.
C: STAT
S: +OK 2 1339
C: NOOP
S: +OK
C: TOP 1 0
S: +OK
Return-path: <pop-test@atlassian.com>
Envelope-to: pop-test@localhost
Delivery-date: Wed, 28 Feb 2007 16:28:26 +1100

```

```
Received: from pop-test by teacup.atlassian.com with local (Exim 4.63)
  (envelope-from <pop-test@atlassian.com>)
  id 1HMHMY-0007gB-80
  for pop-test@localhost; Wed, 28 Feb 2007 16:28:26 +1100
Date: Wed, 28 Feb 2007 16:28:26 +1100
From: Jeff Turner <jeff@atlassian.com>
To: pop-test@localhost
Subject: Testing to me - Wed Feb 28 16:28:23 EST 2007
Message-ID: <20070228052826.GA29514@atlassian.com>
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Disposition: inline
```

```
User-Agent: Mutt/1.5.13 (2006-08-11)
Lines: 0
```

Related pages

- [Logging and profiling](#)

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 data](#) and [Restoring a project from backup](#).

Creating a complete backup of JIRA consists of two stages:

- 1. Backing up database contents
 - Using native database backup tools
 - Using JIRA's XML backup utility
- 2. Backing up the data directory

1. Backing up database contents

There are two possibilities: native database backup tools, or JIRA's XML backup utility.

For production use, it is **strongly recommended** that for regular backups, you use native database backup tools instead of JIRA's XML backup service.

When JIRA is in use, XML backups are not guaranteed to be consistent as the database may be updated during the backup process. JIRA does not report any warnings or error messages when an XML backup is generated with inconsistencies and such XML backups will fail during the restore process. Native database backup tools offer a much more consistent and reliable means of storing (and restoring) data while JIRA is active.

Caveat: if you are migrating your instance, we recommend that you create an XML backup (per the directions in this guide) where possible. In certain cases, such as very large instance sizes, this may not be possible due to the system requirements for an XML backup.

Using native database backup 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](#).

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Using JIRA's XML backup utility

To perform a once-off backup, e.g. before an upgrade, follow the steps below.

 You can also configure scheduled XML backups, as described in [Automating JIRA application backups](#).

1. Choose



> **System.**

2. Select **Import & Export > Backup System** to open the Backup JIRA data page.

Screenshot: The Backup JIRA Data Page

Backup JIRA data ?

This will backup the contents of the database in a portable XML format.

You can use this backup to move JIRA between different databases if required, as well as creating a backup that you can use if something goes wrong. To backup to a file on the server, enter the filename below.

The backup file will be placed here: `C:\jira\home\export`

⚠ Attachments will not be backed up. This needs to be done manually.
XML generation is complex so there might be a delay before it completes!

File name

- i** As shown in the screenshot above, the backup will be stored within the `export` subdirectory of the JIRA application home directory.
- 3. In **'File name'** field, type the name of the backup file.
- i** Ensure that JIRA has the necessary file system permissions to write to this location.
- 4. Click the **'Backup'** button and wait while your JIRA data is backed up.
- i** JIRA will save your XML backup as a zipped archive file.
- 5. When the backup is complete, a message will be displayed, confirming that JIRA has written its data to the file you specified.

2. Backing up the data directory

It's crucial that you back up your JIRA application's `data` directory, which is a sub-directory of the [JIRA application home directory](#) (`jira-home` for short). The `data` directory contains application data for your JIRA instance. For example, issue attachments are stored in the `<jira-home>\data\attachments` directory.

Backing up the JIRA index

The JIRA index is stored in a different sub-directory, `<jira-home>\caches`. On large instances, we recommend that you enable 'restorable index' in the system options to create backups of the index that you can restore later.

There's no one specific way to back up the data directory, but here are a couple of methods you might consider:

- On MS Windows, a batch script copying the directory can be written and scheduled periodically (Programs > Accessories > System Tools > Scheduled Tasks).
- On Linux/Solaris, you can 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.

If you have put your `attachments` directory in a custom location rather than inside the `data` directory, you'll need to back up your `attachments` directory separately.

If you have a large file, refer to the following guide on [How to Transfer Large Files to Atlassian](#).

Automating JIRA application backups

JIRA applications can be configured to automatically create an XML backup of JIRA application data on a routine basis.

i Please note:

- The XML backup includes all data in the database. However, it **does not include** your `attachments` direct

ory, JIRA application home directory, or JIRA application installation directory, which are stored on the filesystem.

- You can also perform XML backups manually. See [Backing up data](#) for details.
- Be aware that after [installing JIRA applications](#) and running the [setup wizard](#), a backup service will automatically be configured to run every 12 hours.

For production use or large JIRA application installations, it is **strongly recommended** that you use [native database-specific tools](#) instead of the XML backup service. XML backups are not guaranteed to be consistent, as the database may be updated during the backup process. Inconsistent backups are created successfully without any warnings or error messages, but fail during the restore process. Database-native tools offer a much more consistent and reliable means of storing data.

To configure automated JIRA application backups:

1. Log in as a user with the [JIRA System Administrators global permission](#).
2. Select **Administration > System > Advanced > Services** (tab) to open the **Services** page, which lists the current services running on this system. By default, there should be at least one 'Mail Queue Service' running, which cannot be deleted.

Services ?

Name / Class	Properties	Delay (mins)	
Service Provider Session Remover <small>com.atlassian.sal.jira.scheduling.JiraPluginSchedulerService</small>	<ul style="list-style-type: none"> • pluginJobName: Service Provider Session Remover • repeatInterval: 28800000 • initiallyFired: true 	480	Edit Delete
com.atlassian.jira.plugin.ext.bamboo.service.PlanStatusUpdateServiceImpl:job <small>com.atlassian.sal.jira.scheduling.JiraPluginSchedulerService</small>	<ul style="list-style-type: none"> • pluginJobName: com.atlassian.jira.plugin.ext.bamboo.service.PlanStatusUpdateServiceImpl:job • repeatInterval: 60000 • initiallyFired: true 	1	Edit Delete
com.atlassian.streams.internal.ActivityProviderConnectionMonitorImpl:activityProviderMonitor <small>com.atlassian.sal.jira.scheduling.JiraPluginSchedulerService</small>	<ul style="list-style-type: none"> • pluginJobName: com.atlassian.streams.internal.ActivityProviderConnectionMonitorImpl:activityProviderMonitor • repeatInterval: 300000 • initiallyFired: true 	5	Edit Delete
NotificationCacheUpdateJob:job <small>com.atlassian.sal.jira.scheduling.JiraPluginSchedulerService</small>	<ul style="list-style-type: none"> • pluginJobName: NotificationCacheUpdateJob:job • repeatInterval: 86400000 • initiallyFired: true 	1440	Edit Delete
Backup Service <small>com.atlassian.jira.service.services.export.ExportService</small>	<ul style="list-style-type: none"> • USE_DEFAULT_DIRECTORY: true 	720	Edit Delete
PluginLicenseExpiryJob:job <small>com.atlassian.sal.jira.scheduling.JiraPluginSchedulerService</small>	<ul style="list-style-type: none"> • pluginJobName: PluginLicenseExpiryJob:job • repeatInterval: 86400000 • initiallyFired: true 	1440	Edit Delete
Mail Queue Service <small>com.atlassian.jira.service.services.mail.MailQueueService</small>		1	Edit

Add Service ?

Add a new service by entering a name and class below. You can then edit it to set properties. Mail handlers should be added or edited via our new [Incoming Mail](#) section.

Name

Class

Built-in Services

- Debugging service
- Run Jelly script
- Backup service

Delay

Delay between running time, in minutes.

3. In the **Add Service** form towards the end of the page, complete the following fields:
 - **Name** — a descriptive name for the backup service, such as `Backup Service`.
 - **Class** — the appropriate fully-qualified class name for the **Backup service** using either of the following methods:
 - a. Select the **Backup service** from the list of JIRA application **Built-in Services**. To do this:
 - i. Click the **Built-in Services** link below the **Class** field to expand the list of JIRA application built-in service classes.
 - ii. Click the **Backup service** link. The **Class** field will automatically be populated with the following class text string `'com.atlassian.jira.service.services.export.ExportService'`
 - b. Type the fully-qualified class name `'com.atlassian.jira.service.services.export.ExportService'` into the **Class** field.
 - **Delay** — enter the number of minutes between backups. A good default for this would be 720 minutes (12 hours) or 1440 minutes (24 hours).

Please Note: The interval specified in the Backup Service Delay (mins) is the time *when the next backup job will run since the last server restart*. Backup services cannot be scheduled to run at a specific time of day - please see [JRA-1865](#) for more on this.

4. Click the **Add Service** button. The **Edit Service** page is displayed.

Edit Service: Backup Service (Long)

Enter text values for service properties below. Any empty fields will be set to NULL in the Service's initialisation.

Use Default Directory Default directory is [jira.home]/export/

Date format Optional simple date format.

Delay Delay - in minutes
You can also adjust the delay period of this service. Note that if you adjust this delay, the service will be restarted.

5. Complete the following items on this page:
- For the **Date format** field, specify the format which JIRA applications 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 the **Delay** field, modify the number of minutes between backups if necessary.
 - If the **Use Default Directory** checkbox is displayed, see the [note](#) below.
6. Click the **Update** button. Your backup service is now configured. XML backups will be performed according to the schedule you specified in the **Delay** field.
- For every successful backup, a zipped file of your XML backup will be saved in the backup directory.
 - If a scheduled backup fails for any reason, the zipped XML backup file will be saved into the 'corrupted' directory, which is directly under your nominated backup directory. 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'.
-  JIRA applications will create the 'corrupted' directory if required - you do not need to create it.

About custom backup directories

The **Use Default Directory** checkbox (not shown in screenshot above) is for legacy JIRA application installations (prior to JIRA 4.2), which have backup services that use custom directories.

If you are using JIRA 5.1.0 or earlier, the **Use Default Directory** will always be displayed, as the option of using custom directories has been deprecated. If you are using JIRA 5.1.1 or later, the **Use Default Directory** checkbox will only be displayed if you upgraded from a version prior to 4.2 and you are editing an existing backup service which used a custom directory.

- If you are not using a legacy backup service with a custom directory, select the the **Use Default Directory** checkbox. If you do not, *your backup service may not work correctly*.
- If you are using a legacy backup service with a custom directory, you can choose between using the default directory or your custom directory (cannot be edited). Note, if you choose the default directory option, you will not be able to choose the custom directory option.

The default directory location is the `export` subdirectory of the [JIRA application home directory](#).

Preventing users from accessing JIRA applications during backups

For production use, it is **strongly recommended** that for regular backups, you use [native database backup tools](#) instead of the JIRA application XML backup service.

When JIRA applications are in use, XML backups are not guaranteed to be consistent as the database may be updated during the backup process. JIRA applications do not report any warnings or error messages when an XML backup is generated with inconsistencies and such XML backups will fail during the restore process. Native database backup tools offer a much more consistent and reliable means of storing (and restoring) data.

If you perform an XML backup (e.g. when upgrading JIRA applications via a test environment or migrating JIRA

applications to another server), you can follow one of these methods to prevent users from accessing JIRA applications and minimize inconsistencies in the backup file:

- **Recommended method:**

- If you have an Apache or other web/proxy server sitting in front of JIRA applications, then you can stop Apache from proxying to JIRA applications, and serve a static HTML page with a nice message along the lines of "JIRA applications are undergoing maintenance". Note:
 - The administrator must be able to access JIRA applications directly (not through Apache) to perform the XML backup.
 - This method does not require JIRA applications to be restarted.

- **Alternative method 1:**

1. Shut down all JIRA applications, configure them to listen on a different port and restart. Do this by editing the `server.xml` file. Change the following section:

```
<Connector port="8080"
           maxHttpHeaderSize="8192" maxThreads="150"
           minSpareThreads="25" maxSpareThreads="75"
           useBodyEncodingForURI="true"
           enableLookups="false" redirectPort="8443"
           acceptCount="100" connectionTimeout="20000"
           disableUploadTimeout="true" />
```

- Note: If you have enabled HTTPS, then you would need to edit the `HTTPS Connector` section as well.
2. Restart all JIRA applications and do the XML backup.
 3. Shut down all JIRA applications, change all the settings back, then re-start the applications.
- **Alternative method 2:**
 - If you have a firewall in front of your JIRA applications, you could stop requests from getting through or change the port number that it uses. Note:
 - The administrator will need to log into your JIRA applications on the temporary port number (or access it from behind the firewall), to perform the XML backup.
 - This method does not require JIRA applications to be restarted.

Before you start:

Whichever method you choose, we recommend setting an [announcement banner](#) to warn your users that JIRA applications will be unavailable for a period of time.

Restoring data

This process is typically conducted towards the end of [migrating JIRA applications to another server](#) or [splitting JIRA applications 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.

Restoring JIRA from backup is a three stage process:

1. (Optional) [Disable email sending/receiving](#)
2. [Restore data from XML to the database](#)
3. (Optional) [Restore the attachments to the attachments directory](#) (if attachments were backed up)

Restoring a project from backup

This page describes how to restore a single project from a backup file into your JIRA instance. This also includes instructions on how to migrate a project from JIRA Cloud to JIRA Server.

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.

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.

On this page:

- [Before you begin](#)
- [Restoring a project from JIRA Cloud to JIRA Server](#)
- [Restoring your project](#)

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.

We strongly recommend that you perform a [full backup](#) of your target JIRA instance before attempting to restore a project into it.

! Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

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 6.4, then your target instance of JIRA must be version 6.4.
- If your instance of JIRA had any [custom field plugins](#) 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 plugins installed for the Project Import tool to automatically work.

In JIRA 7.0, the Project Import functionality between identical instances of JIRA **supports some Active Objects data**. For example:

- Data that will be imported: JIRA Software's sprint data and ranking data
- Data that will *not* be imported: JIRA Software's board configuration data and Service Desk customer portals

For more information about extending the Project Import functionality, see [Guide - Extending the JIRA Import plugin](#).

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. (This will completely overwrite the test instance.)

3. [Create a new backup file](#) from your test JIRA instance. You can now use this backup to import a specific project into your target production 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.

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.

Restoring a project from JIRA Cloud to JIRA Server

You cannot import a project directly from JIRA Cloud to JIRA Server — the importer will display errors about version mismatches. If you want to restore a project from JIRA Cloud to JIRA Server, follow the steps below:

1. Install a new JIRA instance (in addition to the one that you want to import your project into). This will be a temporary instance that is used to store a full JIRA import from JIRA Cloud. Ensure that the version of this temporary instance matches the version of the JIRA instance that you want to import your project into, e.g. JIRA 6.2.
2. Do a full JIRA migration from JIRA Cloud to the temporary JIRA instance. See [Migrating from JIRA Cloud to JIRA Server applications](#).
3. Export the desired project from the temporary JIRA instance.
4. Import the project into your desired JIRA instance, by following the instructions in the [Restoring your project](#) section below.
5. (optional) Delete the temporary JIRA instance, once the project has completed.

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.

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 — perform a search 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 if 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.

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.

See [Adding a custom field](#) for more information on 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, system fields, groups and 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:

- The project import process does not import workflows or workflow schemes. If you wish to retain a customized 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.
- When importing a project, the **Create Issue** transition and the **Issue Created** event will be triggered for each issue along with all corresponding [postfunctions](#). If you change the **Create Issue** transition after the import (for example, setting one of the create issue field values), you may get values that are different from those in the source. As a workaround, you can manually disable postfunctions during the import.
- Read more about creating and editing workflows in the [Working with workflows](#) and [Managing your 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.

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 than 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. You may want to consider [setting 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 with the 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.

Make sure to match the **Linked Status** name, not the **Step Name**, when inspecting your workflow.

Security Levels:

- 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.

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](#) — Not applicable. Issue type screen schemes do not need to match between the backup and target instance of JIRA.
- [Field configuration schemes](#) — 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

While the Project Import tool preserves the existing issue keys from your backed up project during the import process, the tool will also 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

Project Import: Select Backup File ?

i This tool allows you to import a single JIRA project from a backup file. Importing a project into JIRA is a complex operation. It requires that you carry out manual modifications to the configuration of your JIRA instance. These modifications require that you have good understanding of, and experience in, JIRA administration and configuration.

! It is critical that you read our [Project Import documentation](#) and plan how to carry out the project import based on the information in that document. We strongly recommend that you first carry out the project import on a test JIRA instance, and then only carry it out on your production instance once you are sure that the test import was successful.

Please note that the backup file containing the project that you want to import must be from exactly the same version (6.0.3) of JIRA as this one.

While configuring a project import, JIRA will remain available to all users. Please be aware that once the data is actually being imported JIRA will be unavailable until the import has completed.

Please [backup](#) this JIRA instance before you begin the project import. The backup file and attachment paths must be located on the same machine as your JIRA instance.

File name*

Enter filename to restore project data from. Files will be loaded from : /home/rpillai/atlassian/application-data/jiralive/import

Backup Attachment Path

Path Path to the directory holding backed up attachments.

To start the Project Import tool:

1. Choose



> **System.**

2. Select **Import & Export > Project Import** to open the Project Import wizard page.
3. 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).
4. Copy the attachments from the path where you have backed up the attachments to the '**Backup Attachment Path**' shown in the import window. This path is under the JIRA home directory of the instance. Please note that if file attachments are not enabled in your target JIRA instance you will not see the path to which you need to copy the attachments from the backup.

Note: You can choose to not copy the attachments to the '**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 Import: Select Project to Import ?

The list of projects contains all projects present in the XML backup provided. Select the project you wish to import. The importer will attempt to automatically map the backup project's values to correct values in this instance of JIRA. Please make certain you have correctly configured the JIRA project in this instance (i.e. associated the correct schemes with the project, created any missing issue types, custom fields, etc.) See the documentation for full details of what needs to be done.

If a backup project can not be imported the details will be displayed below.

Projects from Backup

! No project with key 'DEMO' exists in this instance of JIRA. The importer will create a project with this key and the details of the backup project using the default schemes.

Project:	Demonstration Project
Key:	DEMO
Description:	<h3>Welcome to the administration of your demonstration project!</h3> <p>This is where you can view and change how the project is configured. Use the tabs on the left to navigate to different project settings.</p>
Lead:	admin
URL:	
Sender Address:	
Default Assignee:	Project Lead
Issues:	6
Components:	0
Versions:	0

1. Select a project to restore from the **'Projects from Backup'** drop-down. This drop-down 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

Project Import: Pre-Import Summary - Demonstration Project ?

The results of automatic mapping are displayed below. You will not be able to continue if any validation errors were raised.

- [Refresh validations](#) - re-maps and validates the backup data against the current state of JIRA.

Errors

- The data mappings have produced errors, you can not import this project until all errors have been resolved. See below for details.

System Fields	Custom Fields																				
<p>✔ Issue Type</p> <p>✘ Custom Field Configuration</p> <p>The custom field 'Ashamedly Khoikhoi's' of type 'Date Time' is required for the import but does not exist in the current JIRA instance.</p> <p>The custom field 'Calisthenics's' of type 'Number Field' is required for the import but does not exist in the current JIRA instance.</p> <p>The custom field 'Demeter's adverbs' of type 'Free Text Field (unlimited text)' is required for the import but does not exist in the current JIRA instance.</p> <p>The custom field 'Heartburn firepower Okhotsk's' of type 'Multi Select' is required for the import but does not exist in the current JIRA instance.</p> <p>The custom field 'Pedalled' of type 'Free Text Field (unlimited text)' is required for the import but does not exist in the current JIRA instance.</p>	<table style="width: 100%; border-collapse: collapse;"> <tr><td>Demeter's adverbs</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Pedalled</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Ashamedly Khoikhoi's</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Laciest</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Heartburn firepower Okhotsk's</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Calisthenics's</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Centralized</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Eugenie</td><td style="text-align: right;">Not checked yet</td></tr> </table>	Demeter's adverbs	Not checked yet	Pedalled	Not checked yet	Ashamedly Khoikhoi's	Not checked yet	Laciest	Not checked yet	Heartburn firepower Okhotsk's	Not checked yet	Calisthenics's	Not checked yet	Centralized	Not checked yet	Eugenie	Not checked yet				
Demeter's adverbs	Not checked yet																				
Pedalled	Not checked yet																				
Ashamedly Khoikhoi's	Not checked yet																				
Laciest	Not checked yet																				
Heartburn firepower Okhotsk's	Not checked yet																				
Calisthenics's	Not checked yet																				
Centralized	Not checked yet																				
Eugenie	Not checked yet																				
<table style="width: 100%; border-collapse: collapse;"> <tr><td>Status</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Priority</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Resolution</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Users</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Project Role</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Project Role Membership</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Group</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Issue Link Type</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Issue Security Level</td><td style="text-align: right;">Not checked yet</td></tr> <tr><td>Attachments</td><td style="text-align: right;">Not checked yet</td></tr> </table>	Status	Not checked yet	Priority	Not checked yet	Resolution	Not checked yet	Users	Not checked yet	Project Role	Not checked yet	Project Role Membership	Not checked yet	Group	Not checked yet	Issue Link Type	Not checked yet	Issue Security Level	Not checked yet	Attachments	Not checked yet	
Status	Not checked yet																				
Priority	Not checked yet																				
Resolution	Not checked yet																				
Users	Not checked yet																				
Project Role	Not checked yet																				
Project Role Membership	Not checked yet																				
Group	Not checked yet																				
Issue Link Type	Not checked yet																				
Issue Security Level	Not checked yet																				
Attachments	Not checked yet																				

Previous
Refresh Validations
Cancel

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 () means that there are no problems with mapping these entities.
 - An exclamation mark symbol () 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 () 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.

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 ?

The project import completed successfully in 0 minutes.

Project Summary		Users	
Key:	DEMO	No users were added during the import.	
Description:	<h3>Welcome to the administration of your demonstration project! </h3> <p>This is where you can view and change how the project is configured. Use the tabs on the left to navigate to different project settings.</p>	Project Roles	
Lead:	Admin	Administrators:	0 users, 1 groups
URL:		Developers:	0 users, 1 groups
Sender Address:		Users:	0 users, 1 groups
Default Assignee:	Project Lead	Issues	
Components:	0	Issues created:	6 out of 6
Versions:	0	No attachments were added during the import.	

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.

The Project Import tool will add an entry to every imported issue's Change History, showing when the issue was imported. Note that old entries in the Change History, from before the import, are retained for historical purposes only. Old entries may contain inconsistent data, since the configuration of the old and new JIRA systems may be different.

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 [anonymize the XML backups](#) if your data contains sensitive information.

Anonymising JIRA application data

Support requests are often resolved **significantly** faster if a data export is provided as it will allow our legendary supporters direct access to a copy of your instance. We understand that sometimes this may be a difficult option due to the sensitivity of your data and have written an anonymizing tool to handle this particular scenario.

Anonymizing JIRA Data

The JIRA inbuilt backup functionality will produce a ZIP file containing either 1 or 2 XML files, depending on the version that is being used. These files are a copy of the entire contents of JIRA's database, encoded in XML, that can be used to restore an instance - we have further detail on this in our [Automating JIRA application backups](#) documentation.

As of JIRA 4.4, the backup functionality will produce a ZIP file that contains 2 XML files. These files will be `activeobjects.xml` and `entities.xml`. Only `entities.xml` will need to be anonymized - please do not attempt to anonymize the `activeobjects.xml`. For versions prior to 4.4, only one XML file will be produced with the same naming convention as the ZIP it is compressed as (for example `1970-Jan-01-0001.zip` will expand to `1970-Jan-01--0001.xml`).

1. Ensure that the `JAVA_HOME` variable has been configured, as in our [Setting JAVA_HOME](#) documentation.
2. Download the [JIRA Anonymizer](#).
3. Create a temporary directory.
4. Unzip the anonymizer in the temporary directory.
5. Unzip the JIRA backup ZIP file (for example `1970-Jan-01--0001.zip`) in the temporary directory.
6. Anonymize the backup file with the below commands:

```
$ java -Xmx512m -jar joost.jar <JIRA BACKUP>.xml anon.stx > <NAME
OF ANONYMISED BACKUP>.xml
```

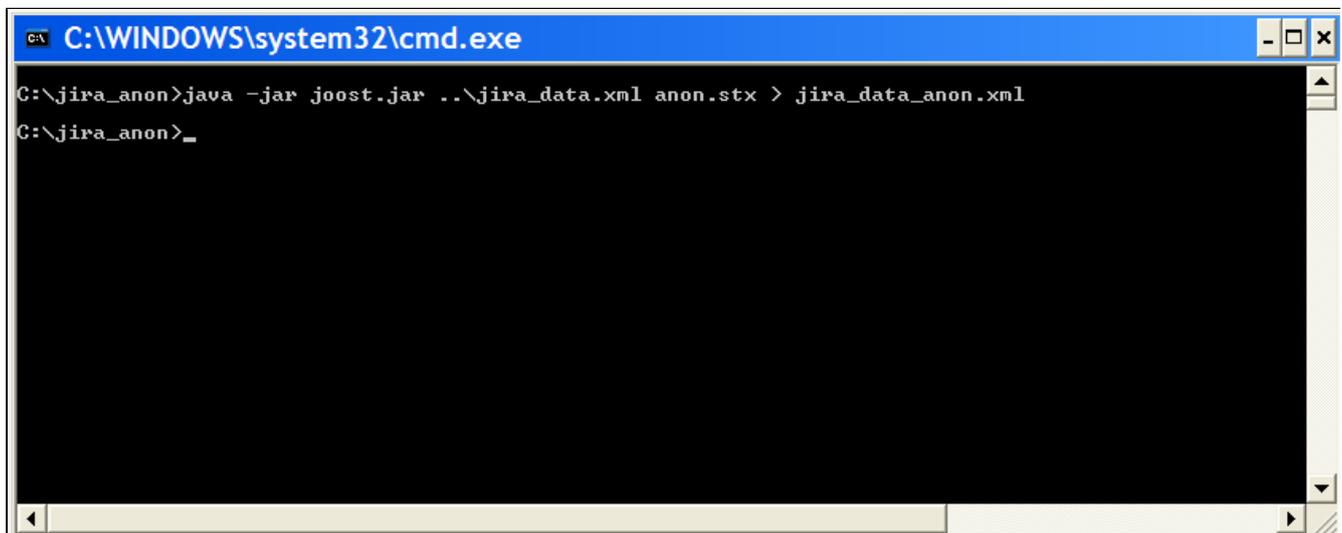
For example, this would be anonymizing a JIRA backup with the naming convention from JIRA 4.4+:

```
$ java -Xmx512m -jar joost.jar entities.xml anon.stx >
anon-entities.xml
```

 Depending on the size of the backup, additional memory may need to be allocated to the JVM. In order to do this, increase the value of the `Xmx` in increments of 128m.

7. Compress the generated anonymized XML backup file (e.g: `anon-entities.xml`) and the `activeobjects.xml` (*JIRA 4.4.x + only*) into a ZIP or tarball.
8. Attach that ZIP or tarball onto the support issues as raised on support.atlassian.com.
9. The temporary directory can now be removed.

The screenshot below is a simple example of how it is run in the command prompt of Windows XP:



```
C:\WINDOWS\system32\cmd.exe
C:\jira_anon>java -jar joost.jar ..\jira_data.xml anon.stx > jira_data_anon.xml
C:\jira_anon>_
```

Information about the Anonymizer

The anonymizer currently replaces the following text with x's:

- Issue summary, environment, and description.
- Comments, work logs, change logs.
- Project descriptions.
- Descriptions for most elements (notification schemes, permission schemes, resolutions).
- Attachment file names.
- "Unlimited text" custom fields.

Please check the anonymized backup, `anon-backup.xml`, to ensure it's clean enough for the needs of your organization before sending it to Atlassian.

Restoring data from an xml backup

Before you begin

Make sure that you have the password to a login in the backup file that has the JIRA System Administrator [global permission](#). Once the restoring procedure begins, all the existing data in the JIRA application database is deleted, including all user accounts.

If you are restoring data from a JIRA Cloud application site to a JIRA Server application, please read [Migrating from JIRA Cloud to JIRA Server applications](#).

For all of the following procedures, you must be logged in with JIRA Administrators [global permission](#).

1. Disable email sending/receiving

If you are restoring production data into a test JIRA instance for experimentation purposes, you have to disable all JIRA application's email 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.senddisab`
`led=true` flag.
- **Disable POP/IMAP email polling** — if JIRA is configured to poll a mailbox (to create issues from mails), you will have to disable polling on your test installation by setting the `-Datlassian.mail.fetchdisab`
`led=true` flag.

Exactly how to set these flags is dependent on your particular application server, but for JIRA, this is done by setting the `DISABLE_NOTIFICATIONS` environment variable before starting JIRA (note, use `startup.sh` instead of `startup.bat` if you are not using Windows):

```
set DISABLE_NOTIFICATIONS=" -Datlassian.mail.senddisabled=true
-Datlassian.mail.fetchdisabled=true -Datlassian.mail.popdisabled=true"
cd bin
startup.bat
```

You could also try un-commenting the `DISABLE_NOTIFICATIONS=" -Datlassian.mail.senddisabled=true -Datlassian.mail.fetchdisabled=true -Datlassian.mail.popdisabled=true"` line from your `/bin/setenv.bat` file (`/bin/setenv.sh` if you are not using Windows) and then running `startup`.

2. Restore the XML data

If you've used native database tools to back up your data, the restore process will be tool-specific and The stage 2 and 3 of these instructions don't apply to you.

1. Choose



> **System.**

2. Select **Import & Export > Restore System** to open the Restore JIRA applications data from Backup

page.

Restore JIRA data from Backup

i The backup file and index paths must be located on the same machine as your JIRA instance.
You will be logged out after the restore process. Make sure you know your login details in the data being restored.

! A This will wipe all existing JIRA content - make sure you **backup first!**

File name*

Enter a filename to restore data from. Files will be loaded from `:/data/jirastudio/jira/home/import`

Index Path **By default JIRA will use the index path specified in the backup file. The following path will be used for backups without an index path: `/data/jirastudio/jira/home/caches/indexes`**

License (if required)

Only enter a license if you want to override the license in the import file.

Outgoing Mail Enable
 Disable

3. In the '**File name**' field, type the file name of the zipped XML backup file generated by JIRA.
 - i Ensure that this backup file has been moved or copied to the location specified below this field.
4. The **Index Path** field indicates where JIRA will restore the search index data from the zipped XML backup file. This location (which cannot be modified) matches the index path specified in the zipped XML backup file. If, however, this backup file does not specify an index path, JIRA will restore the search index to the `caches/indexes` subdirectory of the [JIRA application home directory](#).
 - ! A **Please Note:**
 - The contents of the index directory may be deleted by the restore process.
 - The index directory should *only* contain JIRA index data.
5. Click the '**Restore**' button and wait while your JIRA data is restored.
 - i Once the data has been restored, JIRA will inform you that you have been logged out. This happens because all JIRA users which existed in JIRA prior to JIRA's data being restored will have been deleted and replaced by users stored in the JIRA export file.

i It is recommended that you avoid passing through a proxy when performing an XML restore, especially if your JIRA instance is very large. Using a proxy may cause timeout errors.

3. Restore the 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.

! A If you use a custom directory for storing your attachments, ensure that JIRA has read and write permissions to this directory and its subdirectories.

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](#)).

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 file attachments](#) to change the location of the attachments directory so that JIRA can find the restored attachments

Restoring information from a native backup

Before you begin

Make sure that you have the password to a login in the backup file that has the JIRA System Administrator [global permission](#). Once the restoring procedure begins, all the existing data in the JIRA application database is deleted, including all user accounts.

If you are restoring data from a JIRA Cloud application site to a JIRA Server application, please read [Migrating from JIRA Cloud to JIRA Server applications](#).

For all of the following procedures, you must be logged in with JIRA Administrators [global permission](#).

1. Disable email sending/receiving

If you are restoring production data into a test JIRA instance for experimentation purposes, you have to disable all JIRA application's email 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.sen ddisabled=true` flag.
- **Disable POP/IMAP email polling** — if JIRA is configured to poll a mailbox (to create issues from mails), you will have to disable polling on your test installation by setting the `-Datlassian.mail.fe tchdisabled=true` flag.

Exactly how to set these flags is dependent on your particular application server, but for JIRA, this is done by setting the `DISABLE_NOTIFICATIONS` environment variable before starting JIRA (note, use `startup.sh` in stead of `startup.bat` if you are not using Windows):

```
set DISABLE_NOTIFICATIONS=" -Datlassian.mail.senddisabled=true
-Datlassian.mail.fetchdisabled=true -Datlassian.mail.popdisabled=true"
cd bin
startup.bat
```

You could also try un-commenting the `DISABLE_NOTIFICATIONS=" -Datlassian.mail.senddisabled=true -Datlassian.mail.fetchdisabled=true -Datlassian.mail.popdisabled=true"` line from your `/bin/setenv.bat` file (`/bin/setenv.sh` if you are not using Windows) and then running `startup`.

Follow these steps to restore data from a native backup.

1. Stop JIRA
2. Replace the JIRA Home directory with the backed up files.
3. Re-apply any changes made in the JIRA Install directory

Note

This step is required only if something has changed since the backup, it shouldn't contain any actual data.

4. Restore the database using native database tools (again this depends on the specific database type)
5. Start JIRA

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 when you add a new custom field, or if the index is lost or corrupted.

See [Re-indexing after major configuration changes](#) for more information on when you should re-index.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Re-indexing JIRA

1. Choose



> **System.**

2. Select **Advanced > Indexing** to open the Indexing page.
3. This page allows you to choose one of the following two re-indexing options:

- **Background re-index** — This will re-index all issues in the background.
- **Lock JIRA and rebuild index** — This will delete and rebuild all indices, including the comment and change history indices.

Screenshot: Re-indexing JIRA

Re-Indexing

Re-Index Option Background re-index
This will take longer but will allow users to access JIRA during the re-index.

Lock JIRA and rebuild index
JIRA will be unavailable to all users until the re-index is complete. This is faster, but may still take a while depending on the size of your instance and hardware.

Index path `/data/jirastudio/jira/home/caches/indexes`

Which re-indexing option should I use?

The following table summarizes the differences between both options:

Background re-index	Lock JIRA and rebuild index
Single-threaded, slower to complete.	Multi-threaded, faster to complete.
Can be canceled at any time.	Can't be canceled once started.
Keeps current index and updates it in-place.	Rebuilds the index, optimizes it, and deletes the old one.
Causes disk fragmentation.	Eliminates disk fragmentation.

Background re-index

This option allows any Jira instance to remain usable during re-indexing; however, the instance will be slower. If you have to perform this option, do so during a low-usage period.

In addition, your disk fragmentation increases each time you perform a **Background re-index**. Whenever possible, perform a project re-index instead, especially if you just performed configuration changes that only affect one project. See [Re-indexing a single project](#) for instructions.

Lock JIRA and rebuild index

Use this option when the indexes are corrupt, which may be caused by a system or disk failure. This option deletes all indexes and rebuilds them, and is often called a *full re-index*.

Overall, the **Lock JIRA and rebuild index** option provides greater benefits. The only downside for this option is that it'll lock a single-node instance, making it unavailable to users during the re-index.

On a multi-node Jira Data Center, you can use the **Lock JIRA and rebuild index** option without actually locking the instance. Therefore, if your Data Center instance has multiple nodes, don't bother with a **Background re-index**. See [Re-indexing JIRA Data Center with no downtime](#) for instructions.

Over time, your instance's disk becomes fragmented from normal use, which slows down your instance. Only the **Lock JIRA and rebuild index** option can address this. This means that, regardless of whether you're running a single- or multi-node instance, you should run a *full re-index* to address fragmentation periodically (for example, weekly or monthly).

You can also speed up a full re-index by increasing the number of threads it uses. See [Tune number of](#)

[index threads to make it go faster](#) for details.

Instances with lower data complexity also perform full re-indexes faster. See [Managing custom fields in JIRA effectively](#) for related information.

Choosing a custom Index Path

- If you upgraded JIRA with an [XML backup](#) from a JIRA version prior to 4.2 and used a custom directory for your index path, you can choose between using this custom directory (which cannot be edited) or the default directory for your index path location. However, once you switch to using the default directory, you can no longer choose the custom directory option.
- The default directory location is the `cache/indexes` subdirectory of the [JIRA application home directory](#).

 NFS storage for JIRA indexes is not supported. See [Supported platforms](#) for more information.

Re-indexing Jira Data Center with no downtime

Keeping the integrity of indexes is as important as having your JIRA instance open to users all the time. These steps will help you run the **Lock Jira and rebuild index** option, which deletes and recreates all indexes, with no downtime.

Before you begin:

Choose a node and remove it from the load balancer. You'll use it to perform the re-index.

To re-index Jira Data Center with no downtime:

1. Access JIRA on the node you've chosen, and select  **> System**.
2. Select **Advanced > Indexing** to open the Indexing page. Then, run **Lock Jira and re-build index**.
3. After the re-indexing is complete, take a look around the JIRA instance to make sure everything looks fine.
4. Add the node back to the load balancer.

After completing the re-indexing, the rebuilt indexes will be automatically distributed to other nodes in the cluster (there might be some performance degradation during that time). If some changes were made to the indexes in the meantime, they will also be applied to maintain the integrity.

Backing up and recovering your index

Enabling index recovery will cause a snapshot of the indexes to be taken periodically. This allows you to recover your index quickly, rather than rebuilding the index, if there is a failure. This is particularly useful if you have a large JIRA installation and you cannot afford for it to be offline for long. If you have a small JIRA instance, it may not be worth enabling index recovery, as it rebuilding the index won't take much time.

Whether a full index rebuild is faster than recovering from a snapshot depends on a number of factors, including how recent the snapshot being recovered was taken. Large and complex installations should test this process on a development/testing server before relying on it in production.

To enable index recovery:

1. Navigate to the **Indexing** page (as [described above](#)).
2. Click **Edit Settings** to enable index recovery and choose the frequency of snapshots.
 - Snapshots are stored in the `<yourjirahome>/exports/export/indexsnapshots` directory.

To recover an index:

1. Navigate to the **Indexing** page (as [described above](#)).
2. Enter the name of the previously saved index in **File name** and click **Recover**.
 - JIRA will not be available during the recovery of the index.
 - If changes were made to the configuration that required a re-index after the snapshot was taken,

then you will need to do a background re-index after the recovery. Note, JIRA will be available after the recovery.

Additional information

- JIRA will retain the last three snapshots at any time (in `<yourjirahome>/exports/export/indexsnapshots`). Older snapshots will be automatically deleted. Note, snapshots may occupy considerable disk space and may need to be moved to offline storage or deleted as appropriate.
- The snapshot process is a relatively lightweight process and does not place much of a load on the system.
- The process of taking a snapshot will require temporary disk space equivalent to the index size. The resulting snapshots will each be about 25% the size of the index.
- All issues will be re-indexed appropriately during the recovery, including issues that were added, updated or deleted after the snapshot was taken.
- You can use the index recovery process to bring your index up to date, if you need to restore your JIRA database. The index snapshot must pre-date the database backup being restored.

Re-indexing a single project

If you have made a configuration change that affects a single project, you can re-index just that project. See [Re-indexing after major configuration changes](#) for more information on when you should re-index.

To re-index a single project:

1. Navigate to the desired project and click the **Administration** tab.
2. Click **Actions > Re-index project** to start re-indexing the project.

Re-indexing after major configuration changes

Once issues have been created, modifying the configuration of your JIRA instance can result in the search index becoming out-of-sync with JIRA's configuration. Configuration details such as the following can affect the search index:

- [Field configuration schemes](#)
- [Custom fields](#)
- [Plugins](#)
- [Time tracking](#)

If you make changes to any of these areas of configuration, you might see the following message in your Administration view:

```
USERFULLNAME made configuration changes to 'SECTION' at TIME. It is recommended that you perform a re-index. It is recommended that you perform a re-index. For more information, please click the Help icon. To perform the re-index now, please go to the 'Indexing' section. Note: So that you only have to re-index once, you may wish to complete any other configuration changes before performing the re-index.
```

All users that have access to the Administration Tab will see this message (JIRA Administrators, System Administrators, Project Administrators). The above message means that configuration changes have been made to JIRA, but have not yet been reflected in the search index. Until JIRA's search index has been rebuilt, it is possible that some search queries from JIRA will return incorrect results. For example:

- If a plugin containing a custom field is enabled after being disabled, search queries which specify that the custom field should be empty will return *no issues* instead of *all issues*.
- If a [field configuration](#) is modified by altering the visibility of a particular field so that it is now visible, search queries which specify that field may also return erroneous results (depending on which field is being modified and what query is being executed).

The way to resolve the discrepancy is to [rebuild JIRA's search index](#). This can take anywhere from seconds to hours, depending on the number of issues and comments in your JIRA instance. While re-indexing is taking place, your instance will be unavailable to all users unless you chose Background Indexing. For these reasons, it

is recommended that you:

- Make all your necessary configuration changes in one go before starting the re-index process; and
- Start the re-index process in a time period of low activity for your instance.

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).

Editing robots.txt

JIRA (version 3.7 and later) installs the following robots.txt file at the root of the JIRA web app (`$JIRA-INST ALL/atlassian-jira`):

```
# robots.txt for JIRA
# You may specify URLs in this file that will not be crawled by search
engines (Google, MSN, etc)
#
# By default, all SearchRequestViews in the IssueNavigator (e.g.: Word,
XML, RSS, etc) and all IssueViews
# (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/**.

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`.

If your JIRA instance is published at `jira.mycompany.com/jira`, change the contents of the file to `Disallow: /jira/sr/` and `Disallow: /jira/si/`. 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`).

Licensing your JIRA applications

You can view and manage your JIRA application licenses on the **Versions & licenses** page. You may need to add or update your license if you:

- change the type of license (for example you may want to purchase a full license when your evaluation license expires)
- upgrade your user tier to accommodate new users
- add a new license when your old license has expired
- add a new license for a newly installed application

Before you begin

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** or **System Administrators** global permission.

Updating your JIRA license details

1. Obtain the license key you want to update (you can do this by visiting my.atlassian.com or by contacting the member of your organization who handles IT product licensing).
2. Choose  **> Applications.**
3. Select **Versions & licenses** to view license details for your installed JIRA applications.
4. Locate the license you want to update, and click on the pencil icon to edit the license.
5. Replace the existing license key with your new license key.
6. Click the '**Update license**' button to update the JIRA application with the new license.

JIRA Software 7.0.0 Unlimited users (15 used) 

Trial expires	28/Oct/15
Support entitlement number (SEN)	SEN-L1234567
License type	Evaluation
Organisation name	Example Company
License key	<div style="border: 1px solid gray; padding: 5px; min-height: 40px;">[Blurred license key text]</div>

 The updated license must be [compatible](#) with any other license(s) on your JIRA server.

To update your JIRA license key manually

1. Obtain the license key you want to update (you may do this by logging in to <http://my.atlassian.com> separately, or contacting your organization's member who deals with licensing for IT products).
2. Choose  **> Applications.**
3. Select **Versions and licenses** to view your existing JIRA applications license details.
4. Locate the license you want to update, and click on  to edit the license.
5. Replace the existing license key with your new license key in the License field.
6. Click the '**Update license**' button to update the JIRA application with the new license.

Viewing your licensed user count

1. Choose  **> Applications.**
2. Select **Versions & licenses** to view license details for your installed JIRA applications.
3. Next to the application name you can view the number of licensed users, as well as the number of users already used.

When you are approaching (or have exceeded) the maximum number of users for your license, a warning banner is displayed:



⚠️ If you exceed the user count allowed by your JIRA application's license, your users will not be able to create issues. To prevent this, either upgrade to a larger license or reduce your existing user count.

Upgrading to a larger license

1. Choose



> **Applications.**

2. Select **Versions & licenses** to view license details for your installed JIRA applications.
3. Locate the license you want to upgrade.
 - a. If you're near the maximum users for the license, you'll see a banner prompting you to upgrade. Click the **Upgrade** button to be redirected to the Atlassian order form.
 - b. If you don't see the banner but still want to upgrade, visit the [Atlassian order form](#) directly.
4. Complete your purchase at the desired user tier. When you're finished, you'll have a new license key in your [my.atlassian.com](#) account.
5. Return to the **Versions & licenses** page and update your application with the newly purchased license key.

Reducing your user count

You may want to reduce the user count for a JIRA application if you have *exceeded your user count* or if you want to *change to a lower-tier license* to reduce costs.

The *recommended method* for reducing your user count in JIRA is to remove users from the groups associated with the application. Remember a user may be a member of multiple groups, but will only count as one user on your license. See [Managing users](#) for more information.

Alternatively, if you have [connected JIRA to an LDAP directory](#), you may want configure JIRA to synchronize a subset of users from LDAP rather than all users. See [Reducing the number of users synchronized from LDAP to JIRA applications](#) for more information. However, this can be a complicated procedure and we recommend that you do not use this method unless necessary.

⚠️ Note, if you exceed the user count allowed by your JIRA application's license, your users will not be able to create issues.

Viewing your system information

JIRA provides you with detailed information about your system configuration, as described in the table below. This information can be useful when modifying, troubleshooting, or upgrading your system.

Viewing your JIRA system information

1. Log in as a user with the '**JIRA Administrators**' [global permission](#).
2. Choose



> **System.** Select **System support > System Info** to open the System Info page.

The following categories of information is shown on the 'System Info' page:

Warnings

Any warnings about known issues with your configuration will be displayed here.

System info

Setting	Description
---------	-------------

Base URL	The base URL of this JIRA installation. It is used in outgoing email notifications as the prefix for links to JIRA issues. It can be changed as described in Configuring JIRA options .
System Date	The JIRA server's system date.
System Time	The JIRA server's system time.
Current Working Directory	States the current JIRA Working Directory. Please see Important directories and files for more information.
Java Version	The JIRA server's Java version.
Java Vendor	The JIRA server's Java vendor.
JVM Version	The JIRA server's JVM version.
JVM Vendor	The JIRA server's JVM vendor.
JVM Implementation Version	The JIRA server's JVM implementation version.
Java Runtime	The JIRA server's Java runtime environment.
Java VM	The JIRA server's Java Virtual Machine.
User Name	The operating system login name which JIRA runs under.
User Timezone	The JIRA server's timezone.
User Locale	The JIRA server's locale. Unless the default language is modified in JIRA's general configuration , the User Locale will dictate the default language.
System Encoding	The JIRA server's system encoding.
Operating System	The JIRA server's operating system.
OS Architecture	The JIRA server's operating system architecture (e.g. i386).
Application Server Container	The application server in which your JIRA instance is running. (See Supported platforms for a list of supported application servers.)
Database type	The type of database to which your JIRA instance is connected. (See Supported platforms for a list of supported application servers.)
Database JNDI address	The JNDI address of the database to which your JIRA instance is connected. (For more details, see Connecting JIRA to a database .)
Database URL	The URL of the database to which your JIRA instance is connected. (For more details, see Connecting JIRA to a database .)
Database version	The version of the database to which your JIRA instance is connected. (See Supported platforms for a list of supported application servers.)

Database driver	The driver which your JIRA instance is using to connect to its database.(For more details, see Connecting JIRA to a database.)
External user management	' ON ' / ' OFF ' indicates whether JIRA's users are being managed externally or internally to JIRA (e.g. via Crowd).
Crowd integration	' YES ' / ' NO ' indicates whether Atlassian's Crowd identity management system has been integrated with this instance of JIRA. For more information please see the chapter titled ' Integrating JIRA with Crowd ' in the Crowd documentation.
JVM Input Arguments	A list of any variables that are being passed to your application server when it starts up.(For more information, see Setting properties and options on startup.)
Modified Files	A list of any files in your JIRA installation that have been modified as part installation or customization of JIRA.
Removed Files	A list of any files that have been removed from your JIRA installation.

Java VM Memory Statistics

Java applications, such as JIRA, run in a "Java virtual machine" (JVM) instead of directly within an operating system. When started, the Java virtual machine is allocated a certain amount of memory, which it makes available to applications like JIRA. The following table shows the JVM memory data for your JIRA instance.

Setting	Description
Total Memory	The total amount of memory allocated to the JVM that is available to this instance of JIRA.(For more details, see Increasing JIRA memory.)
Free Memory	The amount of free JVM memory currently available to this instance of JIRA.
Used Memory	The amount of JVM memory currently being used by this instance of JIRA.
Total PermGen Memory	The total amount of PermGen (Permanent Generation) memory available to this instance of JIRA.
Free PermGen Memory	The amount of free PermGen (Permanent Generation) memory currently available to this instance of JIRA.
Used PermGen Memory	The amount of PermGen (Permanent Generation) memory currently being used by this instance of JIRA.
Memory Graph	A bar graph showing the available versus free JVM memory. You can click the ' Force garbage collection ' link to start a clean-up. Note that this is generally not needed (even if the graph shows 100% utilization) unless you want to examine JIRA's baseline heap usage.

PermGen Memory Graph	A bar graph showing the available versus free PermGen (Permanent Generation) memory.
Non-Heap Memory Graph (includes PermGen)	A bar graph showing the available versus free non-heap memory (including PermGen memory).

You can click the '**More Information...**' link at the bottom of this table to view an additional section titled '**Memory Pool Info**' (which lists detailed information about the various parts of memory that the Java virtual machine uses to store its data, and is generally only useful to Atlassian's support engineers.)

JIRA Info

Setting	Description
Uptime	The period of time since your JIRA instance was last started.
Edition	The 'edition' of JIRA you are running.
Version	The version of JIRA you are running.
Build Number	The build number of your JIRA version. This is generally only useful to Atlassian's support engineers.
Build Date	The date on which your JIRA version was built. This is generally only useful to Atlassian's support engineers.
Atlassian Partner	Indicates whether your distribution of JIRA was built by an Atlassian partner company. Blank indicates that it was built directly by Atlassian.
Installation Type	Indicates whether JIRA has been installed as a ' recommended ' distribution or as a 'WAR' distribution. (Note we no longer support WAR installations or builds.)
Server ID	This number is calculated automatically by JIRA, based on your license number.
Last Upgrade	The time at which your JIRA installation was last upgraded, and from which version it was upgraded from (if applicable). Click the ' More Information... ' link to see a list of all upgrades that have been performed on your JIRA system from version 4.1 onwards.
Installed Languages	A list of all language packs available within the JIRA system. (Note: to install additional languages, see Translating JIRA .)
Default Language	The language used throughout the JIRA interface. To change the default language, see Configuring JIRA options . Note that users can override the default language by changing the Language preference in their user profile.

License Info

 To edit your license details, see [Licensing your JIRA applications](#). Note that you will require the '**JIRA**

System Administrators' global permission.

Setting	Description
Date Purchased	The date on which this system's JIRA license was originally purchased. Note: you can verify this information by visiting http://my.atlassian.com
License Type	For information about the different types of JIRA licenses, please see http://www.atlassian.com/software/jira/licensing.jsp
Maintenance Period End Date	For information about JIRA support and maintenance, please see http://www.atlassian.com/software/jira/licensing.jsp
Maintenance Status	For information about JIRA support and maintenance, please see http://www.atlassian.com/software/jira/licensing.jsp
Support Entitlement Number (SEN)	For information about JIRA support and maintenance, please see http://www.atlassian.com/software/jira/licensing.jsp

Configuration Info

Setting	Description
Attachments Enabled	'true' / 'false' indicates whether or not users can attach files and screenshots to issues in this JIRA system (subject to project permissions). For more information, see Configuring file attachments .
Issue Voting Enabled	'true' / 'false' indicates whether or not users can vote on issues in this JIRA system (subject to project permissions). For more information, see Configuring JIRA options .
Issue Watching Enabled	'true' / 'false' indicates whether or not users can watch issues in this JIRA system (subject to project permissions). For more information, see Configuring JIRA options .
Unassigned Issues Enabled	'true' / 'false' indicates whether or not issues can be 'unassigned' (i.e. assigned to no one) in this JIRA system. For more information, see Configuring JIRA options .
Sub-Tasks Enabled	'true' / 'false' indicates whether or not 'sub-task' issues can be created in this JIRA system. For more information, see Configuring sub-tasks .
Issue Linking Enabled	'true' / 'false' indicates whether or not issues can be linked to each other within this JIRA system. For more information, see Configuring issue linking .
Time Tracking Enabled	'true' / 'false' indicates whether or not time (work) can be logged on issues in this JIRA system. For more information, see Configuring time tracking .

Time Tracking Hours Per Day	The number of hours per working day for which work that can be logged on issues in this JIRA system. For more information, see Configuring time tracking .
Time Tracking Days Per Week	The number of days per week for which work that can be logged on issues in this JIRA system. For more information, see Configuring time tracking .

Database statistics

The information in this section can help determine how much resource (e.g. memory) your JIRA system requires.

Setting	Description
Issues	The number of issues that have been created in this JIRA system.
Projects	The number of projects that have been created in this JIRA system.
Custom Fields	The number of custom fields that have been created in this JIRA system.
Workflows	The number of workflows that have been created in this JIRA system.
Users	The number of user IDs that have been created in this JIRA system.
Groups	The number of groups that have been created in this JIRA system.

File Paths

Setting	Description
Location of JIRA Home	The path to your JIRA home directory. (For information about changing the location, see Setting your JIRA application home directory .)
Location of entityengine.xml	The path to your Entity Engine. (For database-specific information about configuring your <code>entityengine.xml</code> file, see Connecting JIRA applications to a database .)
Location of atlassian-jira.log	The path to the JIRA log file. Note that, if you are requesting support, the support engineers will generally need your application server log file as well as your JIRA log file. (For information about changing the logging level, see Logging and profiling ; note that you will require the 'JIRA System Administrators' global permission.)
Location of indexes	The path to your JIRA search indexes, not your database indexes. (For information about moving the indexes, please see Search indexing ; note that you will require the 'JIRA System Administrators' global permission.)

Listeners

This section lists all the listeners that are installed in this JIRA system. For more information, see [Listeners](#). Note that you will require the '**JIRA System Administrators**' [global permission](#) in order to register a listener.

Services

This section lists all the services that are installed in this JIRA system. For more information, please see [Services](#). Note that you will require the '**JIRA System Administrators**' [global permission](#) in order to register a service.

Add-ons

The add-on sections lists all plugins that are installed in this JIRA system, broken down by System Add-ons and User installed Add-ons. For more information, please see [Managing add-ons](#).

System properties

The information in this section is specific to the application server and Java version you are using, and is generally only useful to Atlassian's support engineers.

Trusted Applications

This section lists all 'trusted application' (i.e. applications that JIRA will allow to access specified functions on behalf of any user — without the user logging in to JIRA). Trusted applications have now been superseded by application links, and you can find more information on [application links](#) here.

Live monitoring using the JMX interface

This article describes how to expose JMX MBeans within JIRA for monitoring with a JMX client.

This guide provides a basic introduction to the JMX interface and is provided as is. Our support team can help you troubleshoot a specific JIRA problem, but aren't able to help you set up your monitoring system or interpret the results.

What is JMX?

JMX ([Java Management Extensions](#)) is a technology for monitoring and managing Java applications. JMX uses objects called MBeans (Managed Beans) to expose data and resources from your application. For large instances of JIRA Server or JIRA Data Center, enabling JMX allows you to more easily monitor the consumption of application resources. This enables you to make better decisions about how to maintain and optimize machine resources.

Metrics collected by JIRA

The following table lists metrics (MBeans) that are collected by JIRA. All of them are grouped in the `com.atlassian.jira` property.

Metric	Description	Reset after restarting JIRA
dashboard.view.count	The number of times all dashboards were viewed by users.	Yes
entity.attachments.total	The number of attachments.	-
entity.components.total	The number of components.	-
entity.customfields.total	The number of custom fields.	-
entity.filters.total	The number of filters.	-

entity.groups.total	The number of user groups.	-
entity.issues.total	The number of issues.	-
entity.users.total	The number of users.	-
entity.versions.total	The number of versions created.	-
issue.assigned.count	The number of times issues were assigned or reassigned to users (counts each action).	Yes
issue.created.count	The number of issues that you created after starting your JIRA instance.	Yes
issue.link.count	The number of issue links created after starting your JIRA instance.	Yes
issue.search.count	The number of times you searched for issues.	Yes
issue.updated.count	The number of times you updated issues (each update after adding or changing some information).	Yes
issue.worklogged.count	The number of times you logged work on issues.	Yes
jira.license	The types of licenses you have, the number of active users, and the maximum number of users available for each license type.	-
quicksearch.concurrent.search	The number of concurrent searches that are being performed in real-time by using the quick search. You can use it to determine whether the limit set for concurrent searches is sufficient or should be increased.	Yes
web.requests	The number of requests (invocation.count), and the total response time (total.elapsed.time).	Yes

Monitoring JIRA

Before you can monitor JIRA, you need to enable JMX monitoring and then use a JMX client to view the metrics.

Good to know

Viewing the metrics will always have some performance impact on JIRA. We recommend that you don't refresh them more than once a second.

Enabling JMX monitoring in JIRA

All of the metrics are collected by default, but you need to enable JMX monitoring to expose them. You can do it JIRA, but you need to be a JIRA administrator to get there.

1. In JIRA, go to



> **System > JMX monitoring.**

2. Toggle **Enable JMX monitoring.**

Monitoring with JConsole

After you enabled JMX monitoring, you can use any JMX client to view the metrics. To make it quick and easy, we've described how to view them by using JConsole. You can monitor your JIRA instance either locally, or remotely:

- **Monitoring JIRA locally** is good if you're troubleshooting a particular issue, or only need to monitor JIRA for a short time. Local monitoring can have a performance impact on your server, so it's not recommended for long-term monitoring of your production system.

▼ [Show me how to do this...](#)

To monitor locally:

1. Start JConsole (you'll find it in the `bin` directory of the JDK installation directory).
2. Select **Local Process**.
3. Select the JIRA process. It will be called something like `org.apache.catalina.startup.Bootstrap start`
4. After connecting, expand the `com.atlassian.jira` property that groups all the metrics.

See [Using JConsole](#) for more information on local monitoring.

- **Monitoring JIRA remotely** is recommended for production systems, as it does not consume resources on your JIRA server.

▼ [Show me how to do this...](#)

To monitor remotely:

1. Add the following properties to your `setenv.sh / setenv.bat` file. The port can be any port that is not in use.

```
set CATALINA_OPTS=-Dcom.sun.management.jmxremote
%CATALINA_OPTS% set
CATALINA_OPTS=-Dcom.sun.management.jmxremote.port=8099
%CATALINA_OPTS%
```

2. Decide how you will secure your remote connection. See [Remote Monitoring and Management](#) for more information.

Although it is possible to disable authentication, we do not recommend doing this on a production system.

3. Start JConsole (you'll find it in the `bin` directory of the JDK installation directory).
4. Select **Remote Process**.
5. Enter your hostname and port (this is the port you specified earlier, not the JIRA port).
6. Click **Connect**.
7. After connecting, expand the `com.atlassian.jira` property that groups all the metrics.

See [Using JConsole](#) for more information on remote monitoring.

Monitoring database connection usage

JIRA provides a view of its database connection usage. This provides information on the activity of the connection pool, as well as the frequency of reads/writes to the database. You can use this information to tune your database connections for better performance.

The instructions on this page describe how to navigate to the database connection usage information in the JIRA administration console, and how to interpret the information. If you want to make changes to your database connection pool settings using this information, see this related topic: [Tuning database connections](#).

Note: For all of the following procedures, you must be logged in as a user

On this page:

- [Accessing the Database Monitoring page](#)
- [Interpreting the database monitoring graphs](#)

with the **JIRA Administrators** global permission.

Related pages:

- [Tuning database connections](#)
- [Enterprise Resources](#)

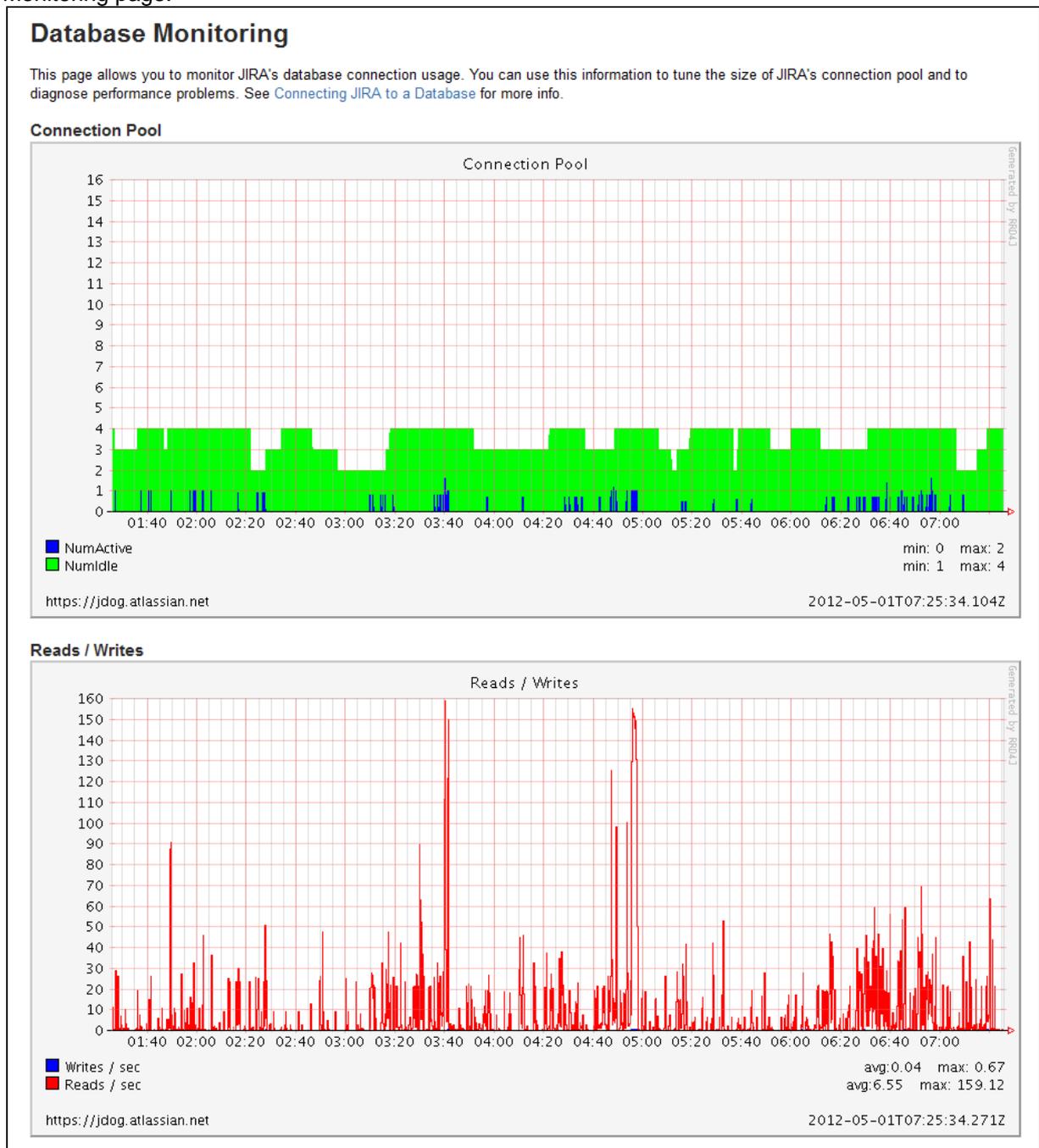
Accessing the Database Monitoring page

1. Choose



> **System.**

2. Select **Database Monitoring**, which can be found under **System support** to display the Database Monitoring page.



Interpreting the database monitoring graphs

Connection Pool graph

The 'Connection Pool' graph shows the activity in the connection pool for the last 6 hours.

- This graph shows the number of active and idle connections, as well as the maximum and minimum for the period.
- The scale of the vertical axis is equal to the maximum number of connections.
- The readings are averages over a period of 5 minutes.

This information can help you to optimize database connection usage. For example, if the number of active connections is consistently or frequently near to the maximum available, then you may need to raise the maximum connections available in the pool. Conversely, if the number of active connections is consistently low compared to the maximum available, then you may want to lower the maximum connections available in the pool. For more information on how to tune database connections, see [Tuning database connections](#).

Reads / Writes graph

The 'Reads / Writes' graph shows the frequency of reads and writes to the database over a period of time. It can be helpful to correlate database usage with connection pool usage. Whenever JIRA needs to access (i.e. read from or write to) the database, a database connection is required. If there are regular spikes in the reads / writes, you may need to consider raising the maximum connections available in the pool.

Viewing JIRA application instrumentation statistics

JIRA provides an **Instrumentation** page, which displays a variety of statistics on a wide range of internal properties within JIRA that have been 'instrumented' (i.e. recorded) for presentation through JIRA's administration area.

This page is mostly useful to help Atlassian Support provide assistance with your support queries, especially if they ask you to quote the statistics of one or more properties listed on this page.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

1. Choose



> **System.**

2. Select **System support > Instrumentation** to display the Instrumentation page.

Name	Type	Value	Invocation	Time (ms)	CPU (nanos)
cache.CachingFieldConfigContextPersister.evictionCount	Counter	0			
cache.CachingFieldConfigContextPersister.hitCount	Counter	1,314			
cache.CachingFieldConfigContextPersister.loadExceptionCount	Counter	0			
cache.CachingFieldConfigContextPersister.loadSuccessCount	Counter	6			
cache.CachingFieldConfigContextPersister.missCount	Counter	6			
cache.CachingFieldConfigContextPersister.size	Gauge	6			
cache.CachingFieldConfigContextPersister.totalLoadTime	Counter	5			
cache.DefaultFieldLayoutManager.evictionCount	Counter	0			
cache.DefaultFieldLayoutManager.hitCount	Counter	14,533			
cache.DefaultFieldLayoutManager.loadExceptionCount	Counter	0			
cache.DefaultFieldLayoutManager.loadSuccessCount	Counter	3			
cache.DefaultFieldLayoutManager.missCount	Counter	4			
cache.DefaultFieldLayoutManager.size	Gauge	1			
cache.DefaultFieldLayoutManager.totalLoadTime	Counter	12			
cache.DefaultIssueLinkManager.evictionCount	Counter	0			

cache.DefaultIssueLinkManager.hitCount	Counter	1,935
cache.DefaultIssueLinkManager.loadExceptionCount	Counter	0
cache.DefaultIssueLinkManager.loadSuccessCount	Counter	602
cache.DefaultIssueLinkManager.missCount	Counter	602
cache.DefaultIssueLinkManager.size	Gauge	602
cache.DefaultIssueLinkManager.totalLoadTime	Counter	4,701
cache.DefaultPermissionSchemeManager.evictionCount	Counter	0
cache.DefaultPermissionSchemeManager.hitCount	Counter	17,824
cache.DefaultPermissionSchemeManager.loadExceptionCount	Counter	0
cache.DefaultPermissionSchemeManager.loadSuccessCount	Counter	2
cache.DefaultPermissionSchemeManager.missCount	Counter	2
cache.DefaultPermissionSchemeManager.size	Gauge	1
cache.DefaultPermissionSchemeManager.totalLoadTime	Counter	6
cache.DefaultUserPropertyManager.evictionCount	Counter	38
cache.DefaultUserPropertyManager.hitCount	Counter	713,077
cache.DefaultUserPropertyManager.loadExceptionCount	Counter	0
cache.DefaultUserPropertyManager.loadSuccessCount	Counter	43
cache.DefaultUserPropertyManager.missCount	Counter	43
cache.DefaultUserPropertyManager.size	Gauge	5
cache.DefaultUserPropertyManager.totalLoadTime	Counter	17
cache.JiraOsgiContainerManager.evictionCount	Counter	80
cache.JiraOsgiContainerManager.hitCount	Counter	34,914
cache.JiraOsgiContainerManager.loadExceptionCount	Counter	0
cache.JiraOsgiContainerManager.loadSuccessCount	Counter	86
cache.JiraOsgiContainerManager.missCount	Counter	86
cache.JiraOsgiContainerManager.size	Gauge	6
cache.JiraOsgiContainerManager.totalLoadTime	Counter	134
cache.VelocityTemplateCache.directives.evictionCount	Counter	0
cache.VelocityTemplateCache.directives.hitCount	Counter	419,353
cache.VelocityTemplateCache.directives.loadExceptionCount	Counter	0
cache.VelocityTemplateCache.directives.loadSuccessCount	Counter	76
cache.VelocityTemplateCache.directives.missCount	Counter	76
cache.VelocityTemplateCache.directives.size	Gauge	76
cache.VelocityTemplateCache.directives.totalLoadTime	Counter	0
cache.VelocityTemplateCache.evictionCount	Counter	0
cache.VelocityTemplateCache.hitCount	Counter	419,353
cache.VelocityTemplateCache.loadExceptionCount	Counter	0
cache.VelocityTemplateCache.loadSuccessCount	Counter	76
cache.VelocityTemplateCache.missCount	Counter	76
cache.VelocityTemplateCache.size	Gauge	76
cache.VelocityTemplateCache.totalLoadTime	Counter	10

concurrent.users	Gauge	1			
db.conns	Operation		112,352	1,058,122,695	0
db.conns.borrowed	Gauge	1			
db.reads	Operation		103,234	2,238	0
db.writes	Operation		4,669	406	0
dbcp.maxActive	Gauge	20			
dbcp.numActive	Gauge	1			
dbcp.numIdle	Gauge	19			
entity.customfields.total	Gauge	1			
entity.groups.total	Gauge				
entity.issues.total	Gauge	301			
entity.projects.total	Gauge	2			
entity.users.total	Gauge				
entity.workflows.total	Gauge	4			
http.session.objects	Gauge	10			
http.sessions	Gauge	2			
index.writes	Operation		9	13,829	0
issue.index.reads	Operation		96	653	0
jmx.class.loaded.current	Gauge	26,029			
jmx.class.loaded.total	Counter	27,594			
jmx.class.unloaded.total	Counter	1,565			
jmx.gc	Operation		194	25,087	0
jmx.memory.heap.committed	Gauge	659,767,296			
jmx.memory.heap.used	Gauge	564,776,504			
jmx.memory.nonheap.committed	Gauge	146,669,568			
jmx.memory.nonheap.used	Gauge	146,505,104			
jmx.system.up.time	Gauge	1,058,358,664			
jmx.thread.cpu.block.count	Counter	0			
jmx.thread.cpu.block.time	Counter	0			
jmx.thread.cpu.time	Counter	323,250,000,000			
jmx.thread.cpu.user.time	Counter	259,125,000,000			
jmx.thread.cpu.wait.count	Counter	0			
jmx.thread.cpu.wait.time	Counter	0			
jmx.thread.daemon.count	Gauge	32			
jmx.thread.ever.count	Gauge	4,445			
jmx.thread.nondaemon.count	Gauge	11			
jmx.thread.peak.count	Gauge	47			
jmx.thread.total.count	Gauge	43			
searcher.lucene.close	Counter	10			
searcher.lucene.open	Counter	13			
web.requests	Operation		4,328	623,512	0

JMX Support Information

- Thread Contention Monitoring - Supported - Not Enabled
- Thread CPU Time Monitoring - Supported - Not Enabled

Turn Thread Contention And CPU Monitoring On

Generating a thread dump

Occasionally, 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 below.

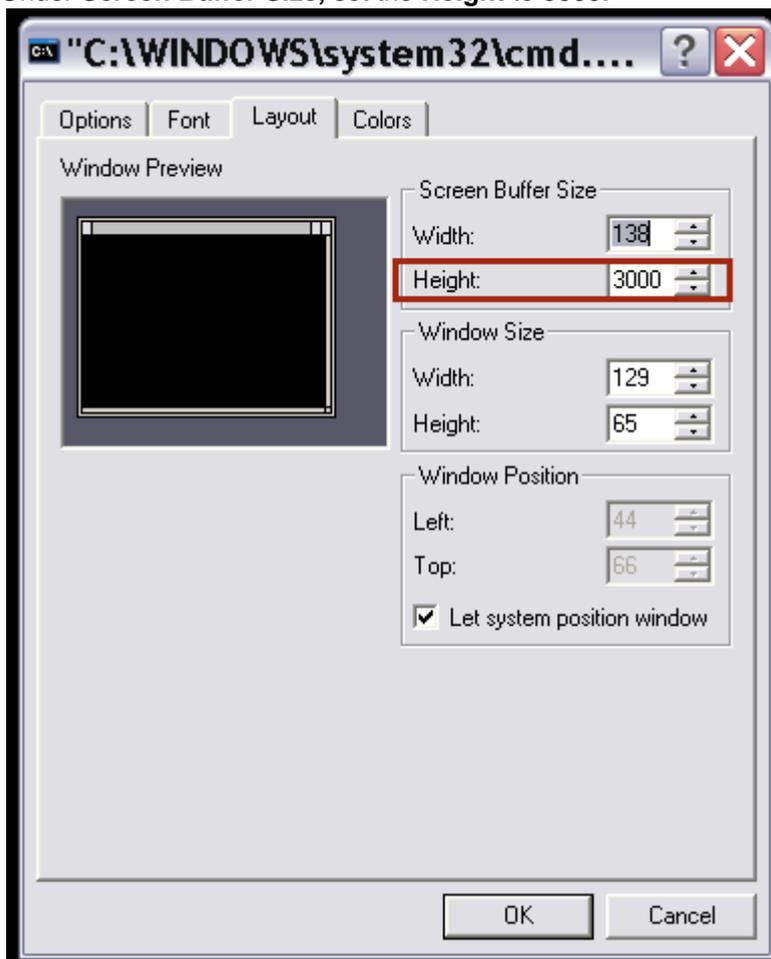
On this page:

- [Windows environment](#)
- [Linux/Unix/OS X environment](#)
- [Analysis tools](#)

Windows environment

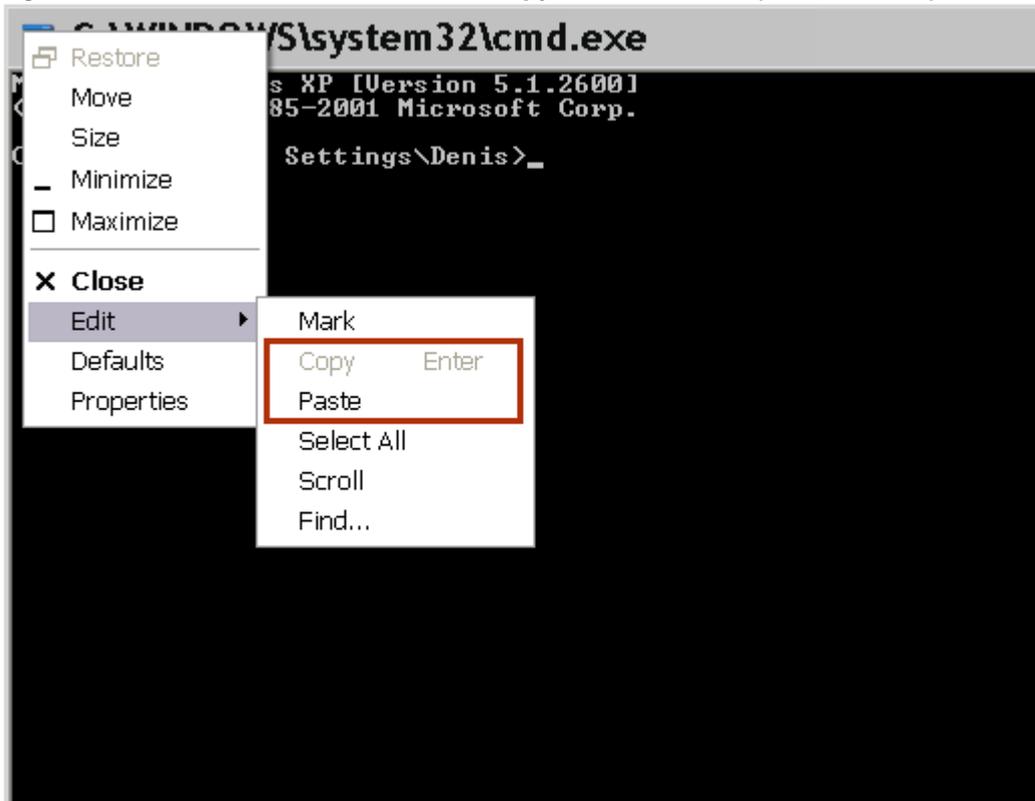
JIRA running from startup.bat

1. In the **Command Console** window where JIRA is running, open the properties dialog box by right clicking on the title bar and select "**Properties**".
2. Select the **Layout** tab.
3. Under **Screen Buffer Size**, set the **Height** to **3000**.



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".
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.



JIRA running as a Windows service

Using jstack

The JDK ships with a tool named `jstack` for generating thread dumps.

1. Identify the process. Launch the task manager by, pressing `Ctrl + Shift + Esc` and find the Process ID of the Java (JIRA) process. You may need to add the PID column using `View -> Select Columns ...`
2. Run `jstack <pid>` to Capture a Single Thread Dump. This command will take one thread dump of the process id `<pid>`, in this case the pid is 22668:

```
C:\Users\Administrator>jstack.exe -l 22668 > threaddump.txt
```

This will output a file called `threaddump.txt` to your current directory.

Common issues with jstack:

- You must run `jstack` as the same user that is running JIRA.
- If you get the error "Not enough storage is available to process this command", download the 'psexec' utility from [here](#), then run the following command using it:
`psexec -s jstack <pid> >> threaddumps.txt`
- If the `jstack` executable is not in your `$PATH`, then please look for it in your `<JDK_HOME>/bin` directory
- If you receive `java.lang.NoClassDefFoundError: sun/tools/jstack/JStack` check that `tools.jar` is present in your JDK's lib directory. If it is not, download a full version of the JDK.

Linux/Unix/OS X environment

Linux/Unix command line

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 -Xmx256m
-Xms128m -Xmx256m -Djava.awt.headless=true
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManage
r
-Djava.awt.headless=true
-Djava.endorsed.dirs=/tmp/atlassian-jira-enterprise-3.6-standalo
ne/common/endorsed
-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 will be written to the Tomcat console output. The console output is redirected to the `logs/catalina.out` file, which can be found in the [JIRA application installation directory](#) for JIRA Standalone / Installer.

Linux/Unix Alternative: Generating thread dumps using jstack

If you have trouble using `kill -3 <pid>` to obtain a thread dump, try using `jstack` a java utility that will output stack traces of Java threads for a given process.

1. Identify the **java** process that JIRA is running in. This can be achieved by running a command similar to:

```
ps -ef | grep java
```

2. The process will appear similarly as follows:

```
adam 22668 0.3 14.9 1691788 903928 ? Sl Jan27 9:36
/usr/lib/jvm/java-6-sun-1.6.0.14/bin/java
-Djava.util.logging.config.file=/home/adam/Products/installs/atlassian-jira-enterprise-4.0.1-standalone/conf/logging.properties
-XX:MaxPermSize=256m -Xms128m -Xmx1048m -Djava.awt.headless=true
-Datlassian.standalone=JIRA
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true
-Dmail.mime.decodeparameters=true
-Datlassian.mail.senddisabled=false
-Datlassian.mail.fetchdisabled=false
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djava.endorsed.dirs=/home/adam/Products/installs/atlassian-jira-enterprise-4.0.1-standalone/common/endorsed -classpath
/home/adam/Products/installs/atlassian-jira-enterprise-4.0.1-standalone/bin/bootstrap.jar
-Dcatalina.base=/home/adam/Products/installs/atlassian-jira-enterprise-4.0.1-standalone
-Dcatalina.home=/home/adam/Products/installs/atlassian-jira-enterprise-4.0.1-standalone
-Djava.io.tmpdir=/home/adam/Products/installs/atlassian-jira-enterprise-4.0.1-standalone/temp
org.apache.catalina.startup.Bootstrap start
```

3. Run `jstack <pid>` to capture a single thread dump

This command will take one thread dump of the process id `<pid>`, in this case the pid is 22668, and log output to the file `JIRAthreaddump.txt`

```
adam@jiratrack:~$ jstack 22668 > JIRAthreaddump.txt
```

4. Take multiple thread dumps

Typically you'll want to take several dumps about 10 seconds apart, in which case you can generate several dumps and output the stack traces to a single file as follows:

```
adam@jiratrack:~$ jstack 22668 >> JIRAthreaddump.txt
adam@jiratrack:~$ jstack 22668 >> JIRAthreaddump.txt
adam@jiratrack:~$ jstack 22668 >> JIRAthreaddump.txt
```

If you are connecting to the server through RDP, `jstack` might fail with following error:

```
Not enough storage is available to process this command
```

You will need to open a RDP session in console mode: `mstsc /admin`

Linux/Unix Alternative: Generating thread dumps using scripts

You can also generate a thread dump by using scripts prepared by our Support team. That's an easy process, and the scripts will do everything for you. As an addition to generating a thread dump, the scripts also allow you to generate heap dumps, check the disk access speed, or the Java SSL connection.

1. (Optional) Install the [Thready](#) plugin if it's supported by your JIRA version. Thready renames Tomcat

- threads to include the URL they serve, which will make it easier to analyze the threads.
2. Download and install the scripts from <https://bitbucket.org/atlassianlabs/atlassian-support/>.
 3. Execute the scripts when your JIRA instance behaves slowly or is unresponsive.
 - a. (Optional) The scripts also allow you to test the disk access speed. This is described in more detail in [Testing disk access speed](#).
 - b. When asked whether you want to capture thread dumps, enter **Y**.
 - c. When asked whether you want to capture heap dumps, enter **N**.
 - d. (Optional) The scripts also allow you to check the Java SSL connection.
 4. After running the scripts, the thread dump will be captured and compressed. You can now open a Support ticket and attach the generated package.

Analysis tools

Try [TDA](#) or [Samurai](#) to inspect your thread dump.

TDA

1. Download [TDA](#).
2. CD to the directory where the JAR exists.
3. Run:

```
java -jar -Xmx512M ~/tda-bin-1.6/tda.jar
```

4. Open your catalina.out file, containing the thread dump.

Issue processing thread dump with TDA (NumberFormatException)

Should you get an error on TDA console like:

```
Exception in thread "AWT-EventQueue-0"  
java.lang.NumberFormatException: For input string: "5 os_prio=0"  
    at  
    java.lang.NumberFormatException.forInputString(NumberFormatException.  
java:65)  
    at java.lang.Long.parseLong(Long.java:441)  
    at java.lang.Long.<init>(Long.java:702)  
    at  
    com.pironet.tda.utils.ThreadsTableModel.getValueAt(ThreadsTableModel.  
java:80)  
    at  
    com.pironet.tda.utils.TableSorter.getValueAt(TableSorter.java:285)  
    at javax.swing.JTable.getValueAt(JTable.java:2717)  
    at javax.swing.JTable.prepareRenderer(JTable.java:5719)  
    at  
    javax.swing.plaf.basic.BasicTableUI.paintCell(BasicTableUI.java:2114)  
    at  
    javax.swing.plaf.basic.BasicTableUI.paintCells(BasicTableUI.java:2016  
)  
    at  
    javax.swing.plaf.basic.BasicTableUI.paint(BasicTableUI.java:1812)  
    at javax.swing.plaf.ComponentUI.update(ComponentUI.java:161)  
    at javax.swing.JComponent.paintComponent(JComponent.java:778)  
    at javax.swing.JComponent.paint(JComponent.java:1054)  
    at javax.swing.JComponent.paintChildren(JComponent.java:887)  
    at javax.swing.JComponent.paint(JComponent.java:1063)  
    at javax.swing.JViewport.paint(JViewport.java:731)  
    at javax.swing.JComponent.paintChildren(JComponent.java:887)  
    at javax.swing.JComponent.paint(JComponent.java:1063)
```

```
    at javax.swing.JComponent.paintChildren(JComponent.java:887)
    at javax.swing.JSplitPane.paintChildren(JSplitPane.java:1047)
    at javax.swing.JComponent.paint(JComponent.java:1063)
    at
javax.swing.JComponent.paintToOffscreen(JComponent.java:5230)
    at
javax.swing.BufferStrategyPaintManager.paint(BufferStrategyPaintManager.java:295)
    at javax.swing.RepaintManager.paint(RepaintManager.java:1249)
    at
javax.swing.JComponent._paintImmediately(JComponent.java:5178)
    at
javax.swing.JComponent.paintImmediately(JComponent.java:4989)
    at javax.swing.RepaintManager$3.run(RepaintManager.java:808)
    at javax.swing.RepaintManager$3.run(RepaintManager.java:796)
    at java.security.AccessController.doPrivileged(Native Method)
    at
java.security.ProtectionDomain$1.doIntersectionPrivilege(ProtectionDomain.java:76)
    at
javax.swing.RepaintManager.paintDirtyRegions(RepaintManager.java:796)
    at
javax.swing.RepaintManager.paintDirtyRegions(RepaintManager.java:769)
    at
javax.swing.RepaintManager.prePaintDirtyRegions(RepaintManager.java:718)
    at
javax.swing.RepaintManager.access$1100(RepaintManager.java:62)
    at
javax.swing.RepaintManager$ProcessingRunnable.run(RepaintManager.java:1677)
    at
java.awt.event.InvocationEvent.dispatch(InvocationEvent.java:312)
    at java.awt.EventQueue.dispatchEventImpl(EventQueue.java:733)
    at java.awt.EventQueue.access$200(EventQueue.java:103)
    at java.awt.EventQueue$3.run(EventQueue.java:694)
    at java.awt.EventQueue$3.run(EventQueue.java:692)
    at java.security.AccessController.doPrivileged(Native Method)
    at
java.security.ProtectionDomain$1.doIntersectionPrivilege(ProtectionDomain.java:76)
    at java.awt.EventQueue.dispatchEvent(EventQueue.java:703)
    at
java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:242)
    at
java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:161)
    at
java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:150)
    at
java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:146)
    at
```

```
java.awt.EventQueueDispatchThread.pumpEvents(EventDispatchThread.java:138)
    at
java.awt.EventQueueDispatchThread.run(EventDispatchThread.java:91)
```

Apply the following command on the thread dump(s) to fix the thread header format to make it processable:

```
sed -i 's/prio=[0-9]\{1,2\} os_prio=[0-9]\{1,2\}/prio=5/g' <filename>
```

▼ For Mac:

```
sed -i '' 's/prio=[0-9]\{1,2\} os_prio=[0-9]\{1,2\}/prio=5/g' <filename>
```

Check the known thread dump knowledge base articles:

- [Poor performance due to limited database connection pooling](#)
- [Searching, Indexing, and filters troubleshooting](#)
- [OutOfMemory or Poor Performance due to XML View of a Filter](#)
- [JIRA Deadlocks when Running Tomcat 6.0.24](#)
- [JIRA applications performance tuning](#)
- [JIRA applications crash due to OutOfMemoryError Java heap space](#)

Finding your JIRA application Support Entitlement Number (SEN)

There are three ways to find your Support Entitlement Number (SEN).

See [How to find your Support Entitlement Number \(SEN\)](#) in the support space for more general information about how Atlassian Support uses this number.

Method 1: Check in the JIRA administration interface

Access the JIRA license page, as described in [Licensing your JIRA applications](#). The JIRA license page will show your Support Entitlement Number (SEN).

License Information	
This page shows your current licensing information. You can use the Update License form to update the license JIRA is running with.	
Organization	dailyplanet
Date Purchased	16/Nov/11
License Type	(Support and updates available until 15/Nov/11)
Server ID	BKPQ-M5IU-F3AD-U6LF
Support Entitlement Number (SEN)	SEN-500
User Limit	Unlimited
Update License	
Copy and paste the license key below. You can access your license key on My Account .	
License	<input type="text"/>
<input type="button" value="Add"/>	

Method 2: Check my.atlassian.com

Your Support Entitlement Number is available from the licenses page after logging in to <http://my.atlassian.com>:

Welcome

Hey [redacted], welcome to the Atlassian customer portal. For questions relating to [managing your account](#), licensing or purchasing, please see the [purchasing and licensing FAQ](#) or [contact a customer service representative](#). To request technical support, please visit our [Support Portal](#).

Licenses [New Evaluation License](#) | [New SourceTree License](#)

You do not have any product licenses.

Evaluations [New Evaluation License](#) | [Expand All](#) | [Collapse All](#)

Product	Name	Support Expires	Support
<input type="checkbox"/> JIRA (Unlimited Users): Evaluation	Atlassian	14 Dec [redacted]	

Server ID: BCQU-[redacted]
 SEN: SEN-[redacted]

License Key: [redacted]

This license is compatible with JIRA 4 or above.

Actions: [Buy](#) | [Download JIRA](#)

Method 3: Check your Atlassian invoice

Your Support Entitlement Number (SEN) also appears on the third page of your Atlassian Invoice.

Auditing in JIRA applications

About auditing in JIRA applications

The auditing feature tracks key activities in JIRA applications. These activities are recorded in an audit log that can be viewed in the JIRA administration console. This can be a handy tool in helping you diagnose problems in JIRA applications or used for security purposes.

The following information is audited by JIRA applications:

- LDAP synchronization
- user management
- group management
- project changes
- permission changes
- workflow changes
- notification scheme changes
- screen changes
- custom field changes
- workflow changes

On this page:

- [About auditing in JIRA applications](#)
- [Viewing the audit log](#)
- [Hiding external directory user events \(LDAP/Crowd events\)](#)
- [Modifying the audit log retention period](#)
- [Exporting the audit log](#)
- [Auditing and the REST API](#)

The audit log is not intended to record all activity in JIRA applications, as can be seen above. For example, it does not track issue updates or pages that are viewed by a user. Rather, the audit log is intended to record configuration changes that can impact users and projects. The full list of events recorded by JIRA applications can be seen [below](#).

Viewing the audit log

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose  **> System > Audit Log**.
3. The following events are audited:

Category	Events
Auditing	auditing enabled, auditing disabled
LDAP synchronization	LDAP synchronization
User management	user added, user removed, user changed
Group management	group added, group removed, user added to group, user removed from group
Project changes	project created, project removed, project updated
Permission changes	scheme created, scheme copied, scheme removed, scheme edited, scheme assigned to a project, scheme unassigned from a project, permission added to scheme, permission removed from scheme, global permission added to a group, global permission removed from a group
Workflow changes	scheme created, scheme copied, scheme removed, scheme edited, scheme assigned to a project, scheme unassigned from a project, workflow created, workflow copied, workflow removed, workflow renamed, workflow draft published
Notification changes	scheme created, scheme copied, scheme removed, scheme edited, scheme added to project, scheme removed from project, notification added to scheme, notification removed from scheme
Screen changes	scheme created, scheme copied, scheme removed, scheme edited, scheme added to project, scheme removed from project, screen added to scheme, screen removed from scheme, screen field configuration changed
Custom field changes	custom field created, custom field updated, custom field removed, scheme added to project, scheme removed from project
Workflow changes	Status added, status updated, status deleted, transition added, transition updated, transition deleted

Notes:

- The audit log cannot be sorted. Try exporting the data and opening it in a spreadsheet to manipulate the data.

Hiding external directory user events (LDAP/Crowd events)

By default, the audit log will display all recorded events. However, you can choose to hide external directory user events (those triggered by LDAP or Crowd) from view. These events are still recorded, and will still be available for export.

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose  **> System > Audit Log**.
3. Select **Actions > Audit Log Settings**.
4. Check the **Hide events from external user directories** checkbox to hide the user events.

Modifying the audit log retention period

Auditing is always enabled in JIRA applications. However, you can configure how long audit events are retained.

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose 
 - > **System > Audit Log.**
3. Select **Actions > Audit Log Settings.**
4. Choose your retention period.

Exporting the audit log

You can export the audit log as a text file. When you export the audit log, all the events are included in the export, even if you currently have filtered the audit log results in the page.

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose 
 - > **System > Audit Log.**
3. Select **Export.**

Auditing and the REST API

The audit log can also be accessed via the REST API. You may use this to:

- Export the audit log
- Add events to the audit log triggered by external plugins

For more information on using the REST API, please refer to the JIRA REST documentation for your appropriate version of JIRA within the developer documentation [here](#).

Important directories and files

On this page:

- JIRA installation directory
 - Important files and directories
 - <jira-application-dir>/atlassian-jira/WEB-INF/classes/jira-application.properties
 - <jira-application-dir>/atlassian-jira/WEB-INF/classes/jpm.xml
 - <jira-application-dir>/atlassian-jira/WEB-INF/lib/
 - <jira-application-dir>/atlassian-jira/WEB-INF/classes/log4j.properties
 - <jira-application-dir>/atlassian-jira/WEB-INF/classes/entityengine.xml
 - <jira-application-dir>/
 - conf/server.xml
 - logs/atlassian-jira-gc-timestamp.log
 - Memory settings
- JIRA home directory
- Important files
 - dbconfig.xml
 - jira-config.properties
- Important subdirectories
 - data
 - export
 - log
 - plugins
 - caches
 - tmp

JIRA installation directory

.nuThe JIRA installation directory is the directory into which the JIRA application files and libraries have been extracted, either:

- by the [Windows](#) installer, or
- by the [Linux](#) installers

JIRA does not modify or store any data in this directory.

Important files and directories

The directories/files described below are found under different sub-directories of the 'JIRA Installation Directory', depending on whether you have installed a recommended Windows, Linux or Archive JIRA. Please substitute the following directories for the `<jira-application-dir>` placeholder (used throughout the rest of this section), as follows:

- **'Recommended' distributions** — the `atlassian-jira` subdirectory of the 'JIRA Installation Directory' installed using the 'Windows Installer' and 'Linux Installer', and there associated installations from archive files (.zip and tar.gz respectively).
- The default installation directory on Linux is:

```
/opt/atlassian/jira/
```

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/jira-application.properties
```

This file tells JIRA where to find the [JIRA application home directory](#).

⚠ Be aware that your JIRA home directory defined in this file can be overridden. See [Setting your JIRA application home directory](#) for more information.

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/jpm.xml
```

This file stores the default values for [JIRA's advanced configuration settings](#) and should not be modified. The default values of properties in this file are customized (i.e. overridden) by redefining them in either the `jira-config.properties` file (in your [JIRA application home directory](#)) or the JIRA database (via the JIRA administration area). See [Advanced JIRA configuration](#) for more information.

```
<jira-application-dir>/atlassian-jira/WEB-INF/lib/
```

This is the directory where plugins built on Atlassian's Plugin Framework 1 (i.e. 'Plugins 1' plugins) are stored. If you are [installing a new 'Plugins 1' plugin](#), you will need to deploy it into this directory. 'Plugins 2' plugins should be stored in the [JIRA application home directory](#).

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/log4j.properties
```

JIRA's logging configuration file. See [Logging and profiling](#).

The actual log files generated by JIRA can be found in the following locations:

- **JIRA application log** — `bin/atlassian-jira.log`
- **Application server log** — generally the application server log file can be found under the `logs` directory. However, this can vary depending on the application server you are running.

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/entityengine.xml
```

This file configures the OFBiz Entity Engine, which JIRA uses to store persist data in a data source.

```
<jira-application-dir>/
```

This file includes garbage collection (GC) logs that are used to improve the performance of JIRA applications. The log statements indicate when Java is collecting garbage, how long this process takes, and how much memory has been freed. The file is generated automatically, and is also included in the Support Zip package. For more info, see [Using garbage collection logs](#).

The sub-directories/files described below are found under the root of the JIRA application installation directory.

```
conf/server.xml
```

This file is used for JIRA SSL configuration. See [Running JIRA applications over SSL or HTTPS](#).

```
logs/atlassian-jira-gc-timestamp.log
```

These files include garbage collection (GC) logs that can be used to monitor the performance of JIRA applications. The log statements indicate when Java is collecting garbage, how long this process takes, and which resources can be freed. The files are created automatically, and then overwritten if the maximum number of files (5) is reached. The timestamp indicates when the JIRA session related to the logs was started. For more info, see [Using garbage collection logs](#).

Memory settings

The file used to edit JAVA_OPTS memory settings will depend on the method used to install JIRA, as well as the operating system used for your installation.

For example, if you are running JIRA on Tomcat in Windows (manual startup), you would update the following file:

```
bin\setenv.bat
```

whereas for JIRA on Tomcat in Linux/Unix, you would update this file:

```
bin/setenv.sh
```

See [Increasing JIRA memory](#) for further details.

JIRA home directory

The JIRA home directory contains key data that help define how JIRA works. This document outlines the purpose of the various files and subdirectories within the JIRA home directory.

If JIRA was installed using the automated [Windows](#) or [Linux](#) installers, the default location of the JIRA home directory is:

- C:\Program Files\Atlassian\Application Data\JIRA (on Windows) or
- /var/atlassian/application-data/jira (on Linux)

If you install JIRA from an archive file, the JIRA home directory can be any suitable location that is accessible by your JIRA installation. Typical example locations might be:

- C:\jira\home (on Windows) or
- /var/jira-home (on Linux or Solaris)

 However, avoid locating the JIRA home directory inside the [JIRA application installation directory](#).

 For information on specifying the location of the JIRA home directory, please see [Setting your JIRA application home directory](#).

Important files

```
dbconfig.xml
```

This file (located at the root of your JIRA home directory) defines all details for JIRA's database connection. This file is typically created by running the [JIRA setup wizard](#) on new installations of JIRA or by configuring a database connection using the [JIRA configuration tool](#).

You can also create your own `dbconfig.xml` file. This is useful if you need to specify additional parameters for your specific database configuration, which are not generated by the setup wizard or JIRA configuration tool. For more information, refer to the 'manual' connection instructions of the appropriate database configuration guide in [Connecting JIRA to a database](#).

```
jira-config.properties
```

This file (also located at the root of your JIRA home directory) stores custom values for most of [JIRA's advanced configuration settings](#). Properties defined in this file override the default values defined in the `jpm.xml` file (located in your JIRA application installation directory). See [Advanced JIRA configuration](#) for more information.

i In new JIRA installations, this file may not initially exist and if so, will need to be created manually. See [Making changes to the `jira-config.properties` file](#) for more information. This file is typically present in JIRA installations upgraded from version 4.3 or earlier, whose [advanced configuration options](#) had been customized (from their default values).

Important subdirectories

data

This directory contains application data for your JIRA instance, including attachments (for every version of each attachment stored in JIRA).

export

JIRA will place its [automated backup archives](#) into this directory.

log

JIRA will place its logs into this directory. (Note: if the JIRA home directory is not configured, then the logs will be placed into the current working directory instead).

The logs will only start showing up once the first log message is written to them. For example, the internal access log will not be created until JIRA starts writing to it.

You can change the location of the log file using `log4j.properties` as described in the documentation on [Logging and profiling](#).

plugins

This is the directory where plugins built on [Atlassian's Plugin Framework 2](#) (i.e. 'Plugins 2' plugins) are stored. If you are [installing a new 'Plugins 2' plugin](#), you will need to deploy it into this directory under the `installed-plugins` sub-directory.

'Plugins 1' plugins should be stored in the [JIRA application installation directory](#).

This directory is created on JIRA startup, if it does not exist already.

caches

This is where JIRA stores caches including:

- Lucene indexes - see [Searching, Indexing, and filters troubleshooting](#)
- OSGi framework caches

These files are vital for JIRA performance and should not be modified or removed externally while JIRA is running.

See [Search indexing](#) for further details.

tmp

Any temporary content created for various runtime functions such as exporting, importing, file upload and indexing is stored under this directory.

You can remove files from this directory while JIRA is running, but we recommend that you shut down JIRA first before altering the contents of this directory.

JIRA application installation directory

The JIRA installation directory is the directory into which the JIRA application files and libraries have been extracted, either:

- by the [Windows](#) installer, or
- by the [Linux](#) installers

JIRA does not modify or store any data in this directory.

Important files and directories

The directories/files described below are found under different sub-directories of the 'JIRA Installation Directory', depending on whether you have installed a recommended Windows, Linux or Archive JIRA. Please substitute the following directories for the `<jira-application-dir>` placeholder (used throughout the rest of this section), as follows:

- **'Recommended' distributions** — the `atlassian-jira` subdirectory of the 'JIRA Installation Directory' installed using the 'Windows Installer' and 'Linux Installer', and there associated installations from archive files (.zip and tar.gz respectively).
- The default installation directory on Linux is:

```
/opt/atlassian/jira/
```

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/jira-application.properties
```

This file tells JIRA where to find the [JIRA application home directory](#).

⚠ Be aware that your JIRA home directory defined in this file can be overridden. See [Setting your JIRA application home directory](#) for more information.

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/jpm.xml
```

This file stores the default values for [JIRA's advanced configuration settings](#) and should not be modified. The default values of properties in this file are customized (i.e. overridden) by redefining them in either the `jira-config.properties` file (in your [JIRA application home directory](#)) or the JIRA database (via the JIRA administration area). See [Advanced JIRA configuration](#) for more information.

```
<jira-application-dir>/atlassian-jira/WEB-INF/lib/
```

This is the directory where plugins built on Atlassian's Plugin Framework 1 (i.e. 'Plugins 1' plugins) are stored. If you are [installing a new 'Plugins 1' plugin](#), you will need to deploy it into this directory. 'Plugins 2' plugins should be stored in the [JIRA application home directory](#).

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/log4j.properties
```

JIRA's logging configuration file. See [Logging and profiling](#).

The actual log files generated by JIRA can be found in the following locations:

- **JIRA application log** — `bin/atlassian-jira.log`
- **Application server log** — generally the application server log file can be found under the `logs` directory. However, this can vary depending on the application server you are running.

```
<jira-application-dir>/atlassian-jira/WEB-INF/classes/entityengine.xml
```

This file configures the OFBiz Entity Engine, which JIRA uses to store persist data in a data source.

```
<jira-application-dir>/
```

This file includes garbage collection (GC) logs that are used to improve the performance of JIRA applications. The log statements indicate when Java is collecting garbage, how long this process takes, and how much memory has been freed. The file is generated automatically, and is also included in the Support Zip package. For more info, see [Using garbage collection logs](#).

The sub-directories/files described below are found under the root of the JIRA application installation directory.

```
conf/server.xml
```

This file is used for JIRA SSL configuration. See [Running JIRA applications over SSL or HTTPS](#).

logs/atlassian-jira-gc-timestamp.log

These files include garbage collection (GC) logs that can be used to monitor the performance of JIRA applications. The log statements indicate when Java is collecting garbage, how long this process takes, and which resources can be freed. The files are created automatically, and then overwritten if the maximum number of files (5) is reached. The timestamp indicates when the JIRA session related to the logs was started. For more info, see [Using garbage collection logs](#).

Memory settings

The file used to edit JAVA_OPTS memory settings will depend on the method used to install JIRA, as well as the operating system used for your installation.

For example, if you are running JIRA on Tomcat in Windows (manual startup), you would update the following file:

bin\setenv.bat

whereas for JIRA on Tomcat in Linux/Unix, you would update this file:

bin/setenv.sh

See [Increasing JIRA memory](#) for further details.

JIRA application home directory

The JIRA home directory contains key data that help define how JIRA works. This document outlines the purpose of the various files and subdirectories within the JIRA home directory.

If JIRA was installed using the automated [Windows](#) or [Linux](#) installers, the default location of the JIRA home directory is:

- C:\Program Files\Atlassian\Application Data\JIRA (on Windows) or
- /var/atlassian/application-data/jira (on Linux)

If you install JIRA from an archive file, the JIRA home directory can be any suitable location that is accessible by your JIRA installation. Typical example locations might be:

- C:\jira\home (on Windows) or
- /var/jira-home (on Linux or Solaris)

 However, avoid locating the JIRA home directory inside the [JIRA application installation directory](#).

 For information on specifying the location of the JIRA home directory, please see [Setting your JIRA application home directory](#).

Important files

dbconfig.xml

This file (located at the root of your JIRA home directory) defines all details for JIRA's database connection. This file is typically created by running the [JIRA setup wizard](#) on new installations of JIRA or by configuring a database connection using the [JIRA configuration tool](#).

You can also create your own *dbconfig.xml* file. This is useful if you need to specify additional parameters for your specific database configuration, which are not generated by the setup wizard or JIRA configuration tool. For more information, refer to the 'manual' connection instructions of the appropriate database configuration guide in [Connecting JIRA to a database](#).

jira-config.properties

This file (also located at the root of your JIRA home directory) stores custom values for most of [JIRA's advanced configuration settings](#). Properties defined in this file override the default values defined in the *jpm.xml* file (located in your JIRA application installation directory). See [Advanced JIRA configuration](#) for more information.

 In new JIRA installations, this file may not initially exist and if so, will need to be created manually. See [Making changes to the jira-config.properties file](#) for more information. This file is typically present in JIRA installations upgraded from version 4.3 or earlier, whose [advanced configuration options](#) had been customized (from their default values).

Important subdirectories

data

This directory contains application data for your JIRA instance, including attachments (for every version of each attachment stored in JIRA).

export

JIRA will place its [automated backup archives](#) into this directory.

log

JIRA will place its logs into this directory. (Note: if the JIRA home directory is not configured, then the logs will be placed into the current working directory instead).

The logs will only start showing up once the first log message is written to them. For example, the internal access log will not be created until JIRA starts writing to it.

You can change the location of the log file using `log4j.properties` as described in the documentation on [Logging and profiling](#).

plugins

This is the directory where plugins built on [Atlassian's Plugin Framework 2](#) (i.e. 'Plugins 2' plugins) are stored. If you are [installing a new 'Plugins 2' plugin](#), you will need to deploy it into this directory under the `installed-plugins` sub-directory.

'Plugins 1' plugins should be stored in the [JIRA application installation directory](#).

This directory is created on JIRA startup, if it does not exist already.

caches

This is where JIRA stores caches including:

- Lucene indexes - see [Searching, Indexing, and filters troubleshooting](#)
- OSGi framework caches

These files are vital for JIRA performance and should not be modified or removed externally while JIRA is running.

See [Search indexing](#) for further details.

tmp

Any temporary content created for various runtime functions such as exporting, importing, file upload and indexing is stored under this directory.

You can remove files from this directory while JIRA is running, but we recommend that you shut down JIRA first before altering the contents of this directory.

Setting your JIRA application home directory

The [JIRA application home directory](#) contains key data that help define how JIRA works. You must have a JIRA home directory specified for your JIRA instance before you can start it. This document describes how to specify the location of the JIRA home directory for your JIRA instance.

One JIRA home per JIRA instance

You can only have one JIRA home directory per JIRA installation. If you have multiple JIRA installations, you will need to set up a JIRA home directory for each installation. A lock is placed at the root level of a JIRA home directory when it is created to ensure that it can only be used by one JIRA installation.

On this page:

- How do I set my JIRA home?
- What location should I specify for my JIRA home?
- How do I change my JIRA home?
- What is stored in the JIRA home directory?
- Notes

You only need to specify the location of the root directory for your JIRA home. The sub-directories will be created automatically when JIRA is started or when you use a function in JIRA that requires a particular sub-directory.

How do I set my JIRA home?

There are a few methods available for specifying the location of your [JIRA application home directory](#) in JIRA. However, please be aware of the [notes below](#) before you specify this location.

Recommended Methods

The recommended methods for specifying the location of your JIRA home directory in JIRA are to:

- Use the [JIRA configuration tool](#) to change the location of your JIRA home directory.
- Edit the `jira-application.properties` file and set the value of the 'jira.home' property to the desired location for your [JIRA home directory](#) (this location should be something different than the application directory, or you may run into problems later). If you are specifying this location's path on Windows, use double back-slashes ("\") between subdirectories. For example, `X:\path\to\JIRA\Home`.
 -  If you define an UNC path in Microsoft Windows, be sure to double escape the leading backslash: `\\machinename\path\to\JIRA\home`
 -  See the [JIRA installation directory](#) page to find where this file is located.
- Set an environment variable named `JIRA_HOME` in your operating system whose value is the location of your [JIRA home directory](#). To do this:
 - On Windows, do one of the following:
 - Configure this environment variable through the Windows user interface (typically through 'My Computer' or 'Computer')
 - At the command prompt, enter the following command (with your own JIRA Home path) before running JIRA from the command prompt:
 - `set JIRA_HOME=X:\path\to\JIRA\Home`
 -  Please set your `JIRA_HOME` environment variable value using this format, where:
 - `x` is the drive letter where your JIRA Home Directory is located and
 - no spacing has been added around the equal sign ('=')
 - Specify the command above in a batch file used to start JIRA.
 - On Linux/Solaris, do one of the following:
 - Enter the following command at a shell/console prompt (with your own JIRA Home path) before running JIRA:
 - `export JIRA_HOME=/path/to/jira/home`
 - Specify the command above in a script used to start JIRA.

⚠ Please note: If you have specified different values for a 'jira.home' property in the `jira-application.properties` file and a `JIRA_HOME` environment variable, the value of the `JIRA_HOME` environment variable takes precedence.

Alternative method

Alternatively, you can specify the location of your JIRA home directory as property within your application server:

- Configure a new web context property called 'jira.home' for your application server. To do this, you need to define this web context property inside a `<parameter/>` element (as a child of the `<context/>` element) in your `server.xml` file.
 - i** The `server.xml` file is located within the `conf` subdirectory of your JIRA application installation directory.

```
<Context ...>
...
  <Parameter name="jira.home" value="c:/jira/home"/>
...
</Context>
```

⚠ Please note: A 'jira.home' web context property defined in your application server overrides the value of the 'jira.home' property defined in your `jira-application.properties` file. However, a `JIRA_HOME` environment variable defining your JIRA home directory will override either of these 'jira.home' values.

What location should I specify for my JIRA home?

You can specify any location on a disk for your JIRA home directory. Please be sure to specify an absolute path.

Please note that you cannot use the same JIRA home directory for multiple instances of JIRA. We recommend locating your JIRA Home Directory completely independently of the [JIRA installation directory](#) (i.e. not nesting one within the other) as this will minimize information being lost during major operations (e.g. backing up and restoring instances).

How do I change my JIRA home?

1. Set your JIRA home to the new location, using your preferred method as described in "[How do I set my JIRA home?](#)" (above).
2. Restart JIRA.

What is stored in the JIRA home directory?

The following page describes the data stored in the JIRA home directory: [JIRA application home directory](#).

Notes

- If you are using the Windows installer, you do not need to configure the JIRA home directory separately, as you will be prompted to specify this location during the installation process.
- The JIRA installer may not be able to create the home due to permission problems. If this is the case, please see [JIRA is Unable to Start due to Could not create necessary subdirectory](#).

Integrating JIRA applications with a Web server

The following pages contain information on integrating JIRA applications with a web server.

- [Integrating JIRA applications with IIS](#)
- [Integrating JIRA with Apache](#)

Integrating JIRA applications with IIS

The content on this page relates to platforms that are not supported by JIRA. Consequently, Atlassian c

an not guarantee providing any support for it. Please be aware that this material is provided for your information only, and using it is done so at your own risk.

This page describes how to configure Microsoft's IIS web server and JIRA such that IIS forwards requests on to JIRA, and responses back to the user. This is useful if you already have IIS running serving web pages (e.g. <http://mycompany.com>), and wish to integrate JIRA as just another URL (e.g. <http://mycompany.com/jira>).

JIRA is written in Java, and needs a Java Application Server (servlet container) to run. As IIS does not provide services of a Java Application Server, it is not possible to deploy JIRA directly into IIS. It is possible, however, to configure IIS to proxy requests for JIRA to an application server where JIRA is deployed. Therefore, if your main website is running in IIS, it is possible to integrate JIRA into this website.

If you need to integrate JIRA with IIS, JIRA needs to be deployed into a Java application server (such as [Apache Tomcat](#)), which provides IIS integration capability.

If you are running JIRA against an application server other than Apache Tomcat, please consult that application server's documentation to determine whether it is possible (and how) to integrate the application server with IIS.

To integrate JIRA with IIS you will need to:

1. [Configure JIRA and test that it works on its own](#)
2. [Configure Tomcat to accept proxied requests from IIS](#)
3. [Configure IIS to forward JIRA requests to Tomcat](#)

1. Configure JIRA

1. Follow [the JIRA installation guide](#) to install and configure JIRA. Note that JIRA can be installed on the same machine as IIS, but this is not necessary.
2. Change the context path of the JIRA web application:

To allow IIS to proxy requests to JIRA, JIRA web application must be deployed with a context path (e.g. the `/jira` in <http://localhost:8080/jira> (http://localhost:8080*/jira*)) in Tomcat. The context path **must** be set to the path in the URL that IIS will use to proxy requests. For example, if your website is running with address www.example.com in IIS, and you would like to make JIRA available under www.example.com/jira, you will need to set JIRA's context path to `/jira` in Tomcat.

To do this, edit the `conf/server.xml` file. Change the `path` attribute of the `Context` element to `/jira`.
3. Restart JIRA after changing the context path.
4. Set the **'Base URL'** to include the context path (see [Configuring JIRA options](#)).
5. Turn JIRA's GZip compression **OFF** (since there will be no benefit from GZip compression once proxying is implemented).
6. Test that JIRA works correctly by pointing your web browser directly at Tomcat (e.g. <http://localhost:8080/jira>) and going through JIRA's Setup Wizard. If you have completed the Setup Wizard previously, try creating an issue or editing one. Please ensure that no errors occur.

2. Configure Tomcat to accept proxied requests

HTTP/1.1 Connector

If you are using the HTTP/1.1 Connector, you will need to add the following attributes to the Connector port in Tomcat's `server.xml`:

```
proxyName="mycompany.com" proxyPort="80"
```

Please refer to the [Integrating JIRA with Apache](#) for reference.

1. Enable **AJP/1.3 Connector** in Tomcat: To allow Tomcat to accept requests for JIRA from IIS, edit the `conf/server.xml` file and ensure that the **AJP/1.3 Connector** is enabled (i.e. *not* commented out). To enable the AJP/1.3 Connector in a JIRA remove the comment symbols around the following section in the `conf/server.xml` file:

```
<Connector port="8009" enableLookups="false" redirectPort="8443"
protocol="AJP/1.3" />
```

The above example configures Tomcat to listen for proxied IIS requests on port 8009. If this port is already in use on the machine where JIRA is running, please change to another port.

- Restart Tomcat and ensure that no errors regarding used ports appear in the logs or in the Tomcat Console.
- Ensure that the AJP Connector is listening on the specified port (8009 by default). One way to do this is to use the "netstat -na" command in the command window and see if port 8009 is listed in the output:

```

C:\dev\jira\atlassian-jira-enterprise-3.3-standalone\bin>startup.bat
Using CATALINA_BASE:   C:\dev\jira\atlassian-jira-enterprise-3.3-standalone
Using CATALINA_HOME:   C:\dev\jira\atlassian-jira-enterprise-3.3-standalone
Using CATALINA_TMPDIR: C:\dev\jira\atlassian-jira-enterprise-3.3-standalone\temp
Using JAVA_HOME:       C:\j2sdk1.4.2_02
C:\dev\jira\atlassian-jira-enterprise-3.3-standalone\bin>netstat -na | findstr 8009
TCP        0.0.0.0:8009          0.0.0.0:*           LISTENING
C:\dev\jira\atlassian-jira-enterprise-3.3-standalone\bin>

```

3. Configure IIS to forward requests to JIRA

On the machine where IIS is deployed:

- Download the ISAPI Redirect DLL from the [Apache site](#). When downloading, choose the version of Windows that IIS is running on (either win32 or win64), and then **choose the latest available jk version**.

The file to download is named **isapi_redirect_X.X.X.dll**, where 'X.X.X' is the version number. You will need to remove the version number from the DLL file (i.e. it needs to be named isapi_redirect.dll).

- Place the DLL and the associated properties files in an installation directory. For the purpose of this document, we will assume the directory is C:\tomcat_iis_connector. Place the **isapi_redirect.dll** in this directory. Then download the **isapi_redirect.properties** file and place this in the same directory as the **isapi_redirect.dll** file.
- Create a directory called 'conf' in your installation directory (C:\tomcat_iis_connector\conf). Download the files **uriworkermap.properties** and **workers.properties.minimal** and place them in the C:\tomcat_iis_connector\conf directory.
- Create a directory called 'logs' (C:\tomcat_iis_connector\logs). This is where the logs associated with the **isapi_redirect.dll** execution will be placed.
- In the "C:\tomcat_iis_connector" directory you may need to modify the **isapi_redirect.properties** file. The **isapi_redirect.properties** file tells the connector where to find its configuration files and where the DLL can be found in relation to the IIS server. There are 5 properties in this file:
 - extension_uri** — the path to the virtual directory that contains the **isapi_redirect.dll**
 - log_file** — the path to write the log file to
 - log_level** — the level at which the logs should be generated

- d. `worker_file` — the path to your `workers.properties.minimal` file in your installation
 - e. `worker_mount_file` — the path to your `uriworkermap.properties1` file in your installation.
If you are installing the connector in `C:\tomcat_iis_connector` and you follow the instructions below about setting up the virtual directory for the `isapi_redirect.dll`, then you should not have to change any properties in the provided file.
6. In the "`C:\tomcat_iis_connector\conf`" directory you may need to modify the `uriworkermap.properties` and the `workers.properties.minimal` files.

The provided files contain the changes mentioned here and should work if you completely follow this document. **If you have deviated from this document, then you will need to modify these files as described below.**

The `workers.properties.minimal` file tells IIS where (IP address and port) Tomcat is running. The `uriworkermap.properties` tells IIS what requests to proxy to Tomcat.

To edit these files:

- a. Edit the `uriworkermap.properties` and ensure that it contains the following mapping for JIRA. You do not need any other mappings.

```
/jira/*=worker1
```

The mapping (e.g. `/jira/`) ***must** be the same as the context path that JIRA has been deployed with in Tomcat as described in the [Configure JIRA](#) section of this document.

- b. Edit the `workers.properties.minimal` file and modify the `worker.ajp13w.host` property if necessary. This property should be set to the host name or the IP address of the machine where Tomcat (with JIRA) is running. If Tomcat is running on the same machine as IIS then you can leave the property set to `localhost`. If you have specified a host name as the value of this property, please ensure that the IIS machine can correctly resolve it to the appropriate IP address.
- c. If you have modified the port for the AJP Connector you will need to modify the `worker.ajp13w.port` property. Here is an example of the file with Tomcat running on the same machine as IIS and using the default port (8009) for AJP:

```
worker.list=worker1

#
# Defining a worker named worker1 and of type ajp13.
# Note that the name and the type do not have to match.
#
worker.worker1.type=ajp13
worker.worker1.host=localhost
worker.worker1.port=8009
```

- 7. Open **Control Panel**, then **Administrative Tools** and open **Internet Information Services**.
- 8. **IIS 7.0 only:** If you are using **IIS 7.0**, you will need to install two required service roles, ISAPI Extensions and ISAPI Filters:
 - a. Navigate to Start Menu > All Programs > Administration Tools > Service Manager.
 - b. Select 'Web Server (IIS)' in Server Manager > Roles.
 - c. Click 'Add Role Services' and follow the Wizard.
- 9. Add an **ISAPI Filter** to IIS, as described below:
 - **IIS 6.0 or earlier:**
 - a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), and click on **Properties**.
 - b. Click the **ISAPI Filters** tab.
 - c. Check if there is a Filter that points to the `isapi_redirect.dll` file and that it is in the right location. If not, click **Add** and create one. Enter `tomcat` as the Filter Name and enter

the location of the `isapi_redirect.dll` file for the executable.

d. Click **Apply** and then **OK**.

- **IIS 7.0:**

a. Click the **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), and click on **ISAPI Filters**.

b. Click the **ISAPI Filters** icon.

c. Check if there is a Filter that points to the `isapi_redirect.dll` file and that it is in the right location. If not, click **Add** and create one. Enter `tomcat` as the Filter Name and enter the location of the `isapi_redirect.dll` file.

d. Click **OK**.

10. Create a **virtual directory** for JIRA in IIS.

a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), choose **New** and then **Virtual Directory**.

b. Go through the creation wizard. Set the `alias` as the value of the Context Path (without slashes) that was set in the [Configure JIRA](#) section of this document (see above). In our example this is `java`.

c. This can point to any directory.

d. Complete the wizard.

The reason for creating a virtual directory is so that requests without the trailing slash still work. For example, if you are deploying JIRA under `http://www.example.com/jira/` without the virtual directory, then requests to `http://www.example.com/jira` will fail.

11. Create a **virtual directory** for access to the `isapi_redirect.dll` in IIS, as described below:

- **IIS 6.0 or earlier:**

a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), choose **New** and then **Virtual Directory**.

b. Go through the creation wizard. Set the `alias` to be `jakarta`.

c. This must point to the directory in which the `isapi_redirect.dll` is installed. In our example this is `C:\tomcat_iis_connector`.

d. Complete the wizard, making sure that you grant the 'Execute' permission for the **Virtual Directory** by checking the 'Execute' checkbox.

- **IIS 7.0:**

a. Right-click on **Default Web Site** (or the Web Site that should be responsible for proxying requests to JIRA), and choose **Add Virtual Directory**.

b. Set the `alias` to be `jakarta`.

c. **Physical Path** must point to the directory in which the `isapi_redirect.dll` is installed. In our example this is `C:\tomcat_iis_connector`.

d. Click the 'jakarta' Virtual Directory and double-click 'Handler Mappings'.

e. Click 'Edit Feature Permissions' in the Action panel on the right-hand side.

f. Check the 'Execute' permission checkbox.

This Virtual Directory is needed for the connector to work. The alias that you give the directory needs to be the same as the path set in the `isapi_redirect.properties` file, `extension_uri` property. In our example this value is: `/jakarta/isapi_redirect.dll`.

12. If using IIS 6.0 or 7.0, you will need to add the dll as a **Web Service Extension**, as described below.

- **IIS 6.0:**

a. Right-click on **Web Service Extensions** and choose **Add a new Web Service Extension...**

b. Enter `tomcat` for the **Extension Name** and then add the `isapi_redirect.dll` file to the required files.

c. Select the **Set extension status to Allowed** checkbox, then click **OK**.

- **IIS 7.0:**

a. Navigate to the servers and highlight your server.

b. Navigate to 'ISAPI and CGI Restrictions'.

c. Add and allow the `isapi_redirect.dll` extension.

13. You will need to restart the IIS Service. To do this, browse to **Control Panel**, click **Administrative Tools**, click on **Services**, find the IIS Admin Service and click **restart**.

14. You are done! To test the configuration, point your web browser at IIS and append JIRA's context path to the URL. For example, if your website is running under the address of <http://www.example.com> and you have deployed JIRA with the context path of `jira`, point your browser at <http://www.example.com/jira>.

Troubleshooting

- **Whenever I go to JIRA in my browser, a login panel pops up. I enter a valid username and password for JIRA, but the panel pops up again.** Make sure that you have Anonymous Access set on the `jira` virtual directory in IIS. It will be set to that if you have followed the above instructions. To check this:
 1. In **'Internet Information Services'**, right click the `jira` virtual directory and choose **'Properties'**.
 2. Click the **'Directory Security'** tab.
 3. Click the **'Edit...'** button in the **'Anonymous access and authentication control'** section.
 4. Make sure that the **'Anonymous access'** tick box is selected, and make sure that nothing is selected in the **'Authenticated access'** section. Do not select **'Basic authentication'**. Do not select **'Integrated Windows authentication'**.
- **Whenever I go to JIRA in Internet Explorer, a login panel pops up. I enter a valid username and password for JIRA, but the panel pops up again. This doesn't happen, however, in another browser such as Firefox or Safari. I can successfully log in to JIRA in those browsers.** Make sure that you have Internet Explorer's **User Authentication** set to **Anonymous login**. To check this:
 1. In Internet Explorer, click the **'Tools'** menu and select **'Internet Options'**.
 2. Click the **'Security'** tab.
 3. Select the security zone that the JIRA server is in.
 4. Click the **'Custom level...'** button.
 5. Scroll right down to the bottom to the **'User Authentication'** section.
 6. Select **'Anonymous logon'** (if it is not already selected).
 7. Click the **'OK'** button on this screen, and again on the next screen.
 8. Restart Internet Explorer.
- **When I try to navigate to my JIRA instance at <http://localhost/jira> in my browser, it prompts me to download a file with nonsensical information, rather than showing me my JIRA instance.** Make sure that you have granted the 'Execute' permission to your Virtual Directory for JIRA in IIS. See step 11 of the **'3. Configure IIS to forward requests to JIRA'** section in this document for detailed instructions.

Known issues

- **64 bit IIS:** If you are running a 64 bit OS, please use a [64 bit version of the Tomcat IIS connector](#).
- **Customer submitted solution:** If you must use a 32 bit IIS connector, you can do so by clicking `Application Pools > Advanced Settings > Allow 32bit applications`.
- **Customer submitted solution:** You need to set the ISAPI extension on the website.

Integrating JIRA with Apache

Error rendering macro 'viewport-redirect' : null

Atlassian applications allow the use of reverse-proxies within our products, however Atlassian Support does not provide assistance for configuring them. Consequently, Atlassian **can not guarantee providing any support for them.**

If assistance with configuration is required, please raise a question on [Atlassian Answers](#).

This page describes how to integrate [Apache HTTP Server](#) (also referred to as `httpd`) with JIRA, utilizing `mod_proxy` so that Apache operates as a reverse-proxy over HTTP. If HTTPS configuration is required, please see our [Integrating JIRA with Apache using SSL](#) documentation. Configuring Apache allows for running JIRA on non-standard HTTP port (such as 8080) and users will be able to access JIRA over standard HTTP as their traffic will be routed through the proxy.

Apache can be configured to allow access to JIRA in any of the following methods:

- Directly on its own domain: <http://jira.com>
- As a subdomain of another domain: <http://jira.atlassian.com>
- It can also be accessed on a context path on either a domain or subdomain: <http://atlassian.com/jira>

This documentation will cover a straightforward implementation of `mod_proxy` using the above three configurations. If a more complicated solution is required, refer to the [Apache HTTP Server Version Documentation](#), consult with the Apache SME within your organization, and if need be raise a question on [Atlassian Answers](#), or get in touch with one of our [Atlassian Experts](#).

▼ [Expand for an example of a common Apache configuration](#)

1. JIRA is running on port 8080 on a server within the LAN that cannot be accessed externally (the router/firewall is not forwarding port 8080 to it).
2. Apache is set up on another server (or the same server as JIRA) that can be accessed externally on HTTP (80).
3. Apache is then accessed over HTTP on the appropriate URL (`VirtualHost`), routing the traffic to and from the JIRA server.

On this page:

- [Step 1: Configure Tomcat](#)
- [Step 2: Configure Apache HTTP Server](#)
 - [2.1 Enable the Proxy Modules](#)
 - [2.2. Configure Apache to use those Modules](#)
- [Step 3: Configure JIRA](#)
- [Troubleshooting](#)
- [See also](#)

Step 1: Configure Tomcat

1. Stop JIRA.
2. Edit Tomcat's `server.xml` to include the required JIRA context path. The below example uses `path="jira"` - this means JIRA is accessible on <http://jiraserver:8080/jira> given the default JIRA port is used.

This step is only required if JIRA will be accessed on a context path, for example <http://atlassian.com/jira>. If this is not required, this step can be skipped.

```

<Engine defaultHost="localhost" name="Catalina">
    <Host appBase="webapps" autoDeploy="true"
name="localhost" unpackWARs="true">
        <Context
docBase="\${catalina.home}/atlassian-jira" path="/jira"
reloadable="false" useHttpOnly="true">

                <!--

=====
=====
                Note, you no longer configure your database
driver or connection parameters here.
                These are configured through the UI during
application setup.

=====
=====
                -->
                <Resource auth="Container"
factory="org.objectweb.jotm.UserTransactionFactory"
jotm.timeout="60" name="UserTransaction"
type="javax.transaction.UserTransaction"/>
                <Manager pathname="" />
        </Context>
    </Host>

```

i Ensure the path value is set with a prepping forward slash (/). For example, path="/jira" rather than path="jira".

3. Edit Tomcat's server.xml to include a separate connector to proxy the requests. This requires the proxyName & proxyPort attributes. Replace them with the appropriate domain and port of the proxy, as in the below example:

```

<Service name="Catalina">

    <!-- Apache Proxy Connector -->
        <Connector acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" enableLookups="false"
maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
port="8080" protocol="HTTP/1.1" redirectPort="8443"
useBodyEncodingForURI="true"
                proxyName="jira.atlassian.com" proxyPort="80"/>

    <!-- Standard HTTP Connector -->
        <Connector acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" enableLookups="false"
maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
port="8081" protocol="HTTP/1.1" redirectPort="8443"
useBodyEncodingForURI="true"/>

```

4. Start JIRA.
5. Test that JIRA is accessible on the normal connector, using a context path if applicable - for example <http://jiraserver:8081/jira>.
6. Test that the new connector is in effect by accessing JIRA on the appropriate proxy connector. The

behavior varies depending on the context path:

- a. If the context path is empty or root (/), visiting JIRA via the proxy connector (e.g. `http://jiraserver:8080/`) should take you to JIRA with a warning:

We've detected a potential problem with JIRA's Dashboard configuration that your administrator can correct. [Hide](#)

Dashboard Diagnostics: Mismatched URL Hostname

JIRA is reporting that it is running on the hostname 'jira.atlassian.com', which does not match the hostname used to run these diagnostics, 'localhost'. This is known to cause JIRA to construct URLs using the incorrect hostname, which will result in errors in the dashboard, among other issues.

The most common cause of this is the use of a reverse-proxy HTTP server (often Apache or IIS) in front of the application server running JIRA. While this configuration is supported, some additional setup might be necessary in order to ensure that JIRA detects the correct hostname.

The following articles describe the issue and the steps you should take to ensure that your web server and app server are configured correctly:

- Gadgets do not display correctly after upgrade to JIRA 4.0
- Integrating JIRA with Apache
- Integrating JIRA with Apache using SSL

If you believe this diagnosis is in error, or you have any other questions, please contact [Atlassian Support](#).

Detailed Error
[Click here to learn more](#)

- b. If the context path is other than empty or root (/), e.g. `/jira`, visiting JIRA via the proxy connector (e.g. `http://jiraserver:8080/jira`) should redirect you to the configured proxy (e.g. `https://jira.atlassian.com/jira`).

Step 2: Configure Apache HTTP Server

The installation of Apache and configuration of a DNS is not covered in this documentation. Additionally, it is assumed that Apache 2.2 has been installed and DNS entries have been configured for the JIRA domain. As Apache's configuration is specific to the operation system that is used, only some distributions and their configurations are currently documented.

2.1 Enable the Proxy Modules

Debian/Ubuntu

▼ [Expand to see Debian/Ubuntu instructions](#)

1. Enable the module with the following:

```
$ sudo a2enmod proxy_http
Considering dependency proxy for proxy_http:
Enabling module proxy.
Enabling module proxy_http.
To activate the new configuration, you need to run:
    service apache2 restart
```

2. Restart Apache.

Windows/Other OS

▼ [Expand to see Windows/Other OS instructions](#)

1. Locate and edit the `httpd.conf` file, adding the below lines:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

2. Restart Apache.

2.2. Configure Apache to use those Modules

Debian/Ubuntu

▼ [Expand to see Debian/Ubuntu instructions](#)

1. Switch into user `root`.
2. Backup the existing instance or create a new one. Creating a new instance is not covered within this documentation (copying the default should be sufficient).
3. Modify the existing instance within `$APACHE_INSTALL/sites-available`, for example default.

4. Add the following inside the `VirtualHost`, replacing `jiraserver` with the hostname of the JIRA server and also modifying the port if required.

On its own domain or subdomain:

```
# JIRA Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ProxyRequests          Off
ProxyPreserveHost      On
ProxyPass               /          http://jiraserver:8080/
ProxyPassReverse        /          http://jiraserver:8080/
```

- i** Missing a forward slash at the end of the URL will cause proxy errors - ensure this is in place!
Using a context path:

```
# JIRA Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ProxyRequests          Off
ProxyPreserveHost      On
ProxyPass               /jira
http://jiraserver:8080/jira
ProxyPassReverse        /jira
http://jiraserver:8080/jira
```

- i** The path used must be identical to the Tomcat context path. For example, forwarding `/jira` to `/jira520` cannot be done without considerable rewrite rules that are not always reliable.
5. (Optional): Enable the instance with the following:

```
# a2ensite jira
Enabling site jira.
To activate the new configuration, you need to run:
    service apache2 reload
```

- i** This is only required if a new instance has been created in favor of using the default.
6. Reload the Apache configuration.
 7. Test by accessing JIRA through Apache, for example <http://jira.com> or <http://atlassian.com/jira>.

Windows/Other OS

▼ Expand to see [Windows/Other OS instructions](#)

1. Locate and edit the `httpd.conf` file.
2. Add the following inside the `VirtualHost`, replacing `jiraserver` with the hostname of the JIRA server and also modifying the port if required.

On its own domain or subdomain:

```
# JIRA Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ProxyRequests          Off
ProxyPreserveHost      On
ProxyPass               /          http://jiraserver:8080/
ProxyPassReverse        /          http://jiraserver:8080/
```

i Missing a forward slash at the end of the URL will cause proxy errors - ensure this is in place!

Using a context path:

```
# JIRA Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ProxyRequests          Off
ProxyPreserveHost      On
ProxyPass               /jira
http://jiraserver:8080/jira
ProxyPassReverse        /jira
http://jiraserver:8080/jira
```

i The path used must be identical to the Tomcat context path. For example, forwarding `/jira` to `/jira520` cannot be done without considerable rewrite rules that are not always reliable.

3. Restart Apache.
4. Test by accessing JIRA through Apache, for example <http://jira.com> or <http://atlassian.com/jira>.

Step 3: Configure JIRA

1. Set **Use gzip compression** to **OFF** as in [Configuring JIRA options](#). GZIP compression is known to cause performance issues using a reverse-proxy, especially if the proxy is also compressing the traffic.
2. Set the **Base URL** to be the FQDN that JIRA will be accessed on, for example <http://jira.atlassian.com>. This is also located in [Configuring JIRA options](#).
 - !** JIRA can only be configured to respond to a single URL and the Base URL (as in [Configuring JIRA options](#)) must match the URL end-users are accessing. Misconfiguration of this may cause significant problems within JIRA such as the Activity Stream and Dashboard Gadgets failing to function correctly.
3. Test by accessing JIRA on the FQDN (e.g.: <http://jira.atlassian.com>), ensuring that JIRA is accessible and all dashboard gadgets correctly display.

Troubleshooting

- **Hijacked Sessions:** Some users have reported problems with user sessions being hijacked when the `mod_cache` module is enabled. If these problems are encountered, try disabling the `mod_cache` module.
 - i** This module is enabled by default in some Apache HTTP Server version 2 distributions.
- **Permission Denied Errors enabling `mod_proxy` (and `mod_jk`) on Linux distros that use SELinux:** Users have reported 'permission denied' errors when trying to get `mod_proxy` (and `mod_jk`) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- **Running Mac OS X:** Disable **webperfcache**, which proxies port 80 by default. A user reported this as

the likely cause of JIRA session problems, in the form of users' identities becoming mixed up, as below.

⚠ Additionally we do not recommend using Max OS X as it is not supported, as in our [Supported platforms](#).

The OSX Servers enable webperfcache by default for Virtual Hosts, which for static content would be great, but for dynamic instances (which ALL of ours are) it is Evil and causes many issues.

Of note recently was the jira session issue. Also see :-

<http://developer.apple.com/documentation/Darwin/Reference/ManPages/man8/webperfcache.8.html>

Unfortunately even if you disable webperfcache for a instance, if there is a single instance enabled then all instances will still proxy through webperfcache with resulting session problems.

- **Too many redirects:** Both Tomcat & Apache are redirecting, when only one should be. Disable redirection in Tomcat (revert any changes as in [Running JIRA over SSL or HTTPS](#)) and check that there is only one redirection in Apache.
- **General Problems:**
 1. Clear the browser cache and try again.
 2. Ensure that JIRA works as expected when running directly from Tomcat and bypassing Apache. For example, accessing `http://jiraserver:8080` instead of `http://jira.atlassian.com`.
 3. Increase the [LogLevel](#) for Apache to debug and restart it.
 4. Attempt to access JIRA and check the [Apache Log Files](#) for any errors.
 5. Raise a question on [Atlassian Answers](#) for assistance.
- **403 Forbidden error:**
 - Add the `RequestHeader unset Authorization` line to the apache configuration page to disable authorization headers.

```
<Location /jira>
  RequestHeader unset Authorization
  ProxyPreserveHost On
  ProxyPass http://jiraserver/jira
  ProxyPassReverse http://jiraserver/jira
</Location>
```

See also

- [Integrating JIRA with Apache using SSL](#)
- [Configuring Apache Reverse Proxy Using the AJP Protocol](#)
- For more advanced `mod_webapp` configurations (eg. SSL), see [this mod_proxy guide](#).
- [Apache Virtual Host documentation](#)

Configuring Apache Reverse Proxy Using the AJP Protocol

Atlassian applications allow the use of reverse-proxies within our products, however Atlassian Support does not provide assistance for configuring them. Consequently, Atlassian **can not guarantee providing any support for them**.

If assistance with configuration is required, please raise a question on [Atlassian Answers](#).

This page describes how to integrate [Apache HTTP Server](#) (also referred to as `httpd`) with JIRA, utilizing `mod_proxy_ajp` so that Apache operates as a reverse-proxy. AJP is a wire protocol and is an optimized version of the HTTP protocol to allow a standalone web server such as [Apache](#) to talk to Tomcat.

This protocol can be used in favor of `HTTP/1.1` as in either of the following Apache configurations:

- [Integrating JIRA with Apache](#)
- [Integrating JIRA with Apache using SSL](#)

On this page:

- [Step 1: Configure Tomcat](#)
- [Step 2: Configure Apache HTTP Server](#)
 - [2.1 Enable the Proxy Modules](#)
 - [2.2. Configure Apache to use those Modules](#)
 - [2.3 Redirect HTTP to HTTPS](#)
- [Step 3: Configure JIRA](#)
- [Troubleshooting](#)
- [See also](#)

Step 1: Configure Tomcat

1. Stop JIRA.
2. Enable the AJP Connector on the Tomcat container hosting JIRA by uncommenting the following element in `$JIRA_INSTALL/conf/server.xml`:

```
<Connector port="8009" URIEncoding="UTF-8" enableLookups="false"
protocol="AJP/1.3" />
```

3. Start JIRA.
4. Test that JIRA is accessible on the standard HTTP connector, for example `http://jiraserver:8080`. This is to ensure that Tomcat has successfully restarted.

Step 2: Configure Apache HTTP Server

The installation of Apache and configuration of a DNS is not covered in this documentation. Additionally, it is assumed that Apache 2.2 has been installed and DNS entries have been configured for the JIRA domain. As

Apache's configuration is specific to the operation system that is used, only some distributions and their configurations are currently documented.

2.1 Enable the Proxy Modules Debian/Ubuntu

▼ [Expand to see Debian/Ubuntu instructions](#)

1. Enable the module with the following:

```
$ sudo a2enmod proxy_ajp
Considering dependency proxy for proxy_ajp:
Module proxy already enabled
Enabling module proxy_ajp.
To activate the new configuration, you need to run:
    service apache2 restart
```

2. Restart Apache.

Windows/Other OS

▼ [Expand to see Windows/Other OS instructions](#)

1. Locate and edit the `httpd.conf` file, adding the below lines:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
```

2. Restart Apache.

2.2. Configure Apache to use those Modules Debian/Ubuntu

▼ [Expand to see Debian/Ubuntu instructions](#)

1. Switch into user `root`.
2. Backup the existing site or create a new one. Creating a new site is not covered within this documentation (copying the default should be sufficient).
3. Modify the existing site within `$APACHE_INSTALL/sites-available`, for example `default` (HTTP) or `default-ssl` (HTTPS).
4. Add the following inside the `VirtualHost`, replacing `jiraserver` with the hostname of the JIRA server and also modifying the port if required.

```
# JIRA AJP Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ProxyRequests            Off
ProxyPass                /          ajp://jiraserver:8009/
ProxyPassReverse         /          ajp://jiraserver:8009/
```

i Missing a forward slash at the end of the URL will cause proxy errors - ensure this is in place!

5. (Optional): Enable the site with the following:

```
# a2ensite jira
Enabling site jira.
To activate the new configuration, you need to run:
    service apache2 reload
```

i This is only required if a new site has been created in favor of using the default.

6. **If using HTTP, skip to step 8.** For HTTPS, the certificates need to be installed by copying the

certificate and private key to the appropriate directories and the following will also need to be added to the site:

```
SSLProxyEngine          On
```

7. Include them in the Apache configuration, within the `VirtualHost` as below:

```
SSLCertificateFile      /etc/ssl/certs/jira.crt
SSLCertificateKeyFile   /etc/ssl/private/jira.key
```

8. Reload the Apache configuration.

9. Test by accessing JIRA through Apache, for example <http://jira.com> or <http://atlassian.com/jira>.

Windows/Other OS

▼ [Expand to see Windows/Other OS instructions](#)

1. Locate and edit the `httpd.conf` file.

2. Add the following inside the `VirtualHost`, replacing `jiraserver` with the hostname of the JIRA server and also modifying the port if required.

```
# JIRA AJP Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

ProxyRequests          Off
ProxyPass               /          ajp://jiraserver:8009/
ProxyPassReverse       /          ajp://jiraserver:8009/
```

i Missing a forward slash at the end of the URL will cause proxy errors - ensure this is in place!

3. **If using HTTP, skip to step 5.** For HTTPS, the certificates need to be installed by copying the certificate and private key to the appropriate directories and the following will also need to be added to the site:

```
SSLProxyEngine          On
```

4. Include them in the Apache configuration, within the `VirtualHost` as below:

```
SSLCertificateFile      /etc/ssl/certs/jira.crt
SSLCertificateKeyFile   /etc/ssl/private/jira.key
```

5. Restart Apache.

6. Test by accessing JIRA through Apache, for example <http://jira.com> or <http://atlassian.com/jira>.

2.3 Redirect HTTP to HTTPS

This is an optional step and is only required if using HTTPS. It can be done by using `mod_rewrite` (this module may require enabling), add the following to the HTTP `VirtualHost`:

```

RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

```

Step 3: Configure JIRA

1. Set **Use gzip compression** to **OFF** as in [Configuring JIRA options](#). GZIP compression is known to cause performance issues using a reverse-proxy, especially if the proxy is also compressing the traffic.
2. Set the **Base URL** to be the FQDN that JIRA will be accessed on, for example <http://jira.atlassian.com>. This is also located in [Configuring JIRA options](#).
 ⚠️ JIRA can only be configured to respond to a single URL and the Base URL (as in [Configuring JIRA options](#)) must match the URL end-users are accessing. Misconfiguration of this may cause significant problems within JIRA such as the Activity Stream and Dashboard Gadgets failing to function correctly.
3. Test by accessing JIRA on the FQDN (e.g.: <http://jira.atlassian.com>), ensuring that JIRA is accessible and all dashboard gadgets correctly display.

Troubleshooting

- **Hijacked Sessions:** Some users have reported problems with user sessions being hijacked when the `mod_cache` module is enabled. If these problems are encountered, try disabling the `mod_cache` module.
 ⓘ This module is enabled by default in some Apache HTTP Server version 2 distributions.
- **Permission Denied Errors enabling `mod_proxy` (and `mod_jk`) on Linux distros that use SELinux:** Users have reported 'permission denied' errors when trying to get `mod_proxy` (and `mod_jk`) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- **Running Mac OS X:** Disable **webperfcache**, which proxies port 80 by default. A user reported this as the likely cause of JIRA session problems, in the form of users' identities becoming mixed up, as below.
 ⚠️ Additionally we do not recommend using Mac OS X as it is not supported, as in our [Supported platforms](#).

The OSX Servers enable webperfcache by default for Virtual Hosts, which for static content would be great, but for dynamic instances (which ALL of ours are) it is Evil and causes many issues.

Of note recently was the jira session issue. Also see :-

<http://developer.apple.com/documentation/Darwin/Reference/ManPages/man8/webperfcache.8.html>

Unfortunately even if you disable webperfcache for a instance, if there is a single instance enabled then all instances will still proxy through webperfcache with resulting session problems.

- **Too many redirects:** Both Tomcat & Apache are redirecting, when only one should be. Disable redirection in Tomcat (revert any changes as in [Running JIRA over SSL or HTTPS](#)) and check that there is only one redirection in Apache.
- **General Problems:**
 1. Clear the browser cache and try again.
 2. Ensure that JIRA works as expected when running directly from Tomcat and bypassing Apache. For example, accessing `http://jiraserver:8080` instead of `http://jira.atlassian.com`.
 3. Increase the **LogLevel** for Apache to debug and restart it.
 4. Attempt to access JIRA and check the [Apache Log Files](#) for any errors.
 5. Raise a question on [Atlassian Answers](#) for assistance.
- **403 Forbidden error:**
 - Add the `RequestHeader unset Authorization` line to the apache configuration page to disable authorization headers.

```
<Location /jira>
  RequestHeader unset Authorization
  ProxyPreserveHost On
  ProxyPass http://jiraserver/jira
  ProxyPassReverse http://jiraserver/jira
</Location>
```

See also

- [Integrating JIRA with Apache](#)
- [Integrating JIRA with Apache using SSL](#)
- [Apache Virtual Host documentation](#)

Integrating JIRA with Apache using SSL

Atlassian applications allow the use of reverse-proxies within our products, however Atlassian Support does not provide assistance for configuring them. Consequently, Atlassian **can not guarantee providing any support for them.**

If assistance with configuration is required, please raise a question on [Atlassian Answers](#).

This page describes how to integrate [Apache HTTP Server](#) (also referred to as `httpd`) with JIRA, utilizing `mod_proxy` & `mod_ssl` so that Apache operates as a reverse-proxy over HTTPS. If a HTTP configuration is required, please see our [Integrating JIRA with Apache](#) documentation. Configuring Apache allows for running JIRA on non-standard HTTP port (such as 8080) and users will be able to access JIRA over standard HTTPS as their traffic will be routed through the proxy and encrypted outside of the network.

Apache can be configured to allow access to JIRA in any of the following methods:

- Directly on its own domain: <https://atlassian.com/>
- As a subdomain of another domain: <https://jira.atlassian.com>
- It can also be accessed on a context path on either a domain or subdomain: <https://atlassian.com/jira>

This means the SSL certificate will be managed within Apache and not Tomcat, additionally the connection between Apache and Tomcat will not be encrypted. However, the connection between the browser and the outside network **will be encrypted**. This is suitable for configurations where the JIRA server is within the same network as the Apache server and is illustrated below:

```
Client Browser -> HTTPS -> Apache Proxy ->
HTTP -> Tomcat (JIRA)
```

On this page:

- Before you begin
- Step 1: Configure Tomcat
- Step 2: Configure Apache HTTP Server
 - 2.1 Enable the Proxy Modules
 - 2.2. Configure Apache to use those Modules
 - 2.3 Redirect HTTP to HTTPS
- Step 3: Configure JIRA
- Troubleshooting
- See Also

This is a common configuration for networks with multiple SSL certificates and/or web applications as they are all managed in one location (Apache).

If a more complicated solution is required, refer to the [Apache HTTP Server Version Documentation](#), consult with the Apache SME within your organization, and if need be, raise a question on [Atlassian Answers](#), or get in touch with one of our [Atlassian Experts](#).

▼ [Expand for an example of a common Apache configuration](#)

1. JIRA is running on port 8080 on a server within the LAN that cannot be accessed externally (the router/firewall is not forwarding port 8080 to it).
2. Apache is set up on another server (or the same server as JIRA) that can be accessed externally on HTTPS (443).
3. Apache is then accessed over HTTPS on the appropriate URL (`VirtualHost`), routing the traffic to and from the JIRA server.

Before you begin

! It is expected that the SSL certificate has been signed by a CA and is in the PEM format prior to configuring Apache. For assistance preparing and generating SSL certificates, please consult with a SSL Vendor (for example, GoDaddy, Verisign, RapidSSL).

Identifying whether to use a domain, subdomain or context path largely depends on the type of SSL certificate provided and also any business rules around website configurations. For SSL to function without error, the domain must match the Common Name (CN) of the certificate.

▼ [Expand for further information on configuring the FQDN to match the certificate's CN](#)

This table indicates which URLs will work with the certificate CN and also makes a recommendation on the URL to use.

JIRA FQDN	Common Name	Valid	Recommend JIRA FQDN
https://jira.atlassian.com	jira.atlassian.com		https://jira.atlassian.com
https://jira.atlassian.com	*.atlassian.com		https://jira.atlassian.com
https://jira.atlassian.com	atlassian.com		https://atlassian.com/jira
https://atlassian.com	atlassian.com		https://atlassian.com/jira
https://atlassian.com	jira.atlassian.com		https://jira.atlassian.com

A certificate that has a CN with an asterisk (*) in it is a **wildcard certificate** and can support any subdomain of that domain. If you are uncertain about the URL to use, please consult with your System Administrator and the SSL vendor that provided the certificate.

Step 1: Configure Tomcat

1. Stop JIRA.
2. (Optional: If JIRA does not require a context path, skip this step.)

Edit Tomcat's `server.xml` to include the required JIRA context path. The below example uses `path="jira"` - this means JIRA is accessible on `http://jiraserver:8080/jira` given the default JIRA port is used.

```

<Engine defaultHost="localhost" name="Catalina">
    <Host appBase="webapps" autoDeploy="true"
name="localhost" unpackWARs="true">
        <Context
docBase="\${catalina.home}/atlassian-jira" path="/jira"
reloadable="false" useHttpOnly="true">

                <!--

=====
=====
                Note, you no longer configure your database
driver or connection parameters here.
                These are configured through the UI during
application setup.

=====
=====
                -->
                <Resource auth="Container"
factory="org.objectweb.jotm.UserTransactionFactory"
jotm.timeout="60" name="UserTransaction"
type="javax.transaction.UserTransaction"/>
                <Manager pathname="" />
        </Context>
    </Host>

```

i Ensure the path value is set with a preceding forward slash (/). For example, path="/jira" rather than path="jira".

3. Edit Tomcat's server.xml to include a separate connector to proxy the requests. This requires the scheme, proxyName & proxyPort attributes. Replace them with the appropriate domain and port of the proxy, as in the below example:

```

<Service name="Catalina">

    <!-- Apache Proxy Connector with values for scheme, proxyName
and proxyPort -->
        <Connector acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" enableLookups="false"
maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
port="8080" protocol="HTTP/1.1" redirectPort="8443"
useBodyEncodingForURI="true"
                scheme="https" proxyName="jira.atlassian.com"
proxyPort="443"/>

    <!-- Standard HTTP Connector -->
        <Connector acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" enableLookups="false"
maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
port="8081" protocol="HTTP/1.1" redirectPort="8443"
useBodyEncodingForURI="true"/>

```

4. Disable any redirections within Tomcat to HTTPS if they have been enabled - for example the changes to WEB-INF/web.xml in [Running JIRA applications over SSL or HTTPS](#) will cause errors

when using Apache.

5. Start JIRA.
6. Test that JIRA is accessible on the normal connector, using a context path if applicable - for example <http://jiraserver:8081/jira>.
7. Test that the new connector is in effect by accessing JIRA on the appropriate proxy connector. The behavior varies depending on the context path:
 - a. If the context path is empty or root (/), visiting JIRA via the proxy connector (e.g. <http://jiraserver:8080/>) should take you to JIRA with a warning:

We've detected a potential problem with JIRA's Dashboard configuration that your administrator can correct. [Hide](#)

Dashboard Diagnostics: Mismatched URL Hostname

JIRA is reporting that it is running on the hostname 'jira.atlassian.com', which does not match the hostname used to run these diagnostics, 'localhost'. This is known to cause JIRA to construct URLs using the incorrect hostname, which will result in errors in the dashboard, among other issues.

The most common cause of this is the use of a reverse-proxy HTTP server (often Apache or IIS) in front of the application server running JIRA. While this configuration is supported, some additional setup might be necessary in order to ensure that JIRA detects the correct hostname.

The following articles describe the issue and the steps you should take to ensure that your web server and app server are configured correctly:

- [Gadgets do not display correctly after upgrade to JIRA 4.0](#)
- [Integrating JIRA with Apache](#)
- [Integrating JIRA with Apache using SSL](#)

If you believe this diagnosis is in error, or you have any other questions, please contact [Atlassian Support](#).

Detailed Error
[Click here to learn more](#)

- b. If the context path is other than empty or root (/), e.g. /jira , visiting JIRA via the proxy connector (e.g. <http://jiraserver:8080/jira>) should redirect you to the configured proxy (e.g. <https://jira.atlassian.com/jira>).

We use two different Tomcat connectors so that testing can be done on JIRA, bypassing the proxy when needed as this is a useful step when troubleshooting. It is expected that the standard connector will not be allowed external access from outside the network (the firewall will not forward any ports to it).

Step 2: Configure Apache HTTP Server

The installation of Apache and configuration of a DNS is not covered in this documentation. Additionally, it is assumed that Apache 2.2 has been installed and DNS entries have been configured for the JIRA domain. As Apache's configuration is specific to the operation system that is used, only some distributions and their configurations are currently documented.

2.1 Enable the Proxy Modules Debian/Ubuntu

▼ [Expand to see Debian/Ubuntu instructions](#)

1. Enable the module with the following:

```
$ sudo a2enmod proxy_http ssl
Considering dependency proxy for proxy_http:
Enabling module proxy.
Enabling module proxy_http.
Enabling module ssl.
See /usr/share/doc/apache2.2-common/README.Debian.gz on how to
configure SSL and create self-signed certificates.
To activate the new configuration, you need to run:
    service apache2 restart
```

2. Restart Apache.

Windows/Other OS

▼ [Expand to see Windows/Other OS instructions](#)

1. Locate and edit the `httpd.conf` file, adding the below lines if they do not already exist:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule ssl_module modules/mod_ssl.so
```

2. Restart Apache.


```
SSLCertificateFile    /etc/ssl/certs/jira.crt
SSLCertificateKeyFile /etc/ssl/private/jira.key
```

8. (OPTIONAL): Configuration of `SSLCertificateChainFile` will contain the intermediate certificates provided by the CA vendor who signed it. Please follow consult with the CA vendor to verify if this is required.

```
SSLCertificateChainFile /etc/ssl/certs/jiraintermediate.crt
```

9. Reload the Apache configuration.
10. Test by accessing JIRA through Apache, for example <http://jira.com> or <http://atlassian.com/jira>.

Windows/Other OS

▼ Expand to see Windows/Other OS instructions

1. Locate and edit the `httpd.conf` file.
2. Add the following inside the `VirtualHost`, replacing `jiraserver` with the hostname of the JIRA server and also modifying the port if required.

On its own domain or subdomain:

```
# JIRA Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

SSLProxyEngine          On
ProxyRequests           Off
ProxyPreserveHost       On
ProxyPass                /          http://jiraserver:8080/
ProxyPassReverse         /          http://jiraserver:8080/
```

i Missing a forward slash at the end of the URL will cause proxy errors - ensure this is in place!

Using a context path:

```
# JIRA Proxy Configuration:
<Proxy *>
    Order deny,allow
    Allow from all
</Proxy>

SSLProxyEngine          On
ProxyRequests           Off
ProxyPreserveHost       On
ProxyPass                /jira
http://jiraserver:8080/jira
ProxyPassReverse         /jira
http://jiraserver:8080/jira
```

i The path used must be identical to the Tomcat context path. For example, forwarding `/jira` to `/jira520` cannot be done without considerable rewrite rules that are not always reliable.

3. Copy the certificate and private key to the appropriate directories.
4. Include them in the Apache configuration, within the `VirtualHost` as below:

```
SSLCertificateFile      /etc/ssl/certs/jira.crt
SSLCertificateKeyFile  /etc/ssl/private/jira.key
```

5. (OPTIONAL): Configuration of `SSLCertificateChainFile` will contain the intermediate certificates provided by the CA vendor who signed it. Please follow consult with the CA vendor to verify if this is required.

```
SSLCertificateChainFile /etc/ssl/certs/jiraintermediate.crt
```

6. Restart Apache.
7. Test by accessing JIRA through Apache, for example <http://jira.com> or <http://atlassian.com/jira>.

2.3 Redirect HTTP to HTTPS

This can be done with either of the following:

- Set up the HTTP `VirtualHost` to forward to the same Tomcat Connector. Tomcat will redirect to HTTPS using the `scheme`, `proxyName` & `proxyPort` parameters. This can be done as in our [Integrating JIRA with Apache](#) documentation.
- Using `mod_rewrite` (this module may require enabling), add the following to the HTTP `VirtualHost`:

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

Step 3: Configure JIRA

1. Set **Use gzip compression** to **OFF** as in [Configuring JIRA options](#). GZIP compression is known to cause performance issues using a reverse-proxy, especially if the proxy is also compressing the traffic.
2. Set the **Base URL** to be the FQDN that JIRA will be accessed on, for example <https://jira.atlassian.com>. This is also located in [Configuring JIRA options](#).
 - ⚠ JIRA can only be configured to respond to a single URL and the Base URL (as in [Configuring JIRA options](#)) must match the URL end-users are accessing. Misconfiguration of this may cause significant problems within JIRA such as the Activity Stream and Dashboard Gadgets failing to function correctly.
3. Test by accessing JIRA on the FQDN (e.g.: <https://jira.atlassian.com>), ensuring that JIRA is accessible and all dashboard gadgets correctly display.

Troubleshooting

- **Hijacked Sessions:** Some users have reported problems with user sessions being hijacked when the `mod_cache` module is enabled. If these problems are encountered, try disabling the `mod_cache` module.
 - ℹ This module is enabled by default in some Apache HTTP Server version 2 distributions.
- **Permission Denied Errors enabling `mod_proxy` (and `mod_jk`) on Linux distros that use SELinux:** Users have reported 'permission denied' errors when trying to get `mod_proxy` (and `mod_jk`) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- **Running Mac OS X:** Disable **webperfcache**, which proxies port 80 by default. A user reported this as the likely cause of JIRA session problems, in the form of users' identities becoming mixed up, as below.
 - ⚠ Additionally we do not recommend using Max OS X as it is not supported, as in our [Supported platforms](#).

The OSX Servers enable `webperfcache` by default for Virtual Hosts, which for static content would be great, but for dynamic instances (which ALL of ours are) it is Evil and causes many

issues.

Of note recently was the jira session issue. Also see :-

<http://developer.apple.com/documentation/Darwin/Reference/ManPages/man8/webperfcache.8.html>

Unfortunately even if you disable webperfcache for a instance, if there is a single instance enabled then all instances will still proxy through webperfcache with resulting session problems.

- **Too many redirects:** Both Tomcat & Apache are redirecting, when only one should be. Disable redirection in Tomcat (revert any changes as in [Running JIRA over SSL or HTTPS](#)) and check that there is only one redirection in Apache.
- **General Problems:**
 1. Clear the browser cache and try again.
 2. Ensure that JIRA works as expected when running directly from Tomcat and bypassing Apache. For example, accessing `http://jiraserver:8080` instead of `http://jira.atlassian.com`.
 3. Increase the [LogLevel](#) for Apache to debug and restart it.
 4. Attempt to access JIRA and check the [Apache Log Files](#) for any errors.
 5. Raise a question on [Atlassian Answers](#) for assistance.
- **403 Forbidden error:**
 - Add the `RequestHeader unset Authorization` line to the apache configuration page to disable authorization headers.

```
<Location /jira>
  RequestHeader unset Authorization
  ProxyPreserveHost On
  ProxyPass http://jiraserver/jira
  ProxyPassReverse http://jiraserver/jira
</Location>
```

See Also

- [Integrating JIRA with Apache](#)
- [Configuring Apache Reverse Proxy Using the AJP Protocol](#)
- For more advanced `mod_webapp` configurations (eg. SSL), see [this mod_proxy guide](#).
- [Apache Virtual Host documentation](#)

Troubleshooting Apache

- **Hijacked Sessions:** Some users have reported problems with user sessions being hijacked when the `mod_cache` module is enabled. If these problems are encountered, try disabling the `mod_cache` module.  This module is enabled by default in some Apache HTTP Server version 2 distributions.
- **Permission Denied Errors enabling `mod_proxy` (and `mod_jk`) on Linux distros that use SELinux:** Users have reported 'permission denied' errors when trying to get `mod_proxy` (and `mod_jk`) working. Disabling SELinux (`/etc/selinux/config`) apparently fixes this.
- **Running Mac OS X:** Disable **webperfcache**, which proxies port 80 by default. A user reported this as the likely cause of JIRA session problems, in the form of users' identities becoming mixed up, as below.  Additionally we do not recommend using Mac OS X as it is not supported, as in our [Supported platforms](#).

The OSX Servers enable webperfcache by default for Virtual Hosts, which for static content would be great, but for dynamic instances (which ALL of ours are) it is Evil and causes many issues.

Of note recently was the jira session issue. Also see :-

<http://developer.apple.com/documentation/Darwin/Reference/ManPages/man8/webperfcache.8.html>

Unfortunately even if you disable webperfcache for a instance, if there is a single instance enabled then all instances will still proxy through webperfcache with resulting session problems.

- **Too many redirects:** Both Tomcat & Apache are redirecting, when only one should be. Disable redirection in Tomcat (revert any changes as in [Running JIRA over SSL or HTTPS](#)) and check that there is only one redirection in Apache.
- **General Problems:**

1. Clear the browser cache and try again.
 2. Ensure that JIRA works as expected when running directly from Tomcat and bypassing Apache. For example, accessing `http://jiraserver:8080` instead of `http://jira.atlassian.com`.
 3. Increase the [LogLevel](#) for Apache to debug and restart it.
 4. Attempt to access JIRA and check the [Apache Log Files](#) for any errors.
 5. Raise a question on [Atlassian Answers](#) for assistance.
- **403 Forbidden error:**
 - Add the `RequestHeader unset Authorization` line to the apache configuration page to disable authorization headers.

```
<Location /jira>
  RequestHeader unset Authorization
  ProxyPreserveHost On
  ProxyPass http://jiraserver/jira
  ProxyPassReverse http://jiraserver/jira
</Location>
```

Securing JIRA applications with Apache HTTP Server

The following outlines some basic techniques to secure a JIRA instance using Apache HTTP Server. These instructions are basic to-do lists and should not be considered comprehensive. For more advanced security topics see the "Further Information" section below.

- [Using Apache to limit access to the JIRA administration interface](#)
- [Using Fail2Ban to limit login attempts](#) (JIRA 4.1 has login-rate limiting, but Fail2Ban can be useful for older versions and more advanced security setups.)

Further information

- [Integrating JIRA with Apache](#)

Using Apache to limit access to the JIRA administration interface

Limiting administration to specific IP addresses

The JIRA administration interface is a critical part of the application; anyone with access to it can potentially compromise not only the JIRA instance but the entire machine. As well as limiting access to users who really need it, and using strong passwords, you should consider limiting access to it to certain machines on the network or internet. If you are using an [Apache HTTP Server](#), this can be done with Apache's **Location** functionality as follows.

1. Create a file that defines permission settings

This file can be in the Apache configuration directory or in a system-wide directory. For this example we'll call it "sysadmin_ips_only.conf". This file should contain the following:

```
Order Deny,Allow
Deny from All

# Mark the Sysadmin's workstation
Allow from 192.168.12.42
```

2. Add the file to your Virtual Host

In your Apache Virtual Host, add the following lines to restrict the administration actions to the Systems Administrator:

```
<LocationMatch Administrators.jspa>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch DeleteAttachment>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AcknowledgeTask>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ActivateWorkflow>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ActivateWorkflowStep2>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddIssueSecurity>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddIssueSecurityScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddLevel>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddNotification>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddNotificationScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddPermission>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddPermissionScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddPopMailServer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddProject>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddProjectCategory>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddRepository>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddSmtplibMailServer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddUser>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowSchemeEntity>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch AddWorkflowTransition>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowTransitionCondition>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowTransitionConditionParams>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowTransitionFunctionParams>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowTransitionPostFunction>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowTransitionValidator>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AddWorkflowTransitionValidatorParams>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AssociateFieldToScreens>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AssociateIssueTypeSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch AssociateIssueTypeSchemesWithDefault>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch BugzillaImport>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch BulkEditUserGroups>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CloneWorkflow>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureCache>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureCsvMapping>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureCustomField>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureFieldLayout>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureFieldLayoutScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureFieldScreen>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch ConfigureFieldScreenScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureFogBugzMapping>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureIssueTypeScreenScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureLogging>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ConfigureOptionSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CopyFieldLayout>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CopyFieldLayoutScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CopyIssueSecurityScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CopyNotificationScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CopyPermissionScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CopyWorkflowScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CreateCustomField>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CreateDraftWorkflow>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CsvImporter>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch CurrentUsersList>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteCustomField>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteGroup>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteIssueSecurity>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteIssueSecurityLevel>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch DeleteIssueSecurityScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteIssueType>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteLinkType>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteMailServer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteNotification>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteNotificationScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteOptionScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeletePermission>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeletePermissionScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeletePriority>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteProject>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteProjectCategory>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteProjectRole>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteRepository>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteResolution>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteStatus>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteSubTaskIssueType>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteTrustedApplication>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteUser>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch DeleteUserProperty>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowSchemeEntity>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowStep>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowTransitionCondition>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowTransitionPostFunction>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowTransitions>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DeleteWorkflowTransitionValidator>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch DisableSubTasks>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditAnnouncementBanner>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditApplicationProperties>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditAttachmentSettings>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditBasicConfig>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditCustomField>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditCustomFieldDefaults>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditCustomFieldOptions>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditDefaultFieldLayoutItem>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldLayout>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldLayoutItem>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch EditFieldLayoutItemRenderer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldLayoutItemRendererConfirmation>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldLayoutScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldScreen>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldScreenScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditFieldScreenSchemeItem>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditIssueSecurities>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditIssueSecurityScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditIssueType>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditIssueTypeScreenScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditLinkType>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditListener>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditLookAndFeel>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditNotifications>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditNotificationScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditPermissions>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditPermissionScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditPriority>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditProjectCategory>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch EditProjectRole>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditResolution>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditService>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditStatus>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditSubTaskIssueTypes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditTrustedApplication>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditUser>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditUserDefaultSettings>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditUserGroups>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditUserProjectRoles>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditUserProperties>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditUserProperty>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflow>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflowScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflowSchemeEntities>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflowStep>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflowTransition>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflowTransitionConditionParams>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EditWorkflowTransitionPostFunctionParams>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch EditWorkflowTransitionValidatorParams>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch EnterpriseSelectProjectRepository>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ExternalImport>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch FogBugzImport>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch GlobalPermissions>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch GroupBrowser>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ImportWorkflowFromXml>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch IndexAdmin>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch IndexOptimize>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch IntegrityChecker>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch JellyRunner>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch JiraSupportRequest>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch LDAPConfigurer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ListEventTypes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ListWorkflows>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch MailQueueAdmin>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch MakeDefaultLevel>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ManageConfiguration>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ManageConfigurationScheme>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch ManageIssueTypeSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ManageSubTasks>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch MantisImport>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch MigrateIssueTypes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectEmail>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportBackupOverviewProgress>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportMappingProgress>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportMissingMandatoryUsersCannotCreate>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportMissingMandatoryUsersExtMgmt>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportMissingOptionalUsersCannotCreate>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportMissingOptionalUsersExtMgmt>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportMissingUsersAutoCreate>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportProgress>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportResults>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportSelectBackup>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportSelectProject>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ProjectImportSummary>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch PublishDraftWorkflow>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch RepositoryTest>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch ResetFailedLoginCount>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchedulerAdmin>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeComparisonPicker>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeComparisonTool>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeGroupToRoleMapper>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeGroupToRoleResult>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeGroupToRoleTransformer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeMerge>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeMergePreview>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeMergeResult>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemePicker>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemePurgeToolPreview>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemePurgeToolResults>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemePurgeTypePicker>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeTools>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SchemeTypePicker>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectFieldLayoutScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectIssueTypeSchemeForProject>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectIssueTypeScreenScheme>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch SelectProjectCategory>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectIssueSecurityScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectPermissionScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectRepository>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectSecuritySchemeStep2>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectWorkflowScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectWorkflowSchemeStep2>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectProjectWorkflowSchemeStep3>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SelectScreenScheme>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SendBulkMail>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SendTestMail>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ServiceExecutor>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SetGlobalEmailPreference>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch SetPassword>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch TaskAdmin>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch TimeTrackingAdmin>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch TrackbackAdmin>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch UpdatePopMailServer>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch UpdateRepository>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch UpdateSmtplibMailServer>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch UserBrowser>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewApplicationProperties>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewAttachmentSettings>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewCustomFields>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewDefaultProjectRoleActors>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewFieldLayouts>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewFieldLayoutSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewFieldScreens>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewFieldScreenSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewGroup>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewIssueColumns>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewIssueFields>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewIssueSecuritySchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewIssueTypes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewIssueTypeScreenSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewLicense>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewLinkTypes>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch ViewListeners>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewLogging>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewLookAndFeel>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewMemoryInfo>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewNotificationSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewPermissionSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewPlugins>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewPriorities>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewProjectCategories>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewProjectRoles>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewProjectRoleUsage>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewResolutions>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewServices>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewStatuses>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewSystemInfo>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewTranslations>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewTrustedApplications>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewUpgradeHistory>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewUser>
  Include sysadmin_ips_only.conf
```

```
</LocationMatch>
<LocationMatch ViewUserDefaultSettings>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewUserProjectRoles>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowSchemes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowStep>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowStepMetaAttributes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowSteps>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowTransition>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowTransitionConditionalResult>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowTransitionMetaAttributes>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch ViewWorkflowXml>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch XmlBackup>
  Include sysadmin_ips_only.conf
</LocationMatch>
<LocationMatch XmlRestore>
```

```
Include sysadmin_ips_only.conf
</LocationMatch>
```

Using Fail2Ban to limit login attempts

JIRA 4.1 includes a [rate-limiting mechanism](#), but older versions and other applications such as Confluence need external help from a tool such as Fail2Ban.

What is Fail2Ban?

We need a means of defending sites against brute-force login attempts. [Fail2Ban](#) is a Python application which trails logfiles, looks for [regular expressions](#) and works with Shorewall (or directly with iptables) to apply temporary blacklists against addresses that match a pattern too often. This can be used to limit the rate at which a given machine hits login URLs for Confluence.

 *The information on this page does not apply to Confluence Cloud.*

Prerequisites

- Requires [Python 2.4](#) or higher to be installed
- Needs a specific file to follow, which means your Apache instance needs to log your Confluence access to a known logfile. You **should adjust the configuration below** appropriately.

How to set it up

This list is a skeletal version of the instructions

- There's an RPM available for RHEL on the [download page](#), but you can also download the source and set it up manually
- Its configuration files go into `/etc/fail2ban`
- The generic, default configuration goes into `.conf` files (`fail2ban.conf` and `jail.conf`). Don't change these, as it makes upgrading difficult.
- Overrides to the generic configuration go into `.local` files corresponding to the `.conf` files. These only need to contain the specific settings you want overridden, which helps maintainability.
- Filters go into `filter.d` — this is where you define regexps, each going into its own file
- Actions go into `action.d` — you probably won't need to add one, but it's handy to know what's available
- "jails" are a configuration unit that specify one regexp to check, and one or more actions to trigger when the threshold is reached, plus the threshold settings (e.g. more than 3 matches in 60 seconds causes that address to be blocked for 600 seconds)
- Jails are defined in `jail.conf` and `jail.local`. Don't forget the `enabled` setting for each one — it can be as bad to have the wrong ones enabled as to have the right ones disabled.

Running Fail2Ban

- Use `/etc/init.d/fail2ban {start|stop|status}` for the obvious operations
- Use `fail2ban-client -d` to get it to dump its current configuration to STDOUT. Very useful for troubleshooting.
- Mind the CPU usage; it can soak up resources pretty quickly on a busy site, even with simple regexp
- It can log either to syslog or a file, whichever suits your needs better

Common Configuration

jail.local

```

# The DEFAULT allows a global definition of the options. They can be
# override
# in each jail afterwards.

[DEFAULT]

# "ignoreip" can be an IP address, a CIDR mask or a DNS host. Fail2ban
# will not
# ban a host which matches an address in this list. Several addresses
# can be
# defined using space separator.
# ignoreip = <space-separated list of IPs>

# "bantime" is the number of seconds that a host is banned.
bantime = 600

# A host is banned if it has generated "maxretry" during the last
# "findtime"
# seconds.
findtime = 60

# "maxretry" is the number of failures before a host get banned.
maxretry = 3

[ssh-iptables]

enabled = false

[apache-shorewall]

enabled = true
filter = cac-login
action = shorewall
logpath = /var/log/httpd/confluence-access.log
bantime = 600
maxretry = 3
findtime = 60
backend = polling

```

Configuring for Confluence

The following is an example only, and you should adjust it for your site.

filter.d/confluence-login.conf

```

[Definition]

failregex = <HOST>.*"GET /login.action

ignoreregex =

```

Configuring for JIRA

The following is an example only, and you should adjust it for your site.

filter.d/jira-login.conf

```
[Definition]

failregex = <HOST>.*"GET /login.jsp

ignoreregex =
```

Changing JIRA application TCP ports**Why change JIRA application TCP ports?**

By default, JIRA applications use TCP listening port **8080** and hence, JIRA applications are typically available at `http://<yourserver>:8080`.

If, however, an existing service running on your machine is claiming port **8080**, there will be a conflict and JIRA applications will fail to start. You may see errors like this:

```
LifecycleException: Protocol handler initialization failed:
java.net.BindException: Address already in use:8080
```

This can be fixed by changing JIRA applications to use another TCP listening port (eg. **8100**) and shutdown port (eg. **8015**).

Changing JIRA application TCP ports

Before you change JIRA application TCP ports, read the following:

- **Which port number should I choose?** If you are not sure which port number to choose, use a tool such as *netstat* to determine which port numbers are free to use by JIRA applications. The highest port number that can be used is 65535 because it is the highest number which can be represented by an unsigned 16 bit binary number. [The Internet Assigned Numbers Authority \(IANA\)](#) lists the registration of commonly used port numbers for well-known Internet services, it's advisable to avoid any of those ports.
- **A note about firewalls:** When you choose a port number for JIRA, bear in mind that your firewall may prevent people from connecting to JIRA based on the port number. Organizations with a local network protected by a firewall typically need to consider modifying their firewall configuration whenever they install a web-based application (such as JIRA) that is running on a new port or host. Even personal laptop and desktop machines often come with firewall software installed that necessitates the same sort of change as described above. If JIRA does not need to be accessed from outside the firewall, then no firewall configuration changes will be necessary.

You can change JIRA's TCP ports by using the **JIRA configuration tool** or by **manually editing the server.xml file**. If you installed JIRA using the 'Windows Installer', 'Linux Installer', or from an 'Archive File', you can use the JIRA configuration tool.

Changing JIRA's TCP ports using the JIRA configuration tool

1. Start the JIRA configuration tool. See [Using the JIRA configuration tool](#) for instructions on where to find the tool.
2. Click the **Web Server** tab.
3. In the **HTTP Port** field, enter the new TCP listening port number.
4. In the **Control Port** field, enter the new TCP shutdown port number.

- Click the **Save** button. Your changes are saved to the `server.xml` file located in the `conf` subdirectory of your JIRA application installation directory.

Changing JIRA's TCP ports by editing the `server.xml` file

Edit the `server.xml` file in the `conf` subdirectory of the JIRA installation directory. The start of the file looks like:

```
<Server port="8005" shutdown="SHUTDOWN" >

  <Service name="Catalina">

    <Connector port="8080"
      maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
      maxSpareThreads="75"
      enableLookups="false" redirectPort="8443" acceptCount="100"
      connectionTimeout="20000" disableUploadTimeout="true" />

    ...
```

For example, change the shutdown port from "**8005**" to "**8015**" and the listening port (i.e. in the `<connector/>` element) from "**8080**" to "**8100**". (See [below](#) to decide which TCP port numbers should be used for JIRA.)

Then, restart JIRA and point a browser to `http://<yourserver>:8100`

– If you are running on a Unix server and bind the ports below 1024 (such as port 80 for example), you will **need to start JIRA as root** in order to successfully bind to the port.

Related topics

[Changing Confluence's listening ports](#)

[Connecting to SSL services](#)

Atlassian applications allow the use of SSL within our applications, however Atlassian Support does not provide assistance for configuring it. Consequently, Atlassian **can not guarantee providing any support for it.**

- If assistance with conversions of certificates is required, please consult with the vendor who provided the certificate.
- If assistance with configuration is required, please raise a question on [Atlassian Answers](#).

This page describes how to get web applications like JIRA and Confluence connecting to external servers over SSL, via the various SSL-wrapped protocols. For instance, you may want to:

- Refer to an `https://...` URL in a Confluence macro.
- Use an IMAPS server to retrieve mail in JIRA.
- Use SMTP over SSL (SMTPS) to send mail in JIRA.
- Connect to a LDAP directory over SSL.
- Set up **Trusted Applications** over SSL.

If you want to run JIRA *itself* over SSL, see [Running JIRA applications over SSL or HTTPS or Integrating JIRA with Apache using SSL](#).

Add SSL Certificates automatically!

We now have a JIRA SSL [Atlassian Labs plugin](#) for this process. Please install and use the plugin before going through these docs.

On this page:

- Problem Symptoms
- The Cause
- Resolution
 - Obtain and Import the Server's Public Certificate
 - Alternative KeyStore Locations
 - Debugging

Problem Symptoms

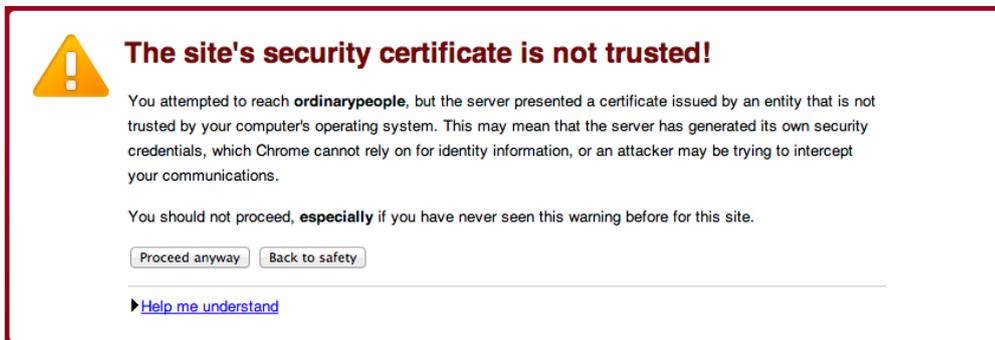
Attempting to access URLs that are encrypted with SSL (for example HTTPS, LDAPS, IMAPS) throws an exception and JIRA refuses to connect to it. For example:

```

javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to
find valid certification path to requested target
  at com.sun.mail.imap.IMAPStore.protocolConnect(IMAPStore.java:441)
  at javax.mail.Service.connect(Service.java:233)
  at javax.mail.Service.connect(Service.java:134)

```

This is the same as the following error that's generated in Chrome when visiting a page that's encrypted with a self-signed certificate, except Java can't "Proceed anyway", it just refuses the certificate:

**The Cause**

Whenever JIRA attempts to connect to another application over SSL (e.g.: HTTPS, IMAPS, LDAPS), it will *only* be able to connect to that application if it can trust it. The way trust is handled in the Java world (this is what JIRA is written in) is that you have a keystore (typically `$JAVA_HOME/lib/security/cacerts`) or also known as the trust store. This contains a list of all the known CA certificates and Java will only trust certificates that are signed by those CA certificate or public certificates that exist within that keystore. For example, if we look at the certificate for Atlassian:

We can see the `*.atlassian.com` certificate has been signed by the intermediate certificates, **DigiCert High Assurance EV Root CA** and **DigiCert High Assurance CA-3**. These intermediate certificates have been signed by the root **Entrust.net Secure Server CA**. Those three certificates combined are referred to as the certificate chain. As all of those CA certificates are within the Java keystore (`cacerts`), Java will trust any certificates signed by them (in this case, `*.atlassian.com`). Alternatively, if the `*.atlassian.com` certificate was in the keystore, Java would also trust that site.

This problem comes from a certificate that is either self-signed (a CA did not sign it) or the certificate chain does not exist within the Java keystore. Subsequently, JIRA doesn't trust the certificate and fails to connect to the application.

Resolution

In order to resolve this, the public certificate need to be imported in the Java keystore that JIRA uses. In the example above, this is *.[atlassian.com](#) and we cover how to install it below.

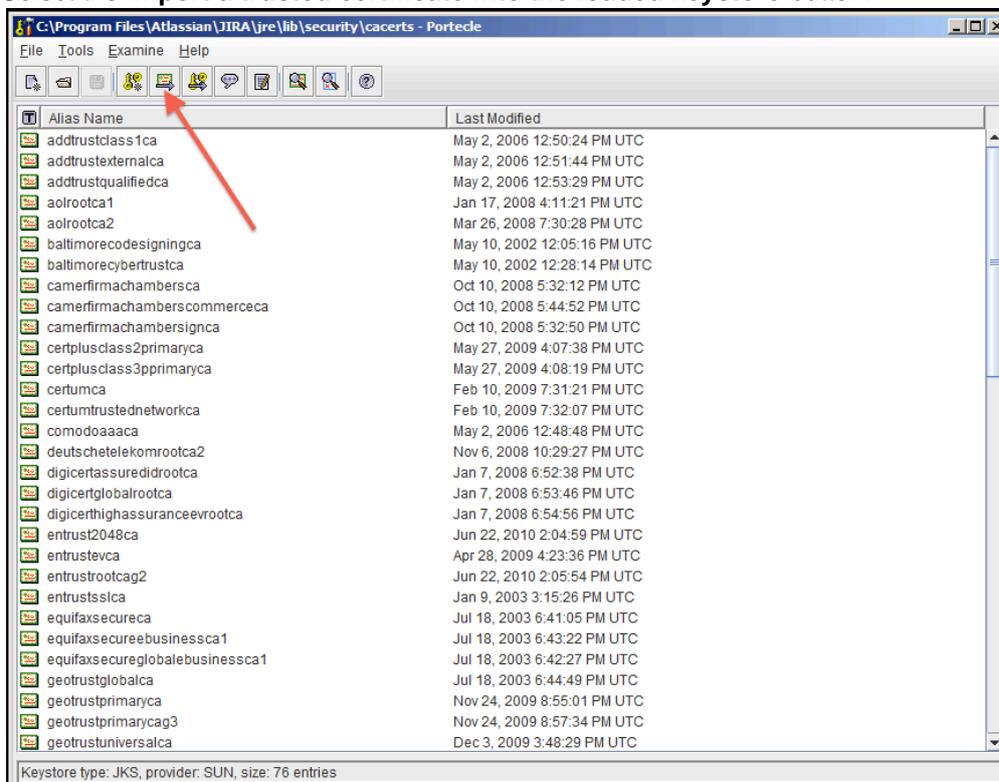
If you're unable to install Portecle on the server or prefer the command line please see our [Command Line Installation](#) section below.

Obtain and Import the Server's Public Certificate

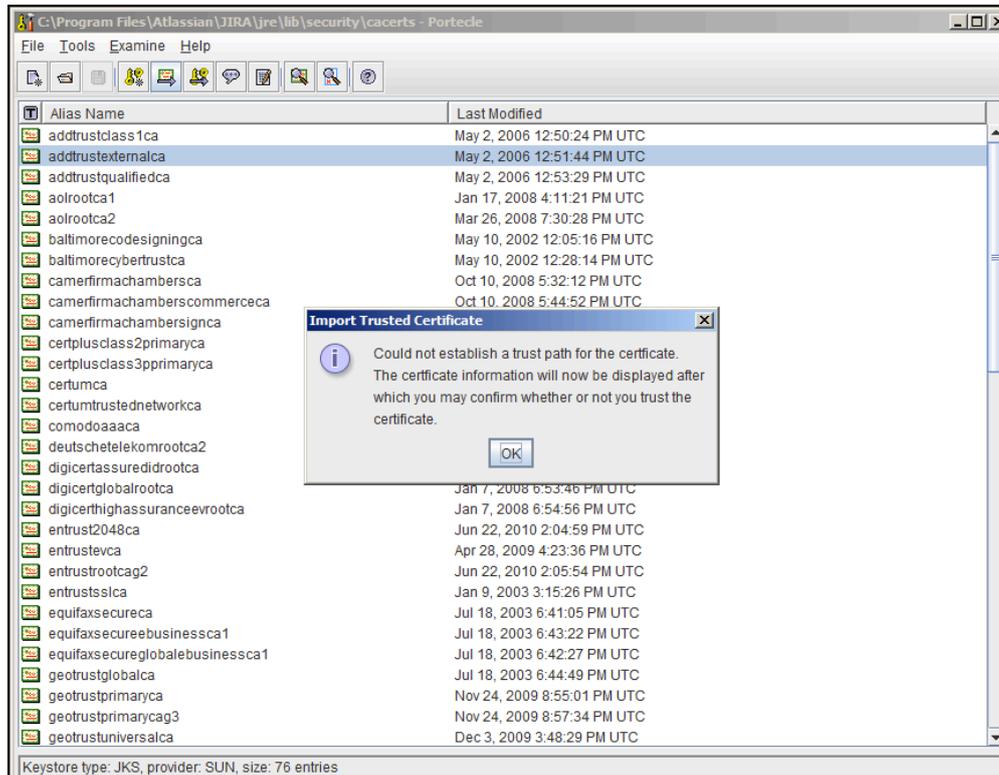
1. Download and install the [Portecle](#) app onto the server that runs JIRA.
 - ⚠ This is a third-party application and not supported by Atlassian.
2. Ensure the <JAVA_HOME> variable is pointing to the same version of Java that JIRA uses. See our [Setting JAVA_HOME](#) docs for further information on this.
 - ℹ If running on a Linux/UNIX server, X11 will need to be forwarded when connecting to the server (so you can use the GUI), as below:

```
ssh -X user@server
```

3. Select the **Examine** menu and then click **Examine SSL/TLS Connection**:
4. Enter the SSL Host and Port of the target system:
5. Wait for it to load, then select the public certificate and click on PEM:
6. Export the certificate and save it.
7. Go back to the main screen and select the **Open an existing keystore from disk** option, select cacerts (for example \$JAVA_HOME/lib/security/cacerts) then enter the password (the default is change it).
8. Select the **Import a trusted certificate into the loaded keystore** button:

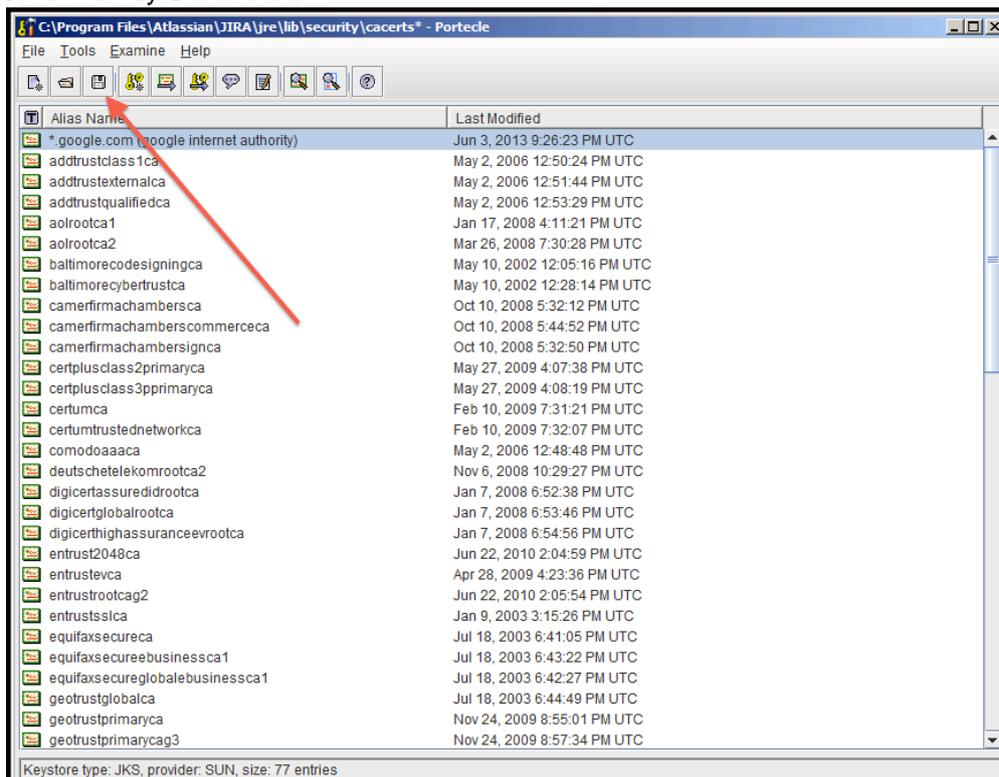


9. Select the certificate that was saved in step 6 and confirm that you trust it, giving it an appropriate alias (e.g.: confluence).
 - a. You may hit this error:



b. If so, hit OK, and then accept the certificate as trusted.

10. Save the Key Store to disk:



11. Restart JIRA.

12. Test that you can connect to the host.

Command Line Installation

1. Fetch the certificate, replacing `google.com` with the FQDN of the server JIRA is attempting to connect to:

Unix:

```
openssl s_client -connect google.com:443 < /dev/null | sed
-ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' >
public.crt
```

Windows:

```
openssl s_client -connect google.com:443 < NUL | sed -ne
' /-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' > public.crt
```

i The command above will only be executed if you have [Sed for Windows](#) as well as [OpenSSL](#) installed on your environment. If you don't have Sed or OpenSSL or you don't want to install it, use the instructions below as an alternative. Issue the following command:

```
openssl s_client -connect google.com:443
```

Save the output to a file called `public.crt`. Edit the `public.crt` file so it contains only what is between the `BEGIN CERTIFICATE` and `END CERTIFICATE` lines. This is how your file should look like after you edited it:

```
-----BEGIN CERTIFICATE-----
< Certificate content as fetched by the command line.
Don't change this content, only remove what is before
and after the BEGIN CERTIFICATE and END CERTIFICATE.
That's what your Sed command is doing for you :-) >
-----END CERTIFICATE-----
```

2. Import the certificate:

```
<JAVA_HOME>/keytool -import -alias <server_name> -keystore
<JAVA_HOME>/lib/security/cacerts -file public.crt
```

Alternative KeyStore Locations

Java will normally use a system-wide keystore in `$JAVA_HOME/jre/lib/security/cacerts`, but it is possible to use a different keystore by specifying a parameter, **`-Djavax.net.ssl.trustStore=/path/to/keystore`**, where `'/path/to/keystore'` is the absolute file path of the alternative keystore.

However, setting this **is not recommended** because if Java is told to use a custom keystore (eg. containing a self-signed certificate), then Java will not have access to the root certificates of signing authorities found in `$JAVA_HOME/jre/lib/security/cacerts`, and accessing most CA-signed SSL sites will fail. It is better to add new certificates (eg. self-signed) to the system-wide keystore (as above).

Debugging

Problems are typically one of two forms:

- The certificate was installed into the incorrect keystore.

- The keystore does not contain the certificate of the SSL service you're connecting to.

See Also

- [Configuring an SSL Connection to Active Directory](#)
- [Running JIRA applications over SSL or HTTPS](#)
- [Integrating JIRA with Apache using SSL](#)

Running JIRA applications over SSL or HTTPS

You can use SSL with Atlassian applications; however, SSL configuration is outside the scope of Atlassian Support.

- If you need help with converting certificates, consult with the vendor who provided the certificate.
- If you need help with configuring SSL, create a question on [the Atlassian Community](#).

Note that SHA-1 is being [phased out](#) due to known weaknesses.

The instructions on this page describe how to run JIRA applications over SSL or HTTPS by configuring Apache Tomcat with HTTPS. This procedure only covers the common installation types of JIRA. It is by no means a definitive or comprehensive guide to configuring HTTPS and may not apply to your environment.

Why should you run JIRA over SSL or HTTPS? When people access web applications, there is always a possibility that their usernames and passwords can be intercepted by intermediaries between your computer and the ISP/company. It's a good idea to enable access via HTTPS (HTTP over SSL) and make this a requirement for pages where passwords are sent. Note, however, that using HTTPS may result in slower performance.

Running JIRA without HTTPS enabled may leave your instance exposed to vulnerabilities, such as Man in the middle or DNS Rebinding attacks. We recommend that you enable HTTPS on your instance.

Before you begin

Please note the following before you begin:

- **Support**
Atlassian Support will refer SSL support to the Certificate Authority (CA) that issues the Certificate. The SSL-related instructions on this page are provided as a reference only.
- **Windows installers**
The '[Windows Installer](#)' installs its own Java Runtime Environment (JRE) Java platform, which is used to run Tomcat. When updating SSL certificates, please do so in this JRE installation.
- **Related bugs**
JIRA 7.3 and later is affected by two bugs that incorrectly set the protocol in the `server.xml` file. You can work around this issue by setting the protocol manually.
 - ▾ [Tell me more...](#)
Two bugs affecting JIRA 7.3.0 and later:

JRASERVER-63734 - Tomcat will not start in 7.3 with protocol="org.apache.coyote.http11.Http11Protocol"
CLOSED

JRASERVER-64082 - JIRA Configuration Tool is setting wrong value of the "protocol" attribute for Tomcat SSL configuration CLOSED

The workaround is to manually edit the `server.xml` file to change the protocol on your HTTPS connector to:

```
protocol="org.apache.coyote.http11.Http11NioProtocol"
```

- **JIRA behind a reverse-proxy**

If hosting JIRA behind a reverse-proxy, such as Apache, please see [Integrating JIRA with Apache using SSL](#) for more information.

Generate the Java KeyStore

In this section, you will create a Java Key Store (JKS), which will hold your SSL certificates. The SSL certificates are required for SSL to work in JIRA. In the SSL world, certificates fall into two major categories:

Certificate	Description	When to use	Steps
Self-signed	These are certificates that have not been digitally signed by a CA, which is a method of confirming the identity of the certificate that is being served by the web server. They are signed by themselves, hence the name self-signed.	Test, dev or internal servers only .	1 - 13
CA-signed	A certificate that has had its identity digitally signed by a Certificate Authority (CA). This will allow browsers and clients to trust the certificate.	Production servers.	1 - 21

Digital Certificates that are issued by trusted 3rd party CAs (Certification Authority) provide verification that your Website does indeed represent your company, thereby verifying your company's identity. Many CAs simply verify the domain name and issue the certificate. Other CAs, such as [VeriSign](#), verify the existence of your business, the ownership of your domain name, and your authority to apply for the certificate, providing a higher standard of authentication.

A list of CA's can be found [here](#). Some of the most well known CAs are:

- [Verisign](#)
- [Thawte](#)
- [CAcert](#) (relatively new CA, providing free CA certificates)

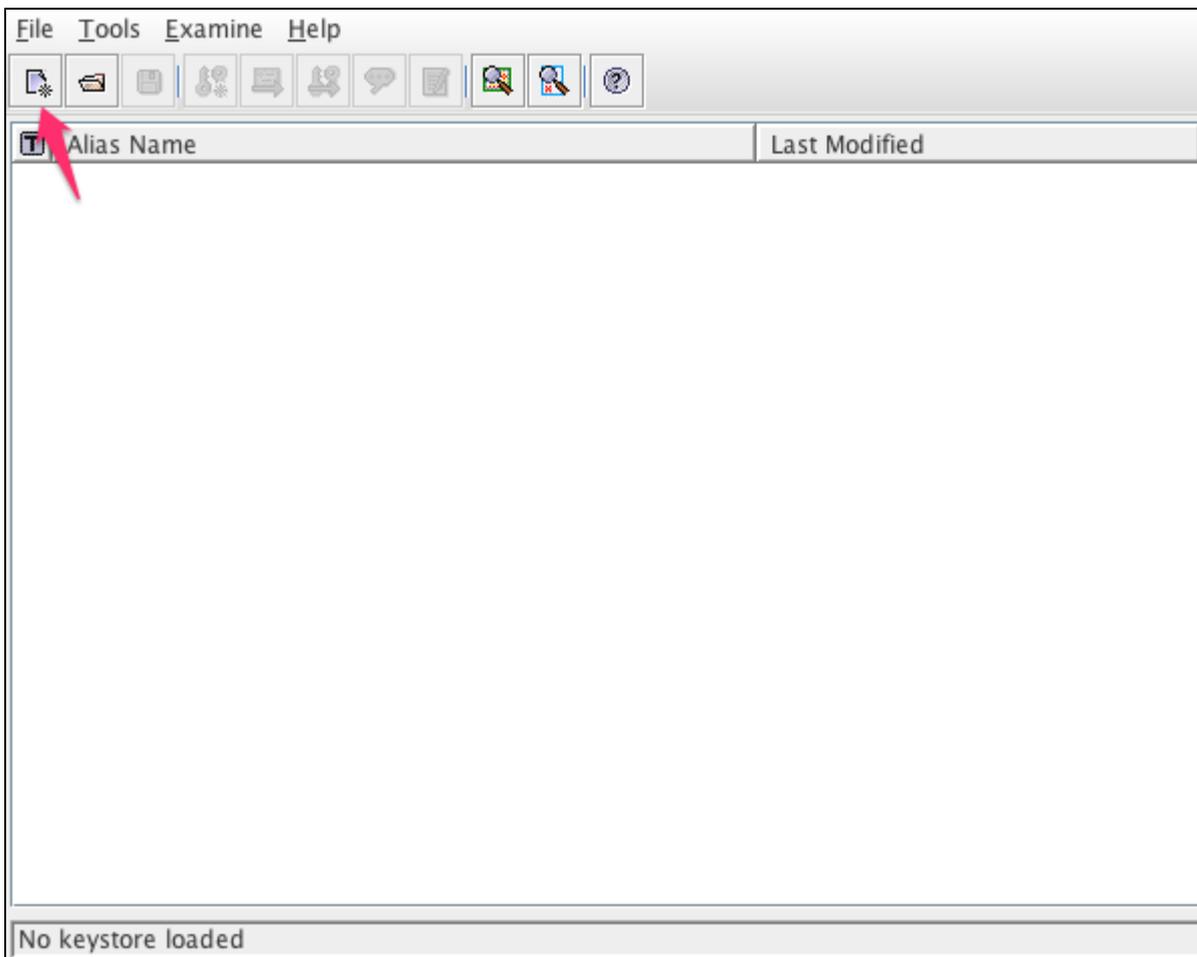
We recommend using a CA-signed certificate.

If you're unable to install Portecle on the server or prefer the command line, please see our [Command Line Installation](#) section below.

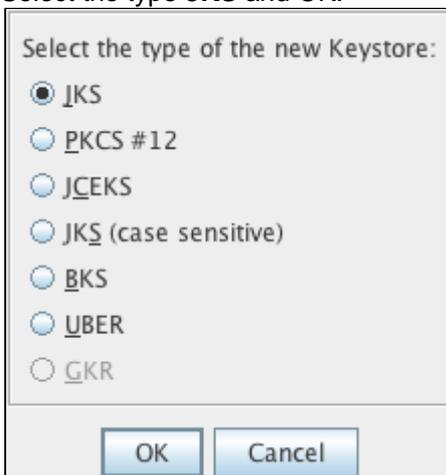
1. Download and install the [Portecle](#) app onto the server that runs JIRA.
 - ⚠ This is a third-party application and is not supported by Atlassian.
2. Run the App as an Administrator, so it will have the appropriate permissions. Also, ensure the `<JAVA_HOME>` variable is pointing to the same version of Java that JIRA uses. See [Setting JAVA_HOME](#) for further information on this.
 - ℹ If running on a Linux/UNIX server, X11 will need to be forwarded when connecting to the server (so you can use the GUI), as below:

```
ssh -X user@server
```

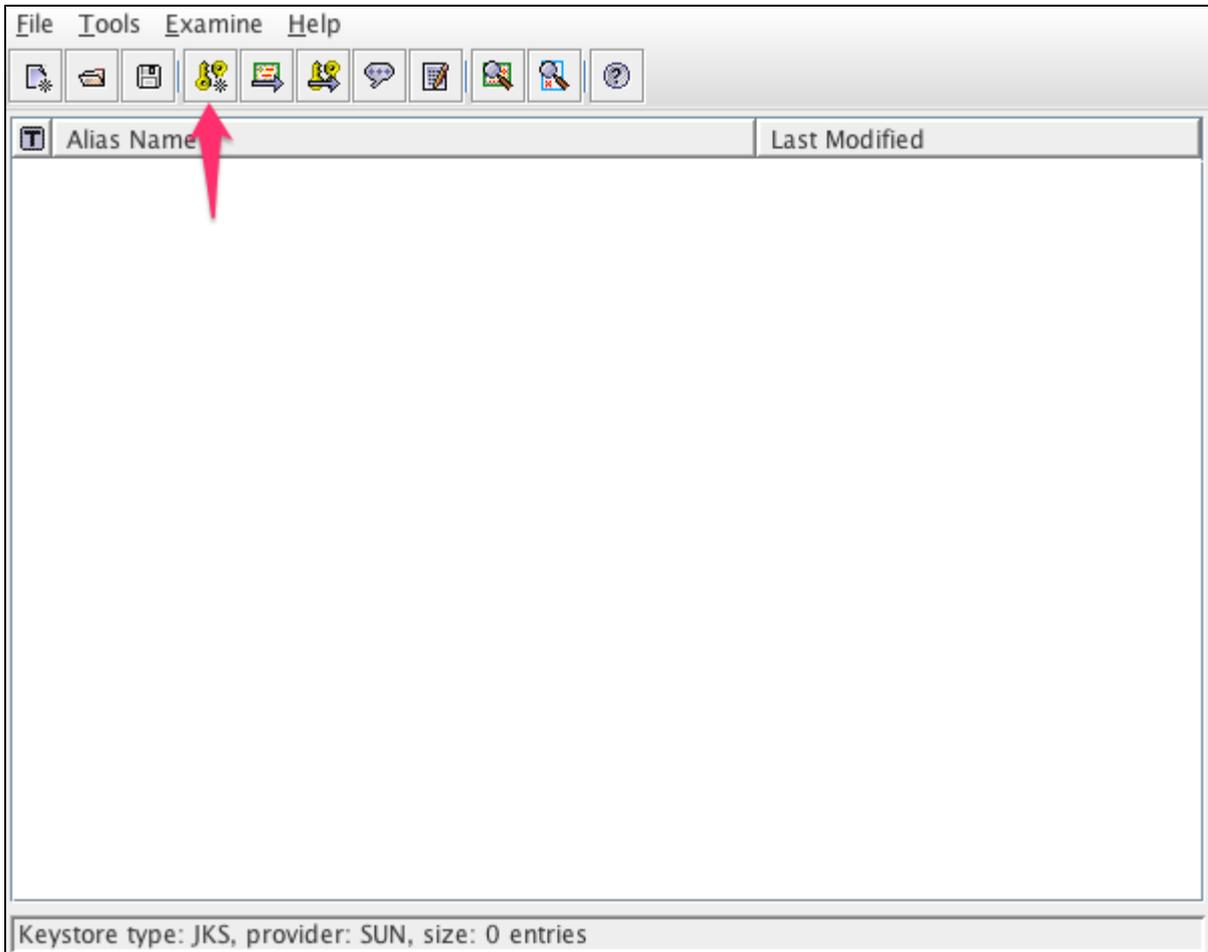
3. Select the **Create a new Keystore** option:



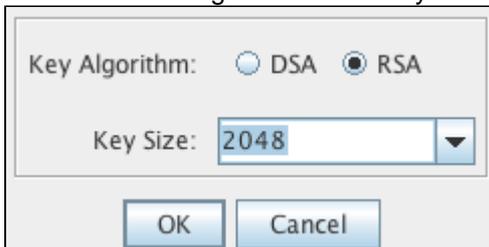
4. Select the type **JKS** and OK:



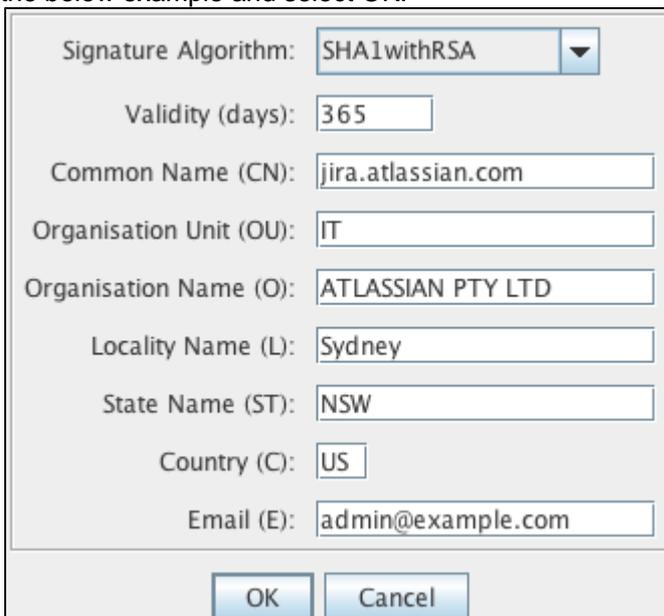
5. Select the **Generate Key Pair** button:



6. Select the RSA algorithm and a Key Size of 2048:



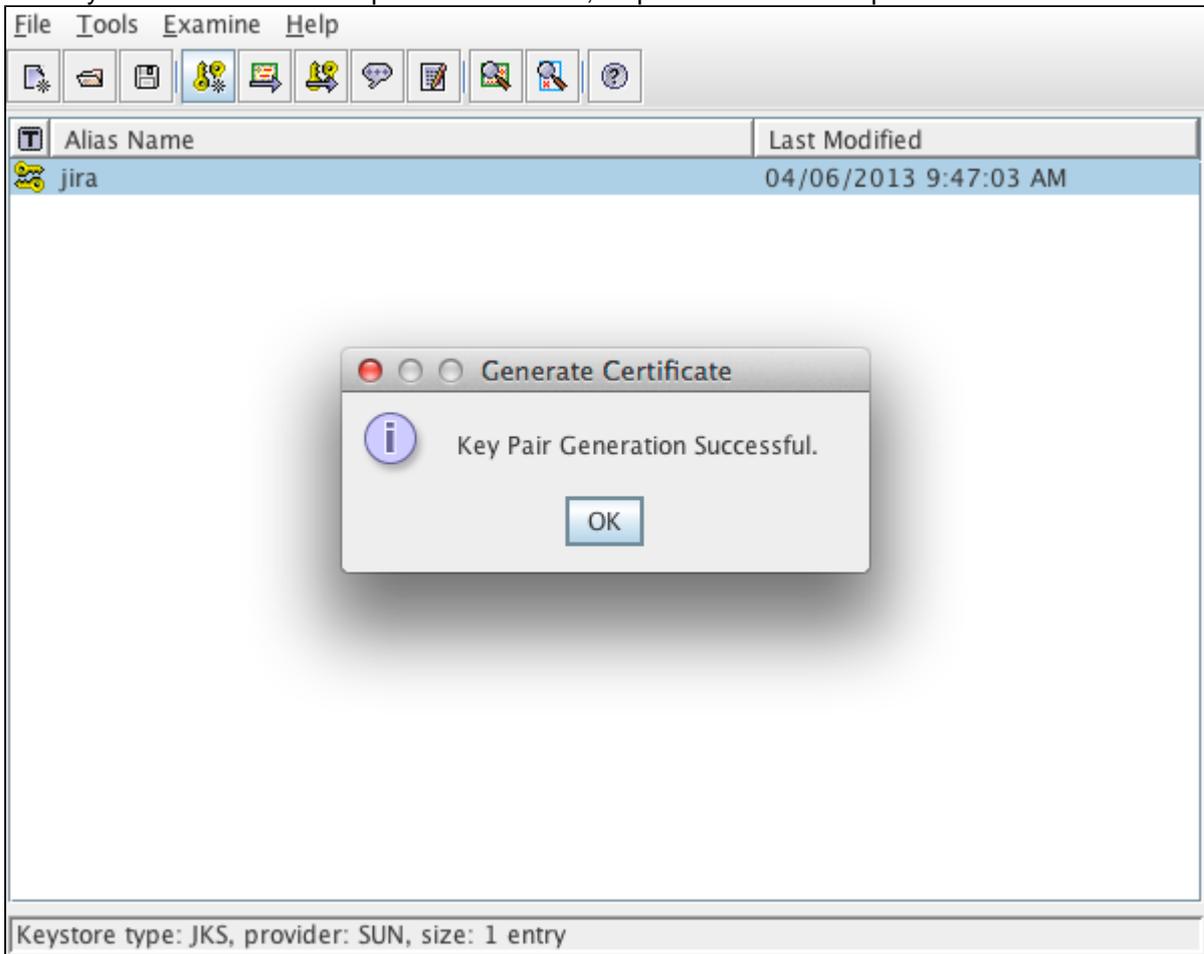
7. Make sure the **Signature Algorithm** is "SHA1withRSA" and then edit the certificate details, as per the below example and select OK:



⚠ The **Common Name** MUST match the server's URL, otherwise errors will be displayed in the browser.

ℹ If you would like to use SHA256withRSA, please use the appropriate Signature Algorithm, and refer to: [Security tools report the default SSL Ciphers are too weak](#).

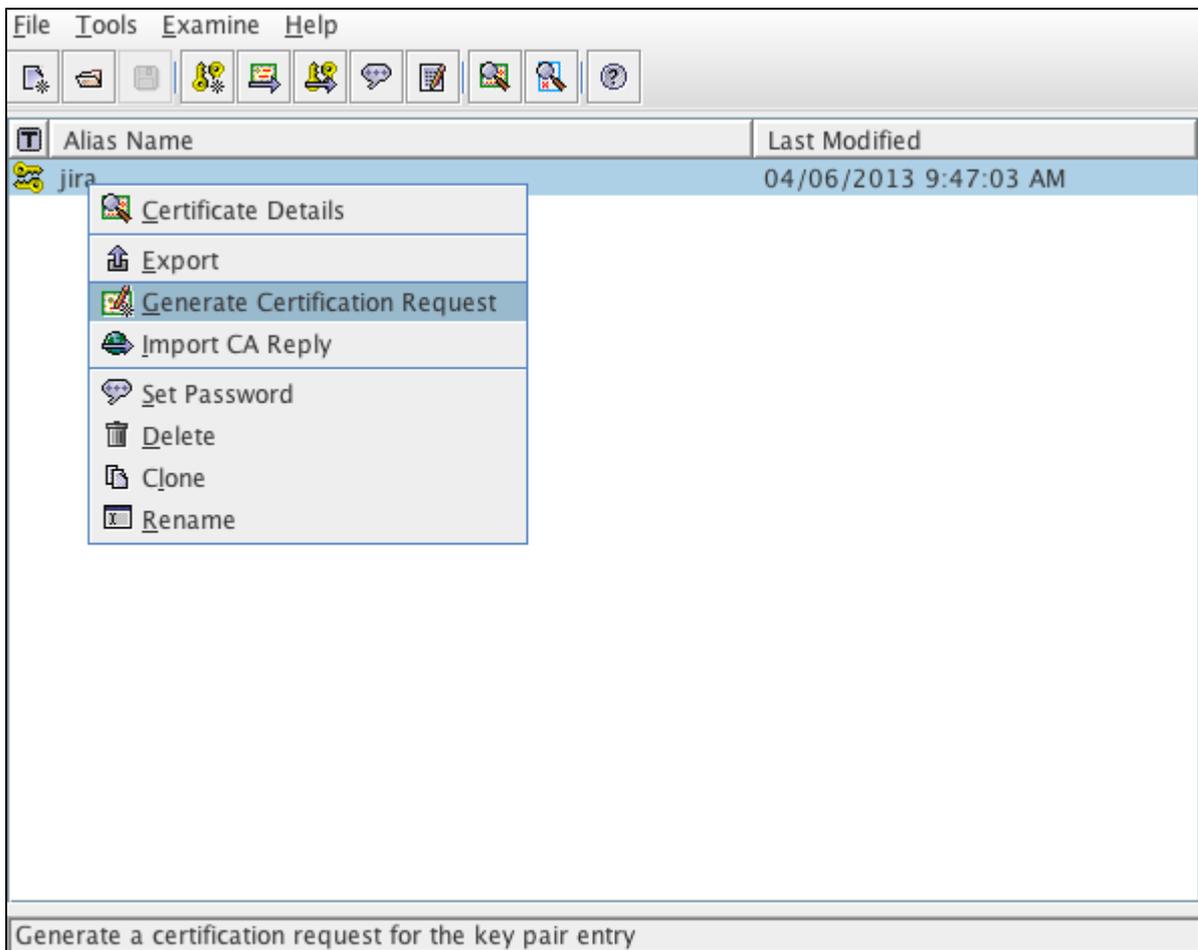
8. Choose an alias for the certificate - for example jira.
9. Enter a password for the KeyStore (the default password used is typically `changeit`).
10. The Key Pair Generation will report as successful, as per the below example:



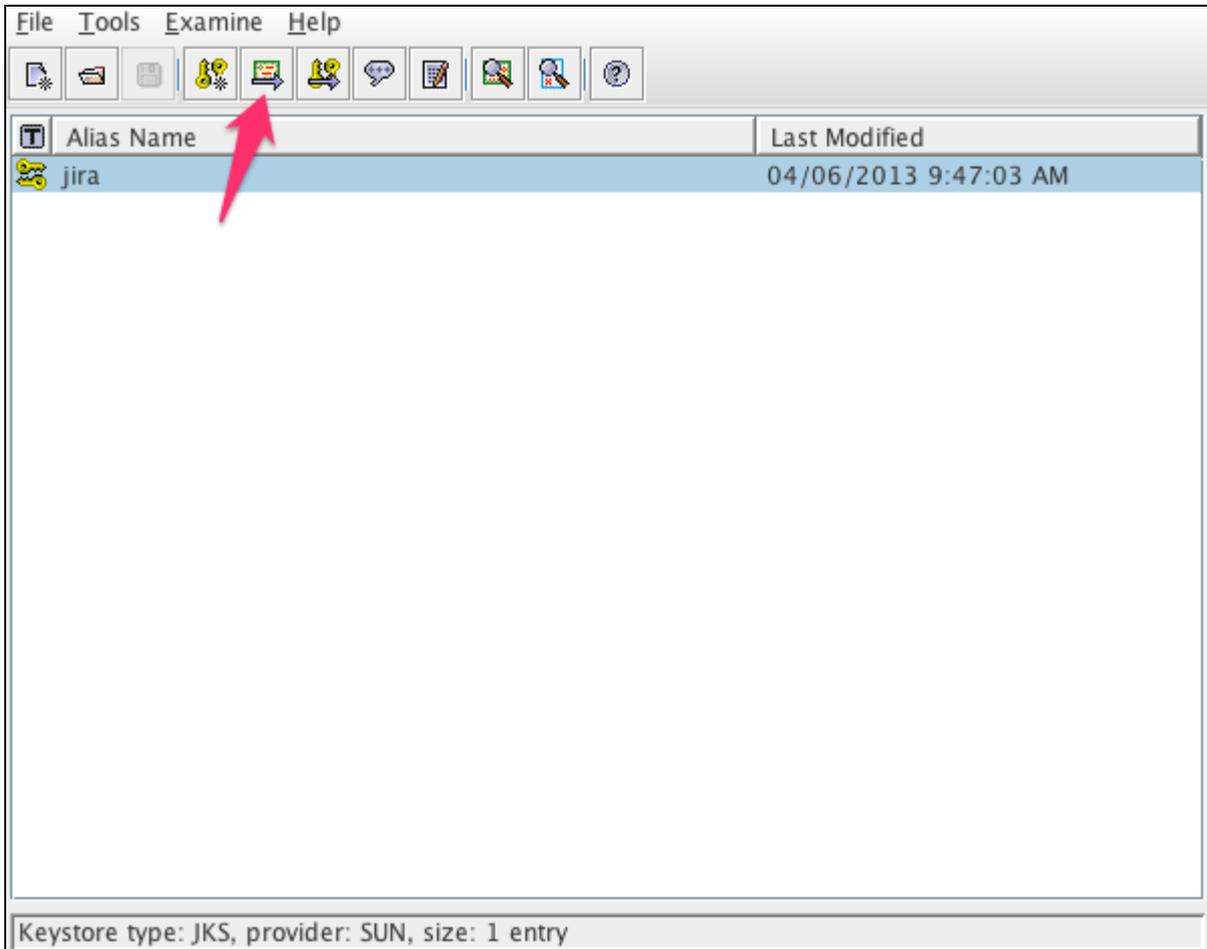
11. Save the KeyStore in `<JIRA_HOME>/jira.jks`, ensuring the use the same password in step 11. This can be done by **File > Save Keystore**.

If using a self-signed certificate certificate, proceed to [Configuring your web server using the JIRA configuration tool](#). Otherwise, continue on.

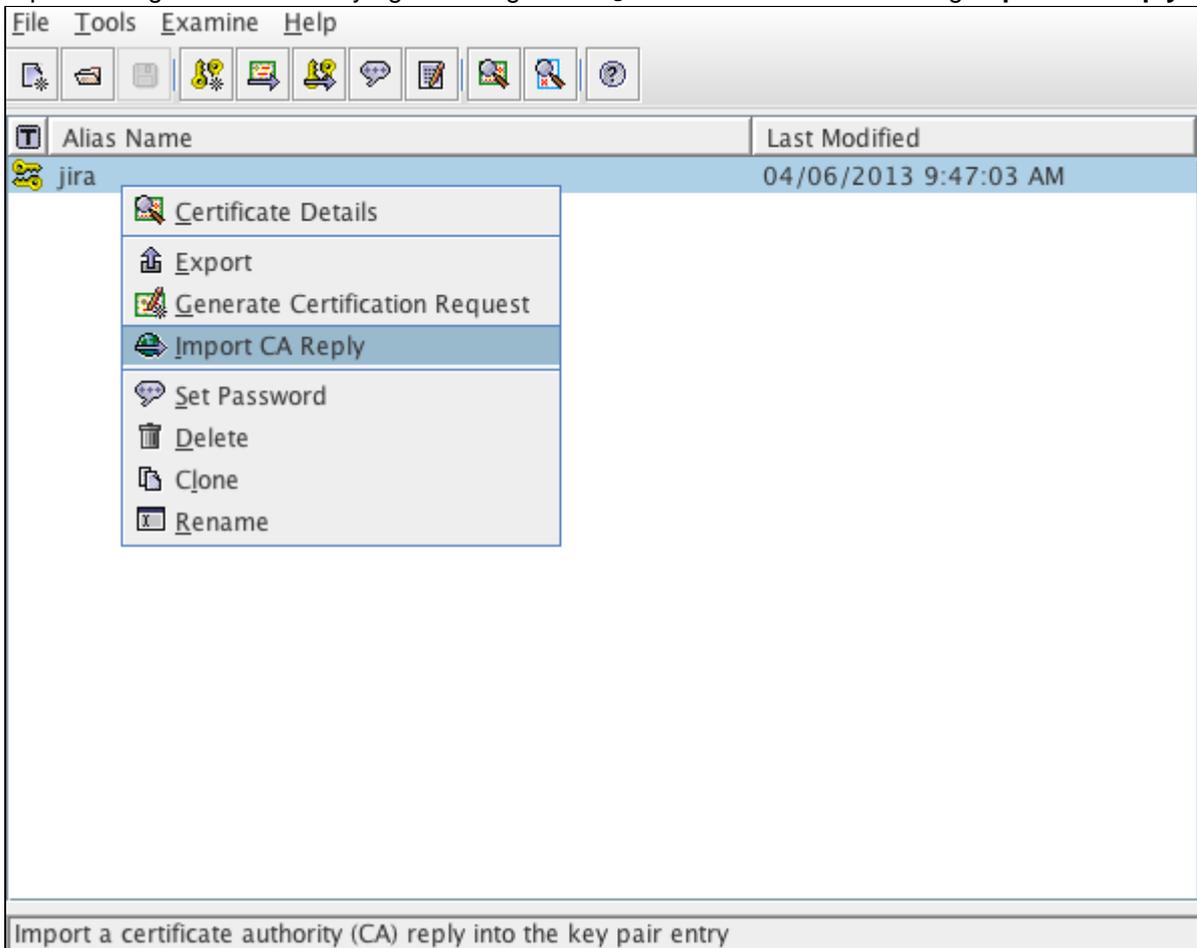
12. We need to generate a Certificate Signing Request for the CA to sign and confirm the identity of the certificate. To do so, right click on the certificate and choose **Generate CSR**. Save it in `<JIRA_HOME>/jira.csr`.



13. Submit the CSR to a Certificate Authority for signing. They will provide a signed certificate (CA reply) and a set of root/intermediate CA certificates.
14. Import the root and/or intermediate CA certificates with **Import Trusted Certificate**, repeating this step for each certificate.



15. Import the signed certificate by right clicking on the `jira` certificate and selecting **Import CA Reply**:



16. Select the certificate provided by the CA, which should be `jira.crt`. This will respond with CA Reply

Import successful.

17. Verify this by checking **Tools > Keystore Report**. It should display the certificate as a child of the root certificates.
18. Save the KeyStore and proceed to the next section.

Configuring your web server using the JIRA configuration tool

In this section, you will finish setting up SSL encryption for JIRA, by configuring your web server using the JIRA configuration tool. For more information on the JIRA configuration tool, see [Using the JIRA configuration tool](#).

1. Run the JIRA configuration tool, as follows:
 - **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the JIRA installation directory.
 - **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the JIRA installation directory.
 - ⓘ This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.
2. Click the **Web Server** tab.

Screenshot: JIRA configuration tool — 'Web Server' tab

The screenshot shows the 'JIRA Configuration Tool' window with the 'Web Server' tab selected. The configuration fields are as follows:

Field	Value
Control Port	8005
Profile	HTTP and HTTPS (redirect HTTP to HTTPS)
HTTP Port	8080
HTTPS Port	8443
Keystore Path	C:\Program Files\Atlassian\JIRA\jira.jks
Keystore Password	changeit
Key Alias	jira

Buttons: Browse, Check Certificate in Key Store, Save, Close.

3. Fill out the fields as follows:

Field	Value
Control Port	Leave as default. You can change the port number if you wish. See Changing JIRA's TCP ports .

Profile	<p>A profile is a preset web server configuration. You can pick from the four following values:</p> <ul style="list-style-type: none"> • Disabled • HTTP only • HTTP & HTTPS (redirect HTTP to HTTPS) • HTTPS only <p>To run JIRA over HTTPS, you must pick either 'HTTP & HTTPS' or 'HTTPS'. Pick 'HTTP & HTTPS' if you want to run JIRA over HTTPS but you have users that access JIRA via HTTP. If you pick 'HTTP & HTTPS', users who try to access JIRA via HTTP will be redirected to the HTTPS address.</p>
HTTP port	<p>Leave as default (8080). You can change the port number if you wish. See Changing JIRA's TCP ports .</p> <p>This will be disabled if you set the Profile to 'HTTPS only'.</p>
HTTPS port	<p>Leave as default (8443). You can change the port number if you wish. See Changing JIRA's TCP ports .</p>
Keystore path	<p>Specify the location of the keystore of your certificate. This will have been chosen when the keystore was saved in step 13 and should be <code><JIRA_HOME>/jira.jks</code>.</p>
Keystore password	<p>Specify the password for your keystore. If you generated a self-signed certificate, this is the password you specified for the key and keystore when generating the certificate in step 13.</p>
Keystore alias	<p>Each entry in the keystore is identified by an alias. We recommend using <code>jira</code> for this certificate as in step 10.</p>

4. Click the **Check Certificate in Key Store** button to validate the following:
 - Test whether the certificate can be found in the key store.
 - Test whether keystore password works.
 - Test whether key can be found using key alias.
5. Click the **Save** button to save your changes.

Advanced configuration

Running more than one instance on the same host

When running more than one instance on the same host, it is important to specify the *address* attribute in the `<JIRA_INSTALLATION>/conf/server.xml` file because by default the connector will listen on **all available network interfaces**, so specifying the address will prevent conflicts with connectors running on the same default port. See the Tomcat Connector documentation for more about setting the address attribute in [The HTTP Connector Apache Tomcat docs](#).

Command Line Installation

Step 1. Create the KeyStore

1. Generate the Java KeyStore:

```
<JAVA_HOME>/keytool -genkey -alias jira -keyalg RSA -keystore
<JIRA_HOME>/jira.jks
```

-  Instead of first and last name, enter the server URL, excluding "https://" (e.g.: `jira.atlassian.com`).
2. Enter a password.
 3. Create the CSR for signing, using the password from step 2:
 4. Submit the CSR to the CA for signing. They will provide a signed certificate and a root and/or intermediate CA.
-  If the certificate will not be signed, skip to step 7.

5. Import the root and/or intermediate CA:

```
<JAVA_HOME>/keytool -import -alias rootCA -keystore
<JIRA_HOME>/jira.jks -trustcacerts -file root.crt
```

6. Import the signed certificate (the CA provides this):

```
<JAVA_HOME>/keytool -import -alias jira -keystore
<JIRA_HOME>/jira.jks -file jira.crt
```

7. Verify the certificate exists within the KeyStore:

```
<JAVA_HOME>/keytool -list -alias jira -keystore
<JIRA_HOME>/jira.jks
```

This must be a `PrivateKeyEntry`, if it is not the certificate setup has not successfully completed. For example:

```
jira, Jan 1, 1970, PrivateKeyEntry,
Certificate fingerprint (MD5):
73:68:CF:90:A8:1D:90:5B:CE:2A:2F:29:21:C6:B8:25
```

Step 2. Update Tomcat with the KeyStore

1. Create a backup of `<JIRA_INSTALL>/conf/server.xml` before editing it.
2. Edit the HTTPS connector so that it has the parameters that point to the KeyStore:

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxHttpHeaderSize="8192" SSLEnabled="true"
    maxThreads="150" minSpareThreads="25"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    sslEnabledProtocols="TLSv1.2,TLSv1.3"
    clientAuth="false" useBodyEncodingForURI="true"
    keyAlias="jira"
keystoreFile="<JIRA_HOME>/jira.jks" keystorePass="changeit"
keystoreType="JKS"/>
```

 Ensure to put the appropriate path in place of `<JIRA_HOME>` and change the port as needed.

If the organization doesn't support the latest TLS version, you can fallback to an earlier version. Change:

```
sslEnabledProtocols="TLSv1.2,TLSv1.3"
```

To:

```
sslEnabledProtocols="TLSv1,TLSv1.1,TLSv1.2,TLSv1.3"
```

3. Edit the HTTP connector so that it redirects to the HTTPS connector:

```
<Connector acceptCount="100" connectionTimeout="20000"
disableUploadTimeout="true" enableLookups="false"
maxHttpHeaderSize="8192" maxThreads="150" minSpareThreads="25"
port="8080" protocol="HTTP/1.1"
redirectPort="<PORT_FROM_STEP_1>" useBodyEncodingForURI="true"/>
```

i Ensure the `<PORT_FROM_STEP_1>` is change to the appropriate value. In this example it would be 8443.

4. Save the changes to `server.xml`.
5. If redirection to HTTPS will be used (this is recommended), edit the `<JIRA_INSTALL>/WEB-INF/web.xml` file and add the following section at the end of the file, before the closing `</web-app>`. In this example, all URLs except attachments are redirected from HTTP to HTTPS.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>all-except-attachments</web-resource-name>
    <url-pattern>*.jsp</url-pattern>
    <url-pattern>*.jspx</url-pattern>
    <url-pattern>/browse/*</url-pattern>
    <url-pattern>/issues/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

6. Restart JIRA after you have saved your changes.

You can also redirect users from HTTP URLs to HTTPS URLs by choosing the 'HTTP & HTTPS' profile in the JIRA configuration tool. This will redirect all HTTP URLs to HTTPS URLs.

If you want to only redirect certain pages to HTTPS, you need to do this manually. To do this, select the 'HTTPS only' profile in the JIRA configuration tool and save the configuration, and then create an `htaccess` file on your web server that will manually redirect the HTTP URLs to the corresponding HTTPS URLs.

Troubleshooting

Here are some troubleshooting tips if you are using a self-signed key created by Portecle, as described above.

When you enter `"https://localhost:<port number>"` in your browser, if you get a message such as "Cannot

establish a connection to the server at localhost:8443", look for error messages in your logs/catalina.out log file. Here are some possible errors with explanations.

▼ [Click here to expand...](#)

- **SSL + Apache + IE problems:** Some people have reported errors when uploading attachments over SSL using IE. This is due to an IE bug, and can be fixed in Apache by setting:

```
BrowserMatch ".MSIE." \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
```

[Google](#) has plenty more on this.

- **Can't find the keystore:**

```
java.io.FileNotFoundException: /home/user/.keystore (No such
file or directory)
```

This indicates that Tomcat cannot find the keystore. The keytool utility creates the keystore as a file called .keystore in the current user's home directory. For Unix/Linux the home directory is likely to be /home/<username>. For Windows it is likely to be C:\Documents And Settings\<UserName>.

Make sure you are running JIRA as the same user who created the keystore. If this is not the case, or if you are running JIRA on Windows as a service, you will need to specify where the keystore file is in conf/server.xml. Add the following attribute to the connector tag you uncommented:

```
keystoreFile="<location of keystore file>"
```

This can also happen ("Cannot find /root/.keystore") if you add a keystoreFile attribute to the http connector in server.xml instead of the https connector.

- **Certificate reply and certificate in keystore are identical:**

```
keytool error: java.lang.Exception: Certificate reply and
certificate in keystore are identical
```

This error will happen if you have identical names or fingerprints, which is the result of attempting to recreate the cert in your existing keystore. If you need to recreate or update the Cert, you may remove the existing keystore and creating a fresh, new keystore. In this case, creating a new keystore and adding the related certs will fix the issue. The default path for it in this documentation is \$JAVA_HOME/jre/lib/security/cacerts

- **Incorrect password:**

```
java.io.IOException: Keystore was tampered with, or password
was incorrect
```

You used a different password than "changeit". You must either use "changeit" for both the keystore password and for the key password for Tomcat, or if you want to use a different password, you must specify it using the keystorePass attribute of the Connector tag, as described above.

- **Passwords don't match:**

```
java.io.IOException: Cannot recover key
```

You specified a different value for the keystore password and the key password for Tomcat. Both passwords must be the same.

- **Wrong certificate:**

```
javax.net.ssl.SSLException: No available certificate
corresponds to the SSL cipher suites which are enabled.
```

If the Keystore has more than one certificate, Tomcat will use the first returned unless otherwise specified in the SSL Connector in `conf/server.xml`.

Add the `keyAlias` attribute to the Connector tag you uncommented, with the relevant alias, for example:

```
<Connector port="8443" maxHttpHeaderSize="8192"
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" disableUploadTimeout="true"
useBodyEncodingForURI="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="/opt/local/.keystore"
keystorePass="removed"
keyAlias="tomcat" />
```

- **Using Apache Portable Runtime:**

APR uses a different SSL engine, and you will see an exception like this in your logs

```
SEVERE: Failed to initialize connector
[Connector[HTTP/1.1-8443]]
LifecycleException: Protocol handler initialization failed:
java.lang.Exception: No Certificate file specified or invalid
file format
```

The reason for this is that the APR Connector uses OpenSSL and cannot use the keystore in the same way. You can rectify this in one of two ways:

- Use the `Http11NioProtocol` to handle SSL connections — Edit the `server.xml` so that the SSL Connector tag you just uncommented specifies the `Http11NioProtocol` instead of the APR protocol

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
maxHttpHeaderSize="8192" SSLEnabled="true"
keystoreFile="${user.home}/.keystore"
maxThreads="150" enableLookups="false"
disableUploadTimeout="true"
acceptCount="100" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
useBodyEncodingForURI="true" />
```

- Configure the Connector to use the APR protocol — This is only possible if you have PEM encoded certificates and private keys. If you have used OpenSSL to generate your key, then you will have these PEM encoded files - in all other cases contact your certificate provider for assistance.

```
<Connector
  port="8443" maxThreads="200"
  scheme="https" secure="true" SSLEnabled="true"
  SSLCertificateFile="{user.home}/certificate.pem"
  SSLCertificateKeyFile="{user.home}/key.pem"
  clientAuth="optional" SSLProtocol="TLSv1.2,TLSv1.3"/>
```

- **Enabling Client Authentication:** To enable client authentication in Tomcat, ensure that the value of the `clientAuth` attribute in your `Connector` element of your Tomcat's `server.xml` file is `true`.

```
<Connector
  ...
  clientAuth="true"
  ... />
```

For more information about `Connector` element parameters, please refer to the [SSL Configuration HOW-TO Tomcat 8](#) documentation.

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 JIRA's internal/bundled H2 database), you should restrict access to the database that your JIRA instance uses.
 - If you are using JIRA's internal/bundled H2 database, 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.)
- SSL — if you are running your JIRA instance over the Internet, you may want to consider using SSL.
- 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](#)

Other security resources

- [User management](#)
- [Securing JIRA applications with Apache HTTP Server](#)
- [JIRA application cookies](#)
- [Configuring permissions](#)

Data collection policy

Why does JIRA collect usage data?

We're proud that JIRA is one of the most advanced and configurable issue trackers on the planet and we will continue to deliver innovative new features as quickly as we can. In order to prioritize the features we deliver, we need to understand how our customers use JIRA, what's important, what's not, and what doesn't work well. The collection of usage data allows us to measure the user experience across many thousands of users and deliver features that matter.

What data is collected?

The type of data we collect is covered in our [Privacy Policy](#). Please read it, as we've tried to avoid legal jargon and make it as straightforward as possible.

To view a sample of data that might be collected from your specific installation:

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose  **> System > Advanced > Analytics**.
3. Select the **Sample Data** link.

How is data collected from Server installations?

Analytics are collected using the Atlassian Analytics add-on. The add-on collects analytics events in a log file which is located in the JIRA home directory under the analytics-logs sub directory. The logs are periodically uploaded using an encrypted session and then deleted. If the JIRA installation is unable to connect to the Internet, no logs are ever uploaded.

Enabling/disabling data collection in JIRA Server

You can switch off analytics collection at any time:

1. Log in as a user with the **JIRA Administrators** [global permission](#).
2. Choose  **> System > Advanced > Analytics**.
3. Select **Disabled**, and **Save** your change.

JIRA Admin Helper

The JIRA Admin Helper is a **free, bundled plugin** that answers questions like:

- Why isn't my field showing up on view/edit/create screens?
- Why can/can't a user see a certain issue?
- Why did/didn't a user get a certain email notification?

 The JIRA Admin Helper plugin is visible only to JIRA Administrators. When you are viewing an issue, it is available from the **Admin** menu.

On this page:

- [Field Helper](#)
- [Permission Helper](#)
- [Notification Helper](#)

Field Helper

If you're logged in as a JIRA administrator, you can use the Field Helper – displayed as a *Where is my field?* link – to help you determine why a field is not appearing on a specific screen. The Field Helper works with custom fields as well as JIRA system fields.

The *Where is my field?* link is available on:

- Create Issue – in *Configure Fields* pop up
- Edit issue - in *Configure Fields* pop up
- View Issue- in *More Actions* menu
- Issue Navigator – in cog menu

Simply click on the link and then enter the field name in the search box!

Here's an example:

Edit Issue : ANGRY-306

Configure Fields

Priority: Minor

Due Date: 31/Mar/13

Labels:

Component/s:

Fix Version/s:

Assignee: Unassigned

Comment:

Show Fields: All Custom Where is my field?

Assignee
 Attachments
 Browser
 Business Value
 Comment
 Component/s
 Database
 Description
 Developer
 Due Date
 Environment
 Epic Link
 External issue ID
 External issue URL
 Fix Version/s
 Issue Type*

Labels
 Priority
 QA Completed On
 Reporter*
 Reviewer
 Scheduled Deployment
 Severity
 Sprint
 Story Points
 Summary*
 Test Deployment Date
 Test Session(s)
 Time Tracking

Click here to open the Field Helper

After you enter the name of the missing field, the Field Helper returns a form that explains why this field is not appearing:

Where is my field?

Begin typing to find your field

Project: Angry Nerds
 Issue type: Bug
 Screen: Edit Issue
 Field: Need Docs?
 Status: ✘

The 'Need Docs?' field is not present on the form you are viewing

Status	Summary	Details
✔	Field configuration	The field 'Need Docs?' is enabled by the Field Configuration 'Angry Nerds Field Configuration' associated with this issue.
✔	Field Screen	The field 'Need Docs?' is present on the screen 'Default Screen' configured for this issue
✘	Project and issue type scope	The field 'Need Docs?' is not configured in scope of the project 'Angry Nerds' and issue type 'Bug' To solve this issue, go to 'Need Docs?' configuration page and add it to the scope

Close

You can then use this information to fix your screen by adding this field to your project and issue type.

Permission Helper

The JIRA Admin Helper can help you diagnose why a user can or cannot see a certain issue.

1. Choose  > **System**.
2. Then choose **Admin Helper > Permission Helper**.
3. Enter the username of the user (leave blank for anonymous users), an issue key (for example, an issue that the user can/cannot see) and the permission to check.
4. Click **Submit**.

Permission Helper

Discover why a user does or does not have certain permissions...

User

Begin typing to find a user, leave blank for Anonymous user

Permission

Begin typing to find a permission or press down to see all

Notification Helper

The Notification Helper can you help figure out why a user didn't get an email notification when a comment was added. It's available from the view issue page, the issue navigator, and from JIRA Administration.

1. Choose  > **System**.
2. Then choose **Admin Helper > Notification Helper**.
3. Enter the username of the user (leave blank for anonymous users) and select the Notification Event from the drop-down list.
4. Click **Submit**.

Notification Helper

Find out why users receive, or do not receive notifications for this issue

User

Begin typing to find a user

Notification Event

Begin typing to find a notification event or press down to see all

Raising support requests as an administrator

If you have a problem with your JIRA instance, there are a number of resources available to help you resolve it. We recommend that you try searching our knowledge base and our customer forums for an answer first. This is often the fastest way to get a problem resolved.

- [JIRA knowledge base](#)
- [Atlassian Answers](#)

If you can't find what you need, the next step is to raise a support request, as described below.

On this page:

- [Before you begin](#)
- [Raising a support request in JIRA](#)
- [Creating a support zip](#)
- [Providing logs when you cannot log in to JIRA](#)

Before you begin

The functionality described on this page is enabled by the Support Tools Plugin, which is bundled with JIRA. This functionality is only available to users with the **JIRA System Administrators** global permission.

If you do not have this permission, you can still raise support requests on our [support site](#). You'll need to provide as much information as possible, including:

- Any error messages that are appearing on the console or in the logs (see the [logging](#) documentation)
- The operating system, database and version of JIRA you are using.

Raising a support request in JIRA

Raising a support request in JIRA gives you the option of including a range of system information with your request, rather than you having to manually describe it. This information helps our support team resolve your issue faster.

1. Log in as a user with the **JIRA System Administrators** global permission.
2. Click



> **System** > **Atlassian Troubleshooting and Support tools** (left menu) > **Create Support Request** (top tabs).

3. Fill out the **Create Support Request** form. Include as much information as possible to help our support team resolve your issue faster.

4. Click **Create**.

Once you've submitted your support request, we'll send you email updates about its progress.

Creating a support zip

The Atlassian Support team may need you to provide a support zip to help them understand and

troubleshoot your problem, after you have raised your initial request. The support zip contains logs from your instance and diagnostic and configuration information.

Note that the Support Tools Plugin sanitizes any usernames and passwords it finds in your configuration files, but does not sanitize usernames and content present in the log files.

1. Log in as a user with the **JIRA System Administrators** global permission.
2. Click



> **System** > **Atlassian Troubleshooting and Support tools** (left menu) > **Create Support Zip** (top tabs).

3. We recommend that you leaving all the boxes ticked on the form, but you can untick any options if you choose.

4. Click **Create**.

The Support Zip will be generated and can be found in your **JIRA HOME/export directory**. You can now provide this to the Atlassian Support team (usually by attaching the Support Zip to your support request).

Providing logs when you cannot log in to JIRA

If you are unable to log in to JIRA, you can still provide helpful diagnostic information to the Atlassian Support team. Create compressed files (zip or tar.gz) of the following log files and attach them to your support request:

- **Latest JIRA logs:** `$JIRA_HOME/log/atlassian-jira.log`
- **Application server (Tomcat) log files:**
 - UNIX: `$JIRA_INSTALL/logs/catalina.out`
 - Windows: `$JIRA_INSTALL/logs/stdout` and `stderr`

If you cannot locate these files, compress the contents of the following directories and attach them to your support request:

1. `$JIRA_INSTALL/logs`
2. `$JIRA_HOME/log`

Start and Stop JIRA applications

How you start and stop your JIRA application depends on whether you are running JIRA as a Service.

Windows

✓ When installed as a service...

If you installed JIRA as a service, you can **Start JIRA Server** and **Stop JIRA Server** from the Windows Start menu.

You can't start or stop JIRA manually using the `start-jira.bat` and `stop-jira.bat` file.

✓ Manually when not installed as a service...

If you didn't install JIRA as a service you'll need to start and stop JIRA manually.

- To start JIRA run `<installation-directory>\bin\start-jira.bat`
- To stop JIRA run `<installation-directory>\bin\stop-jira.bat`

We recommend running JIRA with a dedicated user account. To do this, use the `runas` command to

execute `start-jira.bat`.

```
> runas /env /user:<DOMAIN>\<jira> start-jira.bat
```

Where `<DOMAIN>` is your Windows domain or computer name and `<jira>` is the name of your dedicated user.

Manual start with add-ons disabled...

JIRA can be started with all non-system add-ons, or a selection of these add-ons, disabled. This helps with troubleshooting when these non-system add-ons are causing issues with your JIRA instance, such as causing JIRA to fail on start up, or when the add-on is malfunctioning and can't be removed through UPM. The parameters do not persist, that is, they are applied at start up once, and if you need to restart JIRA, you need to apply them again.

The parameters should be specified at system start up when JIRA is started using the `start-jira.bat` file, for example:

```
<installation-directory>/bin/start-jira.bat /disablealladdons  
/disableaddons=com.atlassian.test.plugin
```

If you don't use `start-jira.bat` for starting JIRA, but still wish to use this feature, you can add the following JVM parameter to the invocation of the servlet container that's running JIRA:

```
-Datlassian.plugins.startup.options="/disablealladdons  
/disableaddons=com.atlassian.test.plugin"
```

To disable **multiple plugins**, use a colon separated list of plugins. Regex/wildcards are not permitted, the full key of the plugin must be provided, for example:

```
-Datlassian.plugins.startup.options="/disablealladdons  
/disableaddons=com.atlassian.test.plugin:com.atlassian.another.test.p  
lugin"
```

Notes

- If the add-on key contains a space, this feature will not work, you need to [manually deal with that add-on](#).
- This feature does **not** work for JIRA Data Center applications.
- This can be used to disable an add-on deemed critical to JIRA starting, and if one of those is disabled, JIRA will fail to start
- This can be used to disable JIRA application OBR bundles, for example, to stop the JIRA Software add-on:

```
<installation-directory>/bin/start-jira.bat  
/disableaddons=com.atlassian.jira.jira-software-application
```

Linux

When installed as a service...

If you installed JIRA as a service, use one of the following commands to **start** or **stop** JIRA.

```
$ sudo /etc/init.d/jira start
$ sudo /etc/init.d/jira stop
```

You can't start or stop JIRA manually using the `start-jira.sh` and `stop-jira.sh` files.

Manually when not installed as a service...

If you didn't install JIRA as a service you'll need to start and stop JIRA manually.

- To start JIRA run `<installation-directory>\bin\start-jira.sh`
- To stop JIRA run `<installation-directory>\bin\stop-jira.sh`

We recommend running JIRA with a dedicated user account:

```
$ su -u <user>
$ ./start-jira.sh
```

Where `<user>` is the name of your dedicated user.

If you're using Ubuntu the command is a little different:

```
$ sudo su <user>
$ ./start-jira.sh
```

Manual start with add-ons disabled...

JIRA can be started with all non-system add-ons, or a selection of these add-ons, disabled. This helps with troubleshooting when these non-system add-ons are causing issues with your JIRA instance, such as causing JIRA to fail on start up, or when the add-on is malfunctioning and can't be removed through UPM. The parameters do not persist, that is, they are applied at start up once, and if you need to restart JIRA, you need to apply them again.

The parameters should be specified at system start up when JIRA is started using the `start-jira.sh` script, for example:

```
./bin/start-jira.sh --disable-all-addons
--disable-addons=com.atlassian.test.plugin
```

If you don't use `start-jira.sh` for starting JIRA, but still wish to use this feature, you can add the following JVM parameter to the invocation of the servlet container that's running JIRA:

```
-Datlassian.plugins.startup.options="--disable-all-addons
--disable-addons=com.atlassian.test.plugin"
```

To disable **multiple plugins**, use a colon separated list of plugins. Regex/wildcards are not permitted, the full key of the plugin must be provided, for example:

```
-Datlassian.plugins.startup.options="--disable-all-addons
--disable-addons=com.atlassian.test.plugin:com.atlassian.another.test
.plugin"
```

Notes

- `--disable-addons` takes a colon-separated list (chosen as a colon is the only prohibited character from a plugin key) of addons to be disabled. These **can** be system add-ons.
- This feature does **not** work for JIRA Data Center applications.
- This can be used to disable an add-on deemed critical to JIRA starting, and if one of those is disabled, JIRA will fail to start
- This can be used to disable JIRA application OBR bundles, for example, to stop the JIRA Software add-on:

```
./bin/start-jira.sh
--disable-addons=com.atlassian.jira.jira-software-application
```

Managing LexoRank

LexoRank is ranking system that JIRA Software uses which provides the ability to rank issues in JIRA Software instances. The user interface can be found in



> **System > Lexorank management**. It provides a user interface for several of the key areas of LexoRank administration.

Balancing

Balancing

LexoRank management consists of two parts: the LexoRank service that determines if a rebalance can be run; and the actual rebalance of the Rank field in the database. Note, if you are running clustered JIRA, a balance can only run on one node of the cluster at a time. Also, the service status will only show you activity on this node.

Database status at Tue Jun 20 2017 13:40:25 GMT+1000 (AEST)						Refresh	Balance all fields
Field name	Field id	Num ranked issues	Percent complete	Rank value distribution	Rank status ⓘ		
Rank	10005	18279	100%	18279,0,0	Status: OK Length: 18 / 254 Next rebalance: Scheduled Issue with highest rank length: JSDS-693	Balance field	
Service status at Tue Jun 20 2017 13:40:25 GMT+1000 (AEST)						Refresh	
Balancing disabled							false
Balancing suspended							false
Balance handler running							false
Balancing in progress on a node of the cluster							false

As per the screenshot above, the UI provides the user with details on the distribution of issues in each bucket, and whether or not a balance is in progress or if it's disabled. Running a balance is similar to running a re-index of JIRA, there isn't any guarantee it'll fix a problem unless the problem is specific to the ranking data.

The **Rank Status** column provides details about the status of the ranked issues. Each issue has a unique rank relative to the issues around it. This rank is stored as a string. As the amount of issues grows and you perform more ranking operations, the length of these strings increase. At some point, the length reaches a threshold, and a **rebalance** is scheduled within the next 12 hours.

A rebalance will evenly distribute the ranked issues, and significantly reduce the rank length. During a rebalance, ranking operations can continue as normal.

Should the length reach a second threshold within 12 hours, an **immediate rebalance** is started.

Should the length reach a third threshold before rebalance is finished, all rank operations are **disabled** until the

rebalance completes.

The **Rank Status** field has these properties:

Property	Possible values
Status	<ul style="list-style-type: none"> OK - the rank length is in a healthy state Warning - a rebalance has been scheduled Critical - an immediate rebalance has started, you are approaching a state where rank operations will be disabled
Length	<current length> / <maximum length> Maximum length indicates when rank operations will be disabled
Next rebalance	<ul style="list-style-type: none"> Scheduled - once the next threshold is reached, a rebalance will be scheduled Immediate - once the next threshold is reached, a rebalance will start immediately

In addition to the above, the field will tell you which project has the issues with the longest rank. This can be useful to diagnose the cause of a rapid increase in rank length.

If you're encountering issues and don't know why, or are unsure whether or not to balance, review the integrity checks first and contact support if need be as per the below details.

A breakdown of the service status is below:

Service	Notes
Balancing disabled	If this is <code>true</code> then JIRA Software has disabled balancing internally: <ul style="list-style-type: none"> A foreground re-index may be running. It is expected that this will disable balancing. JIRA Software may have just been installed / upgraded and requires a reindex. Check the logs for any exceptions and see if there are existing KB articles for these errors. See if there is anything failing in the Integrity checks.
Balancing suspended	<ul style="list-style-type: none"> This will be false unless balancing has been explicitly suspended by Support or an administrator.
Balance handler running	<ul style="list-style-type: none"> This will be <code>false</code> unless a balance is currently in progress. To verify the progress of the balance, hit the refresh button. On a JIRA Data Center cluster this will be true on the node that is running the balance, and false on the other nodes.
Balancing in progress on a node of the cluster	<ul style="list-style-type: none"> It indicates that the balance cluster lock has been taken. This will be <code>true</code> when a balance is running.

Integrity checks

This runs a series of tests against the LexoRank data and returns a true / false result based on the test. The tests are detailed below.

Check	How to Fix Failures
Marker rows present in table for rank field.	If this fails, the minimum or maximum marker rows are missing.
Marker rows correctness check.	<p>If this fails, the minimum or maximum marker rows exist, however they have the incorrect rank.</p> <p>This can be fixed by updating the rank on the row returned in the check to be the expected value.</p>
Marker rows in valid bucket check.	<p>When a balance is in progress, the marker rows are moved to another bucket to indicate where the new rank values should be. The only time they should be in different buckets is if a balance is in progress.</p> <p>Valid states for the marker rows are the below.</p> <ul style="list-style-type: none"> • Minimum is the same as maximum. • Minimum is 0, and max is 1. • Minimum is 1, and max is 2. • Minimum is 0, and max is 2. <p>This test fails if the marker rows are not in those buckets, and is likely caused by exceptions thrown during the rank creation or balance operation. Please check the logs for those and verify them against known problems.</p>
Rank out of bounds check.	Please refer to our How to Fix Rank Out Bound Error KB article for the fix.
Duplicate ranks check.	This can be fixed as detailed in How To Fix Duplicate Rank Values For a Rank Field .
Issue ranks different from marker ranks check.	<p>The below SQL will identify records that have ranking values the same as the min/marker rows. Deleting these records will correct this problem (and also result in a loss of ranking data for those issues only).</p> <pre style="border: 1px dashed #ccc; padding: 10px;">SELECT * FROM "AO_60DB71_LEXORANK" WHERE ("RANK" LIKE '% zzzzzz:' OR "RANK" LIKE '% 000000:') AND "TYPE" not in (0,2);</pre> <p> This may require changing depending upon the DBMS used.</p>
Issue rows in valid bucket check.	If a balance can't fix this, it would need detailed analysis from Atlassian Support.
Balance status check.	Attempt a balance, and check the logs to see if there are any exceptions. It's likely the balance is failing due to a exception in the logs, or one of the other checks above may be failing.

If you're unsure how to proceed, please feel free to contact support for further assistance.

Configuring global settings

This section of the documentation contains information on how to check and configure settings in your JIRA installation that are applied globally to all users. It includes information on default settings for your JIRA installation, and default settings that apply to your users.

- [Configuring time tracking](#)
- [Configuring JIRA application options](#)
 - [Configuring advanced settings](#)
 - [Configuring the Base URL](#)
- [Setting properties and options on startup](#)
 - [Recognized system properties for JIRA applications](#)
- [Advanced JIRA application configuration](#)
 - [Changing the constraints on historical time parameters in gadgets](#)
 - [Changing the default order for comments from ascending to descending](#)
 - [Limiting the number of issues returned from a search view such as an RSS feed](#)
- [Configuring file attachments](#)
- [Configuring issue linking](#)
- [Configuring issue cloning](#)
- [Configuring the whitelist](#)
- [Configuring sub-tasks](#)
- [Managing shared filters](#)
- [Managing shared dashboards](#)
- [Enabling logout confirmation](#)
- [Rich text editing](#)

Configuring time tracking

JIRA's time tracking feature enables users to record the time they spend working on issues.

Note:

- Before users can specify time estimates and log work, they must be granted the **Work On Issues** [permission](#) for the relevant project(s).
- For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Disabling time tracking

Time tracking is ON by default (as shown in [screenshot 1](#) below). However, this feature can be disabled from the Time Tracking administration page.

i *Time tracking will be OFF by default if your JIRA installation was upgraded from a version prior to 4.2 that had time tracking either disabled or never enabled.*

1. Choose  **> Issues.**
2. Select **Issue Features > Time Tracking** to open the Time Tracking page.
3. Click the '**Deactivate**' button to turn time tracking OFF.

i You will not lose any existing time tracking data by disabling/re-enabling time tracking.

Enabling time tracking

1. Choose  **> Issues.**
2. Select **Issue Features > Time Tracking** to open the Time Tracking page.
3. Click the '**Activate**' button to turn time tracking ON.

On this page:

- [Disabling time tracking](#)
- [Enabling time tracking](#)
- [Configuring time tracking settings](#)
- [About 'Legacy Mode'](#)
- [Related topics](#)

Time tracking add-ons for JIRA in the Atlassian Marketplace extend JIRA's time tracking power. [Check them out here.](#)

Screenshot 1: Time tracking is ON

Time Tracking is currently ON. ?

Note: To change these values deactivate and then reactivate Time Tracking.

The number of working hours per day is **8**.
 The number of working days per week is **5**.
 Time estimates will be displayed in the following format: **pretty (e.g. 4 days, 4 hours, 30 minutes)**
 The current default unit for time tracking is **minute**.
 Copying of comments to work description is currently **enabled**.

For the users you wish to be able to log work on issues, ensure that they have the **Work On Issues** permission in the relevant [permission scheme](#).

To deactivate Time Tracking, simply click below.

Deactivate

Configuring time tracking settings

To edit JIRA's time tracking settings, it must first be disabled. Once you have changed the settings, you will then need to re-enable time tracking so that users can log work on issues.

i You will not lose any existing time tracking data by disabling/re-enabling time tracking.

1. Choose



> **Issues.**

2. Select **Issue Features > Time Tracking** to open the Time Tracking page.
3. If Time Tracking is ON (refer to the indication at the top of the Time Tracking screen), click the **Deactivate** button to turn time tracking OFF.
4. The time tracking settings will now be editable as shown in the following screenshot.

Screenshot 2: Time tracking is OFF

Time Tracking is currently OFF.

Activate Time Tracking below.

Hours per day
Please specify the number of hours per working day. The default for this value is 8 hours.

Days per week
Please specify the number of working days per week. The default for this value is 5 days.

Time format
 pretty (e.g. 1 week, 1 day, 30 minutes)
 days (e.g. 6d 0.5h)
 hours (e.g. 36.5h)

Default Unit
Time unit used for input that doesn't explicitly specify one. The default for this value is "minute".

Legacy Mode
In legacy mode, the original estimate and remaining estimate are linked and only one value can be updated at a time. This is no longer the default for new installations of JIRA version 4.2 and later.

Copy Comment To Work Description
When this option is enabled, any comment entered as part of a workflow transition on an issue will be copied to the work log description if work is logged as part of that transition.

Activate

5. Configure time tracking settings by editing the following fields:
 - 'Hours per day' — enter a suitable value (e.g. 8). You can enter fractions if you wish.
 - 'Days per week' — enter a suitable value (e.g. 5). You can enter fractions if you wish.
 - 'Time format' — select **pretty/days/hours**. This will determine the format of the 'Time Spent' field when an issue is displayed.
 - 'Default Unit' — select **minutes/hours/days/weeks**. This will be applied whenever your users log work on an issue without specifying a unit.
 - 'Legacy Mode' — select this checkbox if you prefer to use JIRA's time tracking features as they operated prior to JIRA version 4.2. For more details about this option, please see [About 'Legacy Mode'](#) (below).
 - 'Copy Comment To Work Description' — select this checkbox to ensure that any content entered into a Comment field while logging work as part of an issue operation, is also copied across to the Work Description.

i When 'Copy Comment To Work Description' is enabled, your user's work log entries will be visible only to members of the project role or group selected in the padlock icon drop-down on their issue operation screen. If 'Copy Comment To Work Description' is disabled, your user's work log entries will be visible to anyone by default.

6. Click the '**Activate**' button to turn time tracking ON.
If the permission schemes used by your project(s) already have the appropriate Work On Issues permissions, then there is no need to proceed any further.
However, if you need to configure these permissions, proceed with the remaining steps below:
7. Click the '**permission scheme**' link as shown in [screenshot 1 \(above\)](#). The 'Permissions Scheme' page will be displayed.
8. Click the '**Permissions**' link of the permission scheme associated with the project(s) where you wish to specify Work On Issues permissions. The 'Edit Permissions' page is displayed for your chosen permission scheme.

i See [Managing project permissions](#) for details about the various permissions.

9. Check whether the row labeled 'Work On Issues' contains the appropriate users, groups or project roles who need to specify time estimates or log work. If it does not, click the '**Add**' link in the 'Operations' column:

Screenshot 3: Time tracking Permissions

Time Tracking Permissions	Users / Groups / Project Roles	Operations
Work On Issues <small>Ability to log work done against an issue. Only useful if Time Tracking is turned on.</small>	<ul style="list-style-type: none"> • Project Role (Developers) (Delete) 	<input type="button" value="Add"/>

10. Select the users, groups or project roles to whom you want to allow time tracking and work logging on issues.
11. Click the '**Add**' button.
12. If it is needed to enter the '**Original Estimate**' during issue creation or during issue editing, ensure that the field '**Time Tracking**' is added to the relevant screens associated with those operations. Refer [Associating a screen with an issue operation](#) for more details.

About 'Legacy Mode'

- If Legacy Mode is disabled, your users will be able to change the **Original Estimate** value *irrespective of any work being logged on an issue*. Legacy Mode is disabled by default on new installations of JIRA version 4.2 or later.
- If Legacy Mode is enabled, your users can only specify an **Original Estimate** *before* they start logging work on an issue. This value cannot be changed once *any* work has been logged, unless all work logs for that issue are first deleted.
- By default,
 - **Legacy Mode** is disabled if your JIRA 4.2 installation was conducted cleanly (that is, without upgrading from an earlier version of JIRA).
 - **Legacy Mode** is enabled if you upgraded JIRA from a version prior to 4.2.
- With Legacy Mode enabled, if you change the Remaining Estimate field in a workflow post function the Original Estimate is also cleared. This issue is tracked at

JRASERVER-25031 - Time Tracking Legacy Mode and Workflow Post Functions Error
GATHERING IMPACT

Related topics

- [Defining a screen](#)

Configuring JIRA application options

JIRA has a number of configuration options that allow your JIRA applications to be customized for use within your organization. These options can be accessed and edited on JIRA's 'General Configuration' page.

On this page:

- Editing JIRA's general configuration
- General settings
- Internationalization
- Options

Editing JIRA's general configuration

1. Choose



> **System.**

2. Select **General Configuration** to open the Administration page.
3. Click the **Edit Settings** button to edit the three sections as described below:
 - Settings
 - Internationalization
 - Options

i The Advanced Settings button is only visible if you have the **JIRA System Administrators** global permission.

General settings

(If marked with an * the function is not available or editable in the Cloud)

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. Also see logo, which is displayed on every JIRA page.
Mode *	JIRA can operate in two modes: <ul style="list-style-type: none"> - Public — Anyone can sign themselves up with self-registration and create issues (within the bounds of your JIRA system's permissions). - Private — Useful for internal issue-tracking systems where you do not want public users to login. Self-signup is disabled; only Administrators can create new users. <p>If the JIRA application has a LDAP directory configured with delegated authentication and the option Copy user is enabled, users will be able to login and create a new accounts.</p> <p><i>Default: Private</i></p>
Maximum Authentication Attempts Allowed *	The maximum authentication attempts that are allowed before CAPTCHA is shown to a user. If you leave it blank then CAPTCHA will never be shown and users will have unlimited authentication attempts. It is recommended that you set this to a small number (e.g. below 5). <i>Default: 3 (for new installations of JIRA)</i>
CAPTCHA on signup *	If you are running JIRA in Public mode (see above), it is strongly recommended that you enable CAPTCHA. This will show a CAPTCHA image on signup to prevent spambots from signing up. <i>Default: OFF</i>

Base URL*	The base URL of this JIRA installation. You can only configure JIRA to respond to a single URL and this setting <i>must</i> match the URL that your users request for accessing your JIRA instance. You cannot (for example) have a different hostname or URL for internal and external users. This URL is also used in outgoing email notifications as the prefix for links to JIRA issues. Check out Configuring the Base URL for more information.
Email from	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}'. Note that the actual address (e.g. 'jira@company.com') cannot be specified here. <div style="border: 1px solid #ccc; padding: 5px; text-align: center;">The address is determined by the mail server or individual project configuration.</div>
Introduction	A short introduction message displayed on the dashboard . Also see the announcement banner , which is displayed on every JIRA page. You can include HTML, but ensure all tags are correctly closed.

Internationalization

Setting	Description
Indexing language	JIRA uses Lucene , a high-performance text search engine library, in full-text searches for issues stored in JIRA. This option is designed to enhance JIRA's search indexing and issue searching features for issues entered in the languages available in this list. Hence, choose the language that matches the language used in your issues. Choosing a specific language in this list has the following effects when conducting searches in JIRA (with respect to your chosen language): <ul style="list-style-type: none"> • Reserved words in text fields will not be indexed. • Stemming of words in all JIRA fields will be active. If multiple languages are used in your issues (or you wish to disable the two effects above), choose Other . You will need to re-index JIRA if you change this value.
Installed languages	This section lists all language packs available within the JIRA system. (Note: to install additional languages, see Internationalization .)
Default language	The language used throughout the JIRA interface (as selected from the list displayed in Installed Languages above). Users can override the default language by using the Language setting in their user profile.
Default user time zone	This is the time zone used throughout the JIRA interface. Users can override the default time zone by using the Time Zone preference in their user profile. (To choose the time <i>format</i> , see Configuring the look and feel of your JIRA applications .) Date fields that have no time component, such as due dates, release dates (associated with versions), and custom date fields, solely record date information (and no time zone-related information). These are not affected by time zone settings.

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 . <i>Default: ON</i>
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 . <i>Default: ON</i>
Allow users to share dashboards and filters with the public	Controls whether "Public" is an option for sharing a filter. Users can choose who to share a filter with, and "Public" allows them to share it with anonymous users i.e. user <i>n</i> of logged into JIRA. If your instance can be accessed publicly, these filters will be accessible by the general public. Changing the option to OFF will not change any existing filters that are shared as "Public". <i>Default: ON</i>
Maximum project name size	Controls the maximum number of characters allowed for a project name. Changing this value will not affect the names of existing projects. <i>Default: 80</i>
Maximum project key size	Controls the maximum number of characters allowed for a project key. Changing this value will not affect the keys of existing projects. You can set this to any value between 2 and 255, inclusive. <i>Default: 10</i>
Allow unassigned issues	When turned ON , the default assignee for the project is Unassigned . When turned OFF , issues must always be assigned to someone - by default, the assignee will be the Project Lead as defined for each project . <i>Default: ON</i>
External user management	When turned ON , JIRA will not display options for users to change their password and edit their profile. This will also disable the Forgot your password link on the login page. Generally you would only turn this ON if you are managing all your users from outside JIRA (e.g. using Crowd , Microsoft Active Directory , or another LDAP directory) <i>Default: OFF</i>
Logout confirmation	Controls whether to obtain user's confirmation when logging out: NEVER COOKIE - prompt for confirmation if the user was automatically logged in (via a cookie). ALWAYS <i>Default: NEVER</i>
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. <i>Default: ON</i>
User email visibility	Controls how users' email addresses are displayed in the user profile page. <ul style="list-style-type: none"> - 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. <i>Default: PUBLIC</i>
Comment visibility	Determines what will be contained in the list that is presented to users when specifying comment visibility and worklog visibility. <ul style="list-style-type: none"> - Groups & Project Roles - the list will contain groups and project roles. - Project Roles only - the list will only contain project roles. <i>Default: Project Roles only</i>

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). <i>Default: OFF</i>
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. <i>Default: ON</i>
JQL Auto-complete	Provides auto-completion of search terms when users perform an advanced (JQL) search. Turn OFF if you prefer not to use this feature, or are experiencing a performance impact. <i>Default: ON</i>
Internet Explorer MIME Sniffing Security Hole Workaround Policy	Attachment viewing security options for cross-site site scripting vulnerabilities present in Internet Explorer 7 and earlier. Changes the default browser action for attachments in JIRA. Options are: - Insecure: inline display of attachments - allows all attachments to be displayed inline. Only select this option if you fully understand the security risks. - Secure: forced download of all attachments for all browsers - force the download of all attachments. This is the most secure option, but is less convenient for users. - Work around Internet Explorer security hole - forced download of high-risk attachments (IE-only Workaround) - for IE browsers, force the download of attachments that IE would mistakenly detect as an HTML file. Declared HTML attachments are also never displayed inline. Use this option to reduce the risk of attacks to IE users via attachments. <i>Default: Work around Internet Explorer security hole</i>
Contact Administrators Form	Provides an email form for users to fill in when they click the 'Contact Administrators' link (which appears when appropriate in JIRA, e.g. on Login panels and pages). <div style="border: 1px solid black; padding: 5px; text-align: center;">Applies only if outgoing email is enabled.</div> Can be used with or without the custom 'Contact Administrators Message' below. Users with the JIRA Administrators global permission (not JIRA System Administrators - see JIRA-27454 for details) will be notified as a result of this feature being used. <i>Default: OFF</i>
Contact Administrators Message	Displays a custom message when users click the 'Contact Administrators' link (which appears when appropriate in JIRA, e.g. on Login panels and pages). The 'Contact Administrators Message' will be displayed at the top of the 'Contact Administrators Form', only if the form is enabled (see above).
Allow Gravatars	Enables users to use Gravatars in their user profile instead of JIRA-specific avatars. Users will not be able to use JIRA-specific avatars if Gravatars are enabled, and vice versa. <i>Default: ON</i>
Inline edit	Enables inline editing, i.e. click to edit a field on the screen. <i>Default: ON</i>
Auto-update search results	Enables search results to be automatically updated when criteria are modified in a basic search. <i>Default: ON</i>

Configuring advanced settings

JIRA has a small number of commonly edited advanced configuration options, which are stored in the JIRA database. These options can be accessed and edited from the **Advanced Settings** page. You must be a **JIRA**

System Administrator to do this.

Editing JIRA's advanced settings

To access and edit options on the 'Advanced Settings' page:

1. Choose



> **System.**

2. Click the **Advanced Settings** button on the 'General Configuration' page.
3. Edit the value of a **Key** by clicking its value on the right of the page and modifying the existing value. The table below has extended information on some of the **Key** values.

Key	Key configuration
jira.attachments.number.of.zip.entries	Configuring the number of files shown in the content of ZIP-format files on issues
jira.clone.prefix	Configuring the cloned issue summary field prefix
jira.date.picker.java.format jira.date.picker.javascript.format jira.date.time.picker.java.format jira.date.time.picker.javascript.format	Configuring date picker formats
jira.issue.actions.order	Changing the default order for comments from ascending to descending
jira.projectkey.pattern	Changing the Project Key Format
jira.search.views.default.max	The default maximum number of issues exported from search results. Users can override the default by changing the URL <code>tempMax</code> parameter. This value must always be lower than or equal to the <code>jira.serach.views.max.limit</code> value.
jira.search.views.max.limit	The absolute limit on the number of issues that can be exported from search results.
jira.table.cols.subtasks	Configuring sub-task fields displayed on parent issues
jira.view.issue.links.sort.order	Configuring the order of linked issues displayed on the 'view issue' page
jira.text.field.character.limit	This property limits the number of characters that can be entered into Description , Environments , Comments and text custom fields . The maximum is 2147483647. A value of 0 means unlimited characters.
jira.comment.collapsing.minimum.hidden	The minimum number of comments needed before the comment collapsing is enabled.
jira.newsletter.tip.delay.days	The number of days before a prompt to sign up to the JIRA applications insiders newsletter is shown. A value of -1 disables this functionality.
jira.bulk.create.max.issues.per.import	This property allows you to set the maximum number of issues a user can import via CSV at one time. The maximum is 2147483647. Entering a value of 0 will disable the importer for users.

jira.export.html.enabled	Specifies whether users can export the JQL search results to HTML.
jira.quicksearch.max.concurrent.searches	<p>The maximum number of concurrent searches that your users can perform by using the quick search. The limit applies to a single JIRA instance (if you have Data Center with 5 nodes, the limit increases fivefold.)</p> <p>Many concurrent searches will affect JIRA's performance. You can use JMX monitoring to see how your users are searching, and determine the best limit.</p>

- Click the **Update** button (which will appear in the **Operations** column on the right) to save the new value in the JIRA database.

i Please Note:

- Any changes you make to these properties/keys become effective immediately.
- Click the **General Settings** button to return to the **General Configuration** page.

Related information

There are a handful of other advanced configuration options (which are of little interest to most JIRA system administrators) whose default values can be customized in the `jira-config.properties` file located in the JIRA application home directory, which you may want to edit. For details, please see [Advanced JIRA configuration](#).

Configuring the Base URL

The **Base URL** is the URL via which users access JIRA applications. The base URL **must** be set to the same URL by which browsers will be viewing your JIRA instance.

JIRA will automatically detect the base URL during setup, but you may need to set it manually if your site's URL changes or if you set up JIRA from a different URL to the one that will be used to access it publicly.

You need to have the **system administrator global permission** in order to perform this function.

To configure the Base URL:

- Choose  **> System.**
- Choose **General Configuration** in the left-hand panel.
- Choose **Edit Settings.**
- Enter the new URL in the **Base URL** text box.
- Choose **Save.**

Example

If JIRA is installed to run in a non-root context path (that is, it has a context path), then the server base URL should include this context path. For example, if JIRA is running at:

```
http://www.foobar.com/JIRA
```

then the server base URL should be:

```
http://www.foobar.com/JIRA
```

Notes

- **Using different URLs.** If you configure a different base URL or if visitors use some other URL to access JIRA, it is possible that you may encounter errors while viewing some pages.
- **Changing the context path.** If you change the context path of your base URL, you may also need to edit the web server's `server.xml` file to reflect the new path:
 1. Stop the JIRA server.
 2. Go to your JIRA 'destination directory'. This is the directory where the Confluence installation files are stored. For example, `C:\Program Files\Atlassian\JIRA`. Let's call this directory '{JIRA_INSTALLATION}'.
 3. Edit the configuration file at `{JIRA_INSTALLATION}\conf\server.xml`.
 4. Change the value of the `path` attribute in the `Context` element to reflect the context path. For example, if JIRA is running at `http://www.foobar.com/JIRA`, then your `path` attribute should look like this:

```
<context path="/JIRA" docBase="../JIRA" debug="0"
reloadable="false" useHttpOnly="true">
```

5. Save the file.
- **Proxies.** If you are running behind a proxy, ensure that the proxy name matches the base URL. For example: `proxyName="foobar.com" proxyPort="443" scheme="https"`. This will make sure we are passing the information correctly.
 - This information needs to be added in the `Connector` element at `{JIRA_INSTALLATION}\conf\server.xml`.

Setting properties and options on startup

This page describes how to set Java properties and options on startup for JIRA.

On this page:

- [Linux](#)
- [Windows \(starting from .bat file\)](#)
- [Windows service](#)
- [Verifying your settings](#)
- [List of startup parameters](#)

Linux**To Configure System Properties in Linux Installations,**

1. From `<jira-install>/bin`, open `setenv.sh`.
2. Find the section `JVM_SUPPORT_RECOMMENDED_ARGS=`
3. Refer to the list of parameters [below](#).

i Add all parameters in a space-separated list, inside the quotations.

Windows (starting from .bat file)

To Configure System Properties in Windows Installations When Starting from the .bat File,

1. From <jira-install>/bin, open **setenv.bat**.
2. Find the section **set JVM_SUPPORT_RECOMMENDED_ARGS=**
3. Refer to the list of parameters [below](#).

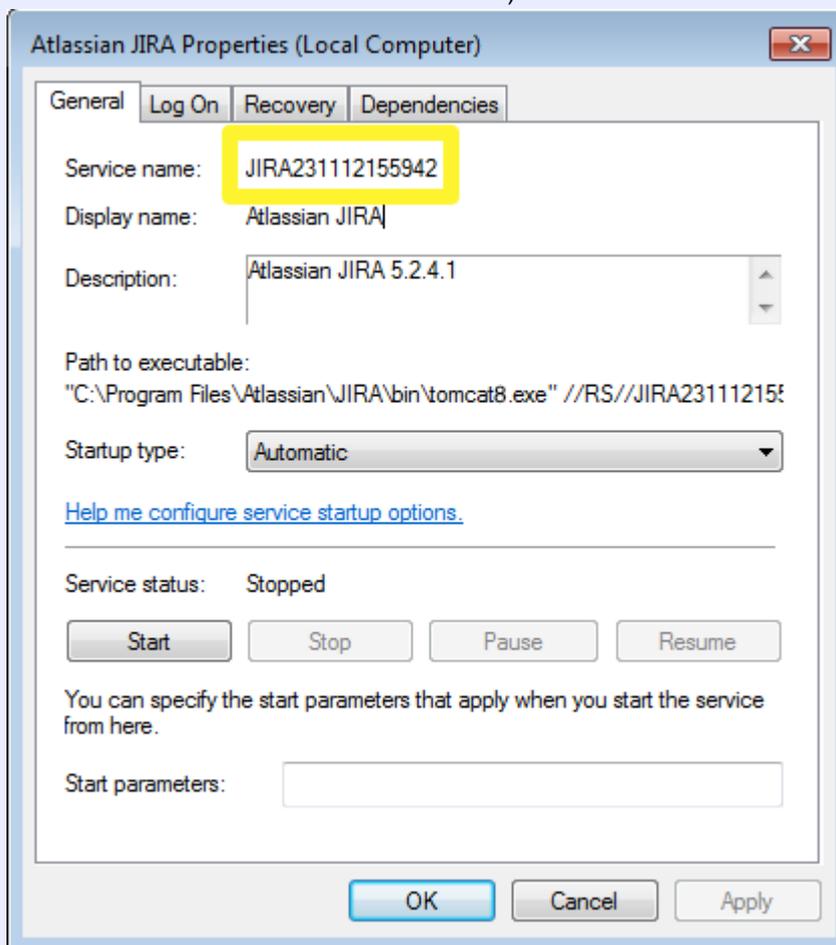
i Add all parameters in a space-separated list, inside the quotations.

Windows service

There are two ways to configure system properties when starting Running JIRA as a Windows service, either via [command line](#) or in the [Windows registry](#).

Setting properties for Windows services via command line**Setting Properties for Windows Services via Command Line**

1. Identify the name of the service that JIRA is installed as in Windows (Control Panel > Administrative Tools > Services):

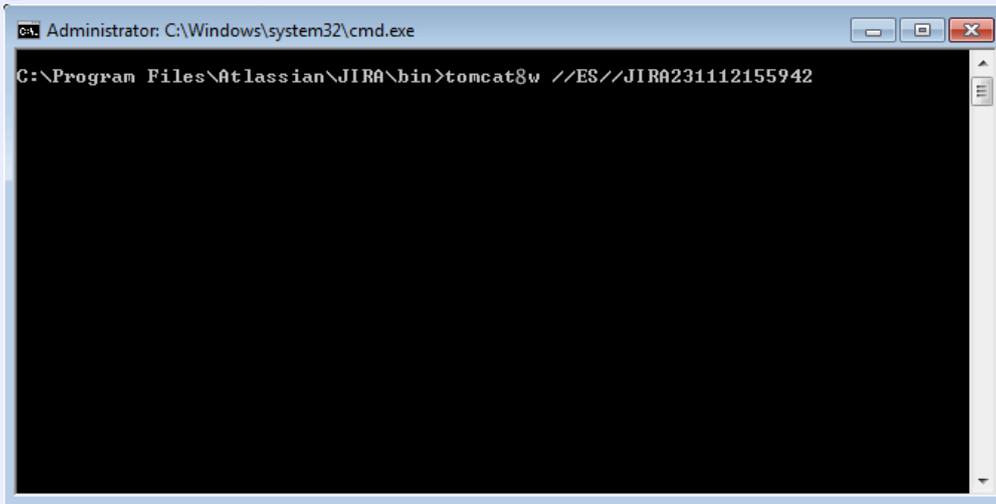


- i** In the above example, the **SERVICENAME** is: JIRA231112155942
2. Open the command window from Start >> Run >> type in 'cmd' >> Enter
 3. cd to the bin directory of your JIRA application installation directory.

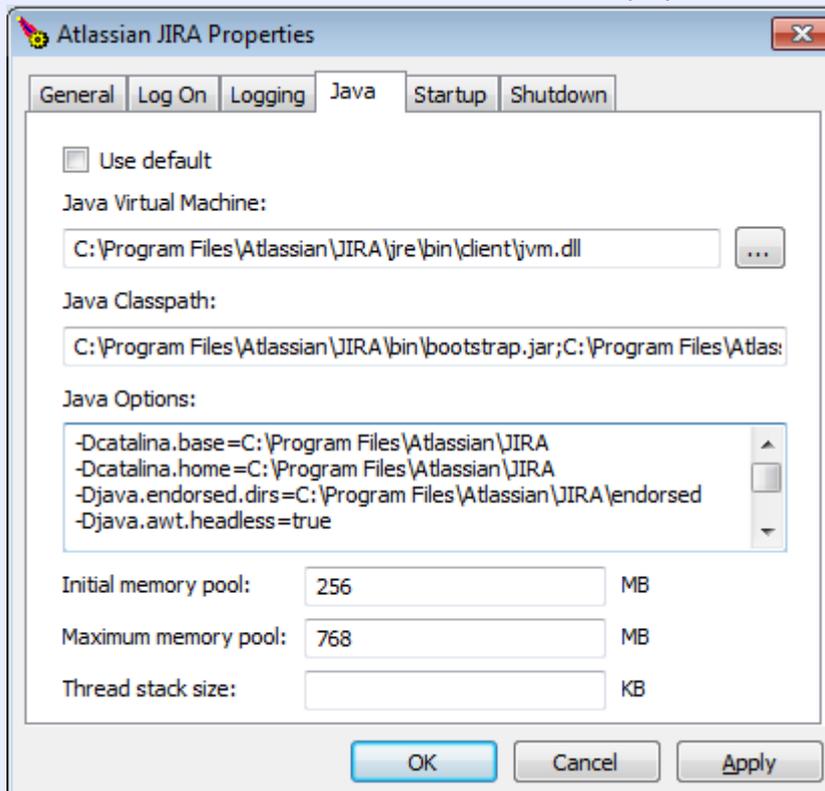
4. Run:

```
tomcat8w //ES//%SERVICENAME%
```

i In the above example, it would be `tomcat8w //ES//JIRA231112155942`



5. Click on the Java tab to see the list of current start-up options:



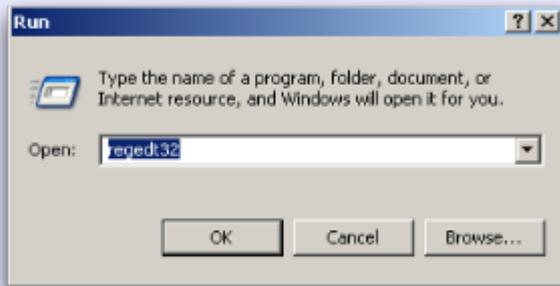
6. Append any new option on its own new line by adding to the end of the existing Java options. Refer to the list of parameters below.

Setting properties for Windows services via the Windows registry

In some versions of Windows, there is no option to add Java variables to the service. In these cases, you must add the properties by viewing the option list in the registry.

To Set Properties for Windows Services via the Windows Registry,

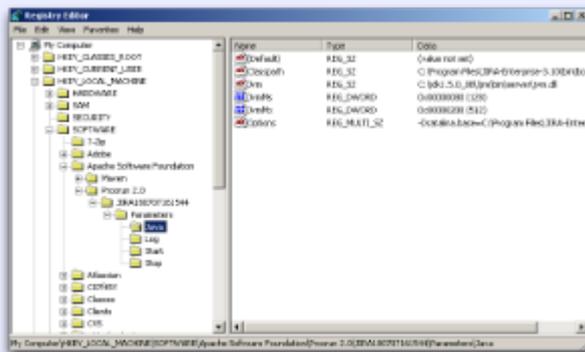
1. Go to Start >> Run, and run "regedit32.exe".



2. Find the Services entry:

32-bit: HKEY_LOCAL_MACHINE >> SOFTWARE >> Apache Software Foundation >> Procrun 2.0 >> JIRA

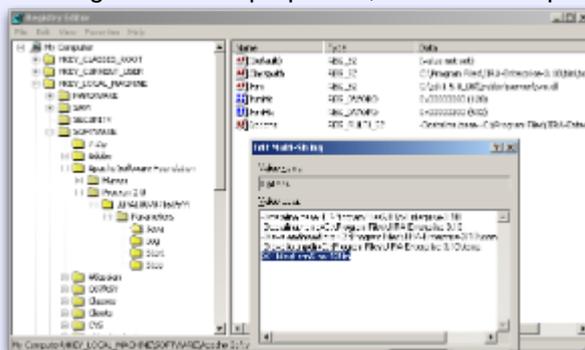
64-bit: HKEY_LOCAL_MACHINE >> SOFTWARE >> Wow6432Node >> Apache Software Foundation >> Procrun 2.0 >> JIRA



3. To change existing properties, especially increasing Xmx memory, double-click the appropriate value.



4. To change additional properties, double-click options.



5. Refer to the list of parameters below. Enter each on a separate line.

Verifying your settings

To verify what settings are in place, check the <jira-home>/logs/atlassian-jira.log or catalina

.out file. A section in the startup appears like this:

```
JVM Input Arguments :
-Djava.util.logging.config.file=/usr/local/jira/conf/logging.properties -XX:MaxPermSize=256m -Xms256m -Xmx384m -Djava.awt.headless=true
-Datlassian.standalone=JIRA
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER=true
-Dmail.mime.decodeparameters=true
-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djava.endorsed.dirs=/usr/local/jira/endorsed
-Dcatalina.base=/usr/local/jira -Dcatalina.home=/usr/local/jira
-Djava.io.tmpdir=/usr/local/jira/temp
```

This display is also available by [viewing your system information](#).

List of startup parameters

Memory property	Notes	Rela
-Xmx -Xms XX:MaxPermSize	These properties are pre-existing. See related pages for instructions.	Incre
-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+PrintGCCause -Xloggc:C:\Program Files\Atlassian\Application Data\JIRA\log\atlassian-jira-gc-%t.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=20M	These properties are pre-existing, and are used for Garbage Collection tuning.	Usin to A Perf Usin Ana
-agentlib:yjpagent=onexit=memory,dir=/path/to/write/snapshots		Prof Usa
Mail property	Notes	Rela
-Datlassian.mail.senddisabled -Datlassian.mail.fetchdisabled -Datlassian.mail.popdisabled	Set to 'true' to disable mail. In Linux setenv.sh, there is a pre-existing flag to uncomment.	Migr serv Noti Inco
-Dmail.debug	If set to "true", logs statements related to mail	Con serv Crea from
-Dmail.mime.decodeText.strict		Una or B Fror
-Dmail.imap.auth.plain.disable -Dmail.imaps.auth.plain.disable		Auth conr

-Dmail.imap.starttls.enable		'java No l due
-Dmail.mime.decodeparameters	Sets mail handler to work correctly with emails from RFC 2231-compliant mail clients.	
-Dmail.smtp.localhost		Prot JIRA/ addr
Encoding property	Notes	Rela
-Dfile.encoding -Dsun.jnu.encoding	Set to utf-8 for encoding consistency	Cha ASC Que Inter Enc SQL issu appl enc Inter Noti Are Que
Other Properties	Notes	Rela
-Duser.timezone		Inco JIRA/
-Dsvnkit.http.methods	Values include Basic,Digest,Negotiate,NTLM	JIRA/ 'java Una conf Sub unkr actic Auth
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER	true	Out/ Men JRA
-ea/-da	Enable/Disable assertions	java Sen
-Djava.net.preferIPv4Stack		Soc 'Inve Avai
-Djavax.net.ssl.trustStore		Una 'java Due
-Djava.awt.headless	Ships with true by default. Allows thumbnail generation.	

-Dhttp.proxyHost -Dhttp.proxyPort -Dhttps.proxyHost -Dhttps.proxyPort	Outbound Proxy Server hostname and port	How HTTP/JIRA
-Dorg.apache.catalina.SESSION_COOKIE_NAME		Log application Console
-Datlassian.plugins.enable.wait	Time JIRA waits for plugins to load.	JIRA/Whitelabel/Enable
-Datlassian.plugins.startup.options="--disable-all-addons --disable-addons=com.atlassian.test.plugin"	Allows JIRA to start with all user installed or specific user installed add-ons disabled. For more information on manual start up and specifying add-ons, see Start and Stop JIRA applications .	Startup application
-Dhide.system.error.details	Hides the details of errors that are displayed after starting Jira. The page (johnson) where these are displayed will still notify you about the errors.	
-Djira.startup.warnings.disable	Disables the page (johnson) that displays errors after starting Jira if there are only dismissible warnings. The page will appear if there are any important errors.	

Recognized system properties for JIRA applications

JIRA supports some configuration and debugging settings that can be enabled through Java system properties. System properties are usually set by passing the `-D` flag to the Java virtual machine in which JIRA is running. See [Setting properties and options on startup](#).

List of startup parameters

Memory property	Notes	Related pages
-Xmx -Xms XX:MaxPermSize	These properties are pre-existing. See related pages for instructions.	Increasing JIRA memory

-XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintGCTimeStamps -XX:+PrintGCCause -Xloggc:C:\Program Files\Atlassian\Application Data\JIRA\log\atlassian-jira-gc-%t.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=5 -XX:GCLogFileSize=20M	These properties are pre-existing, and are used for Garbage Collection tuning.	Using Garbage Collection Logs to Analyze JIRA Application Performance Using Memory Dumps to Analyze OutOfMemoryErrors
-agentlib:yjpagent=onexit=memory,dir=/path/to/write/snapshots		Profiling Memory and CPU Usage with YourKit
Mail property	Notes	Related pages
-Datlassian.mail.senddisabled -Datlassian.mail.fetchdisabled -Datlassian.mail.popdisabled	Set to 'true' to disable mail. In Linux setenv.sh, there is a pre-existing flag to uncomment.	Migrating JIRA to another server Notifications Are Issued for Incorrect Issues
-Dmail.debug	If set to "true", logs statements related to mail	Configuring an SMTP mail server to send notifications Creating issues and comments from email
-Dmail.mime.decodeText.strict		Unable to Decode Mail Subject or Body when Creating Issue From Email
-Dmail.imap.auth.plain.disable -Dmail.imaps.auth.plain.disable		Authenticate failed error when connecting to Exchange
-Dmail.imap.starttls.enable		'javax.mail.MessagingException No login methods supported' due to IMAP over SSL
-Dmail.mime.decodeParameters	Sets mail handler to work correctly with emails from RFC 2231-compliant mail clients.	
-Dmail.smtp.localhost		Problems Sending Email from JIRA - EHLO requires domain address
Encoding property	Notes	Related pages

-Dfile.encoding -Dsun.jnu.encoding	Set to utf-8 for encoding consistency	Characters Not Supported by ASCII are Being Displayed as Question Marks Internationalization and Encoding Troubleshooting SQL Exception while updating issues or importing data in JIRA applications with MySQL due to encoding International Characters in Notification Email Subject Lines Are Being Replaced with Question Mark
Other Properties	Notes	Related pages
-Duser.timezone		Incorrect Times Displayed in JIRA
-Dsvnkit.http.methods	Values include Basic,Digest,Negotiate,N TLM	JIRA Startup Fails Due to 'java.lang.SecurityException Unable to locate a login configuration' Subversion Plugin Displays 'An unknown error occurred - actions == null' Due to SVN Authentication
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER	true	OutOfMemory Due to Tomcat Memory Leak JRA-10145
-ea/-da	Enable/Disable assertions	java.lang.AssertionError When Sending Mail Via SMTP
-Djava.net.preferIPv4Stack		SocketException to Announce 'Invalid argument' for an Available Port
-Djavax.net.ssl.trustStore		Unable to Send Email 'javax.net.ssl.SSLEnabledException' Due to SMTP Server via SSL
-Djava.awt.headless	Ships with true by default. Allows thumbnail generation.	
-Dhttp.proxyHost -Dhttp.proxyPort -Dhttps.proxyHost -Dhttps.proxyPort	Outbound Proxy Server hostname and port	How to Configure an Outbound HTTP and HTTPS Proxy for JIRA applications
-Dorg.apache.catalina.SESSION_COOKIE_NAME		Logging into another Atlassian application logs me out of Confluence
-Datlassian.plugins.enable.wait	Time JIRA waits for plugins to load.	JIRA System Plugin Timeout While Waiting for Plugins to Enable

-Datlassian.plugins.startup.options="--disable-all-addons --disable-addons=com.atlassian.test.plugin"	Allows JIRA to start with all user installed or specific user installed add-ons disabled. For more information on manual start up and specifying add-ons, see Start and Stop JIRA applications .	Start and Stop JIRA applications
-Dhide.system.error.details	Hides the details of errors that are displayed after starting Jira. The page (johnson) where these are displayed will still notify you about the errors.	
-Djira.startup.warnings.disable	Disables the page (johnson) that displays errors after starting Jira if there are only dismissible warnings. The page will appear if there are any important errors.	

Advanced JIRA application configuration

JIRA has a number of advanced configuration options, each of which is defined as an individual property (or 'key' associated with a value). These key-value pairs are stored in one of three areas for use by JIRA:

- [The JIRA database](#)
- [The jira-config.properties file](#)
- [The jpm.xml file](#)

The JIRA database

The values of a small number of most commonly edited advanced configuration options are stored in the JIRA database. These values can be edited from the **Advanced Settings** page of JIRA's administration area. To access the values for editing, see [Configuring JIRA options](#).

Once any of these properties' values are changed, they become effective immediately.

The jira-config.properties file

Custom values for JIRA's remaining advanced configuration options (i.e. not stored in the [JIRA database](#)) are stored as individual key-value pairs in a file called `jira-config.properties` (located in the [JIRA application home directory](#)). Typically, these options are of little interest to most JIRA system administrators. While these key-value pairs can be edited, JIRA must be restarted for any changed values to take effect.

Example contents to demonstrate format

```
jira.projectkey.warning = testwarning
jira.projectkey.description = testdescription
```

i In new JIRA installations, this file may not initially exist and if so, needs to be created manually. For more information about editing the `jira-config.properties` file, see [How to edit the jira-config.properties file](#).

The jpm.xml file

Default values for all* of JIRA's available advanced configuration options are stored in a file called `jpm.xml` (located in the `<jira-application-dir>/WEB-INF/classes` subdirectory of the [JIRA application installation directory](#)). These default values are only used by JIRA if a property's value has not already been customized in either the [JIRA database](#) (via JIRA's 'Advanced Settings' page) or the `jira-config.properties` file.

! The `jpm.xml` file should not be edited because any values that you customize in it will not be migrated

automatically during subsequent JIRA upgrades. To change the value of a property for an advanced configuration option in JIRA, override the value of this property by redefining it in either:

- The JIRA database (via JIRA's 'Advanced Settings' page).
- OR**
- The `jira-config.properties` file.

* JIRA recognizes a small number of properties, which can be set in your `jira-config.properties` file *but have no definition in the `jpm.xml` file*. These properties:

- typically represent advanced configuration options that are disabled when they are not defined in your `jira-config.properties` file and
- when not specified in your `jira-config.properties` file, typically affect JIRA's behavior differently to when they are specified in your `jira-config.properties` file *with no value*.

Making changes to the `jira-config.properties` file

1. Shut down JIRA (for example, by executing either the `/bin/stop-jira.sh` or `\bin\stop-jira.bat` file in your [JIRA application installation directory](#), or by stopping the JIRA service).
2. Open the `jira-config.properties` file (located at the root of your [JIRA application home directory](#)) in a text editor.
 - ⚠ This file may not exist if you are using a new JIRA installation or an upgraded JIRA installation where your previous JIRA version(s) had never been customized. If this file does not exist, create it using a text editor.
3. Edit the appropriate properties in this file.
 - ✔ **Editing tips:**
 - To determine the default value of a property whose value you wish to redefine, search for that property in the `<jira-application-dir>/WEB-INF/classes/jpm.xml` file (of your JIRA Installation Directory). The default value is defined in the `<default-value/>` sibling element of the relevant property's `<key/>` element.
 - To override a property's default value in `jpm.xml` (which is not already defined in your `jira-config.properties` file or available on the ['Advanced Settings' page](#)):
 - a. Copy the value of the relevant property's `<key/>` element from the `jpm.xml` file to the `jira-config.properties` file.
 - b. In the `jira-config.properties` file, add an '=' after that property's key, followed by your custom value.
 - To disable a custom property's value in the `jira-config.properties` file, either 'comment out' the property with a preceding '#' symbol or remove the property from the file.
4. Save your modifications to the `jira-config.properties` file.
5. Restart JIRA.

See also

[Setting properties and options on startup](#) — for changes like setting available memory, disabling email, etc.

Changing the constraints on historical time parameters in gadgets

A number of JIRA gadgets show historical data from your JIRA server. You can generally configure the time constraints on this data via gadget parameters, such as those parameters defining how far back should data be retrieved. For performance reasons, however, the JIRA server can impose an overriding maximum limit on historical data retrieved by gadgets. These maximum limits imposed by the JIRA server are defined by the following [advanced configuration options](#) in JIRA and can be customized in your `jira-config.properties` file (located in the [JIRA application home directory](#)).

```
jira.chart.days.previous.limit.yearly=36500
jira.chart.days.previous.limit.quarterly=22500
jira.chart.days.previous.limit.monthly=7500
jira.chart.days.previous.limit.weekly=1750
jira.chart.days.previous.limit.daily=300
jira.chart.days.previous.limit.hourly=10
```

To update these properties:

1. Shut down your JIRA server.
2. Edit your `jira-config.properties` file in your JIRA home directory.
 - i** See [Making changes to the `jira-config.properties` file](#) for more information.
3. Locate these properties.
 - i** If any of these properties do not exist in your `jira-config.properties` file, add them to the file.
4. Update the values of these properties as desired.
5. Save your changes to the `jira-config.properties` file.
6. Restart your JIRA server.

Changing the default order for comments from ascending to descending

1. Access JIRA's 'Advanced Settings' page. (See [Configuring advanced settings](#) for more information.)
2. Edit the value of the `jira.issue.actions.order` property by clicking the existing value and changing it from `asc` to `desc`
3. Click the **Update** button to save the new value in the JIRA database.

Limiting the number of issues returned from a search view such as an RSS feed

JIRA allows you to view search results in several different formats, including Word, Excel, RSS, or XML.

A search view that returns too many issues can take a long time for JIRA to complete and can use a large amount of memory. It can be a factor in [OutOfMemoryErrors](#) in JIRA.

An large RSS feed of search results can be particularly problematic, because:

- the user's RSS reader will continue to make the request periodically (for example, every hour)
- since the RSS reader makes the request, not the user directly, the user is unaware that the request takes a long time or is failing

You can use the following three properties in `jira-config.properties` to limit the number of issues returned by a search view.

See [Making changes to `jira-config.properties`](#) for the details of how to make and apply changes to your `jira-config.properties` file.

```
jira.search.views.default.max
```

The `jira.search.views.default.max` property sets a 'soft' limit on the number of issues returned. It has a default value of 1000. You can set it to 100 (for example), by specifying the following in your `jira-config.properties` file:

```
jira.search.views.default.max = 100
```

For an RSS or XML view, JIRA applies the limit by appending the `tempMax` parameter to the URL of the search view. For example:

- <http://jira.atlassian.com/sr/jira.issueviews:searchrequest-xml/temp/SearchRequest.xml?&type=2&pid=10240&resolution=-1&sorter/field=issuekey&sorter/order=DESC&tempMax=200>

In the above example, JIRA will limit the number of issues returned to 200 (in this example).

However users can override this 'soft' default by removing the `tempMax` parameter from the URL or by

increasing the value of `tempMax`.

```
jira.search.views.max.limit
```

The `jira.search.views.max.limit` property sets a 'hard' limit on the number of issues returned. It has a default value of 1000. You can set this property's value to 200 (for example), by specifying the following in your `jira-config.properties` file:

```
jira.search.views.max.limit = 200
```

If a user makes an issue view request that would return more than 200 issues (in this example), JIRA does not return the issues but instead returns a 403 (Forbidden) error. While the user might not be happy, it prevents JIRA from consuming lots of resources and possibly running out of memory.

Make sure you set the value of `jira.search.views.max.limit` to greater than or equal to the 'soft' limit set by `jira.search.views.default.max`. Otherwise all search views that would return issues limited by the default 'soft' limit will instead return a 403 (Forbidden) error.

```
jira.search.views.max.unlimited.group
```

You may have a requirement for most users to have the limit imposed on them, but a few users to be exempt from the limit. One example of this is if your JIRA instance is Internet facing. You may want external (Internet) users to have the limit imposed on them, but for internal users to be able to produce unlimited search views. You can use the `jira.search.views.max.unlimited.group` property to achieve this.

The `jira.search.views.max.unlimited.group` property is disabled by default, by being either absent from your `jira-config.properties` file or present but disabled with a preceding '#'. If you enable this property in your `jira-config.properties` file, you must specify a valid group for its value or leave it empty. For example:

```
jira.search.views.max.unlimited.group = jira-administrators
```

Users exempted from the limit via this technique will still have to add the `tempMax` parameter to the URL for an RSS or XML view, as described [above](#), in order to exceed the `jira.search.views.default.max` soft limit.

Configuring file attachments

When file attachments are enabled, your users will be allowed to attach files and screenshots to JIRA issues. This requires space on the server for storing the attachments.

File attachments are enabled by default. If you wish, you can [configure](#) the way JIRA handles attachments, or disable this feature altogether. ⚠ Attachments are not stored in JIRA's database and so will need to be [backed up](#) separately. **Note:**

- Your users must also have the **Create Attachments** [permissions](#) to attach files to issues.
- 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).
- For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global permission](#).

Configuring attachment settings

1. Choose



> **System**.

2. Select **Advanced > Attachments** to open the Attachment page, which states whether attachments are on or off.

Attachments [Edit Settings](#) ?

? To enable a user to attach files, ensure that the user has the **Create Attachments** permission for a particular project.

Allow Attachments	ON
Attachment Path The absolute or relative path to the directory under which attached files will be stored. When changing this path you will have to manually copy any existing attachments across to the new directory.	Default Directory [/data/jirastudio/jira/home/data/attachments]
Attachment Size The total upload size limit of attachments.	100.00 MB
Enable Thumbnails Enables the creation of thumbnail images of image attachments.	ON
Enable ZIP support Enables the ability for users to download all the attachments of an issue as a single ZIP file.	ON

3. Click the **Edit Settings** button, which opens the **Edit Attachment Settings** dialog box:

Edit Attachment Settings

Attachment Path **Use Default Directory**
/data/jirastudio/jira/home/data/attachments

Disable Attachments

Attachment Size
The total upload size limit in bytes.

Enable Thumbnails **ON**
 OFF
Enable creation of thumbnails of image attachments.
Attachments must be enabled to enable thumbnails.

Enable ZIP support **ON**
 OFF
Enable ZIP support for attachments. Attachments must be enabled to enable ZIP support.

4. In the **Attachment Path** field, choose the **Use Default Directory** option. If you see more attachment path options than what is shown in the screenshot above, please refer to the [note below](#).
- ? As mentioned above, if you have not logged in as a user with the **JIRA System Administrators** global permission, then this option will not be available to you.
5. In the **Attachment Size** field, specify the maximum attachment size. The default (per file) is 10485760 bytes (10 MB). The maximum attachment size (per file) is 2147483647 bytes (2 GB).
6. (Optional) In the **Enable Thumbnails** field, ensure that **ON** is selected if you wish to display image file attachments as thumbnails (or miniature previews) when viewing an issue. When this setting is enabled, JIRA automatically creates thumbnails of the following types of image attachments:

- GIF
- JPEG
- PNG

Please refer to the [info note below](#) for more information about thumbnails. If you use Linux, please refer to the [Linux note below](#).

7. (Optional) In the **Enable ZIP Support** field, ensure that **ON** is selected if you wish to view the contents of zip files attached to an issue and allow all files attached to an issue to be downloaded as a single ZIP file.
8. Click the **Update** button to update JIRA's attachment settings.

To attach files to issues, the appropriate users, groups or project roles must first be assigned the **Create Attachments** permission for the relevant project(s).

To allow these users or group/project role members to delete their own attached files from issues, they must also be assigned the **Delete Own Attachments** permission for these projects too.

There is no need to proceed any further if:

- the permission schemes used by your project(s) already have the **Create Attachments** (and **Delete Own Attachments**) permission, or
- your project(s) use JIRA's built-in **Default Permission Scheme**.

However, if you wish to configure these permissions, proceed with the steps in the section below.

Configuring create/delete attachment permissions

1. Choose



> **Issues.**

2. Select **Permission Schemes** to open the Permission Schemes page, which displays a list of all permission schemes in your JIRA system and the projects that use each scheme.
3. For each relevant permission scheme:
 - a. Click the **Permissions** link associated with the relevant permission scheme to edit that scheme's permissions.

Permission Schemes ?		
Permission Schemes allow you to create a set of permissions and apply this set of permissions to any project. All permissions within a scheme will apply to all projects that are associated with that scheme. The table below shows the permission schemes currently configured for this server. For permissions that apply to all projects see Global Permissions .		
Name	Projects	Operations
Copy of Design Permissions		Permissions · Copy · Edit · Delete
Default Permission Scheme <small>This is the default Permission Scheme. Any new projects that are created will be assigned this scheme.</small>	<ul style="list-style-type: none"> • Angry Nerds • AtlasBoard • Atlassian Customer Search • Atlassian Design Guidelines • Confluence Family GTM 	Permissions · Copy · Edit

- b. On the **Edit Permissions** page, locate **Create Attachments** within the **Attachment Permissions** section and click the **Add** link.
- c. In the user selection options on the right of the **Add New Permission** page, select the relevant (groups of) users or roles and then click the **Add** button.

Add New Permission ?

Permission Scheme: **Copy of Design Permissions**

Please select the type of permission you wish to add to this Permission Scheme

Permissions

- Manage Watchers
- Add Comments
- Edit All Comments
- Edit Own Comments
- Delete All Comments
- Delete Own Comments
- Create Attachments

(Select the permissions that you want to assign).

Reporter

Group Anyone

Single User Start typing to get a list of possible matches.

Project Lead

Current Assignee

User Custom Field Value Choose a custom field

Project Role Choose a project role

Group Custom Field Value Choose a custom field

✔ To allow these users or group/project role members to delete their own attachments, do not forget to assign them the **Delete Own Attachments** permission too.

i Choosing a custom Attachment Path:

- If you upgraded JIRA with an [XML backup](#) from a JIRA version prior to 4.2 and used a custom directory for your attachment path, you can choose between using this custom directory (which cannot be edited) or the default directory for your attachment path location. However, once you switch to using the default directory, you can no longer choose the custom directory option.
- The default directory location is the `data/attachments` subdirectory of the [JIRA home directory](#).
- To be able to change the default path, [create a symbolic link](#) to the new path.

i More information about thumbnails:

- You can 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, together with the original attachments. The thumbnail images are denoted by `'_thumb_'` in their file names.

i Thumbnail image generation on Linux:

- Your 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.
- The following java system property must be set: `-Djava.awt.headless=true`

Advanced configurations

You can implement the following [advanced configurations](#) to modify the way JIRA handles attachments. However, these are not accessible through JIRA's attachment settings. One of these advanced configurations can be modified as an 'Advanced Setting' in JIRA's administration area, although the remaining two are implemented by defining properties in your `jira-config.properties` file.

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 `jira-config.properties` file in your [JIRA home directory](#).
 - i** See [Making changes to the jira-config.properties file](#) for more information.
3. Edit the values of the following properties:
 - `jira.thumbnail.maxwidth` — thumbnail width in pixels

- `jira.thumbnail.maxheight` — thumbnail height in pixels
- i** If neither of these properties exist in your `jira-config.properties` file, add them to the file. For example, specify the following for a thumbnails that are 100 pixels wide:

```
jira.thumbnail.maxwidth = 100
```

4. Delete all existing thumbnail images within the `attachments` directory (that is, those containing `'_thumb_'` in the filename).
5. Restart JIRA.

After restarting JIRA, all thumbnails will be recreated automatically using the new dimensions.

Configuring ZIP-format file accessibility

By default, JIRA allows you to access common ZIP-format files, with file extensions like `'.zip'` and `'.jar'` (Java archive files). However, there are numerous other ZIP-format files to which JIRA does not permit access by default. You can permit access to these files by doing the following:

1. Stop JIRA.
2. Edit the `jira-config.properties` file in your **JIRA home directory**.
 - i** See [Making changes to the `jira-config.properties` file](#) for more information.
3. Remove the extensions from the `jira.attachment.do.not.expand.as.zip.extensions.list` property of the file types whose contents you wish to access in JIRA.
 - ✓** If this property does not exist in your `jira-config.properties` file, add the name of this property, followed `'='`, followed by the content of the `<default-value/>` element copied from your JIRA installation's `jpm.xml` file. Then, begin removing the extensions of file types whose contents you wish to access in JIRA.
4. Restart JIRA.

Configuring the number of files shown in the content of ZIP-format files on issues

By default, JIRA shows a maximum of 30 files in the content of ZIP-format files attached to an issue. To change this maximum value:

1. Access JIRA's **Advanced Settings** page. (See [Advanced JIRA configuration](#) for more information.)
2. Edit the value of the `jira.attachment.number.of.zip.entries` property by clicking the existing value and specifying the maximum number of attachments you want to show on an issue.
3. Click the **Update** button to save the new value in the JIRA database.

Configuring issue linking

About issue linking

Issue linking allows you to create an association between issues on either the same or different JIRA servers. For instance, an issue may *duplicate* another, or its resolution may *depend* on another's. New installations of JIRA come with four default types of links:

- **relates to / relates to**
- **duplicates / is duplicated by**
- **blocks / is blocked by**
- **clones / is cloned by**

Issue linking also allows you to:

- Create an association between a JIRA issue and a Confluence page.
- Link a JIRA issue to any other web page.

You can add, edit or delete link types to suit your organization, as described below.

i Note:

- Your users must have the [Link Issues permission](#) before they can link issues.
- Issue linking must be enabled in order for your users to be able to link issues. **Issue linking is enabled**

by default. If your organization does not require the ability to link issues, you can disable it globally for all users, as described [below](#).

- If you want to link JIRA issues to those on a different JIRA server or to Confluence pages, see [Configuring issue linking for external applications](#) (below) for details on how to set this up.
- For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Adding a link type

1. Choose



> **Issues.**

2. Select **Issue Features > Issue Linking** to open the Issue Linking page.
3. In the 'Add New Link Type' form at the end of the page:
 - Enter 'Causes' in the **Name** text field.
 - Enter 'causes' in the **Outward Link Description** text field.
 - Enter 'is caused by' in the **Inward Link Description** text field.
4. Click the **Add** button.
5. This returns to the **Issue Linking** page, with a new section listing the **Causes** link type.

Screenshot: the 'Issue Linking' administration page

Issue linking is currently ON. ?

To deactivate issue linking, simply click below.

i For the users you wish to be able to link issues, ensure that they have the **Link Issues** permission for that particular project.

Name	Outward Description	Inward Description	Operations
Blocked	blocks	blocked by	Edit · Delete
Bonfire Testing	testing discovered	discovered while testing	Edit · Delete
Cloners	is cloned by	clones	Edit · Delete
Dependent	depends on	is depended on by	Edit · Delete
Derived	derived from	derives	Edit · Delete
Design	has design on	is design for	Edit · Delete
Duplicate	duplicates	is duplicated by	Edit · Delete
Epic	belongs to Epic	is the Epic for	Edit · Delete
Relates	relates to	relates to	Edit · Delete
User Test	was in test	participants were	Edit · Delete
caused	causes	caused by	Edit · Delete

Add New Link Type

Add a new link type

Name
(eg "Duplicate")

Outward Link Description
(eg "duplicates")

Inward Link Description
(eg "is duplicated by")

Editing or deleting a link type

i It is recommended that you do not edit or delete the **Clones** link type, as this is used to automatically link

issues when they are cloned.

1. Choose



> **Issues.**

2. Select **Issue Features > Issue Linking** to open the Issue Linking page.
3. Locate the link type you wish to edit or delete, and click the link type's associated **Edit/Delete** link in the **Operations** column.

Configuring issue linking for external applications

It is possible to create links to issues on a remote JIRA instance or pages on a Confluence instance (running Confluence version 4.0 or later). To do this, create *fully reciprocal application links* between your JIRA instance and the remote JIRA or Confluence instance. Fully reciprocal application links mean that:

1. An application link must be configured on each server to the other.
2. Each of these application links must have both [incoming and outgoing authentication](#) configured to each other's servers.

To configure fully reciprocal application links between your JIRA instance and a remote JIRA or Confluence instance:

1. Log in as a user with the **JIRA System Administrators** [global permission](#).
2. Create an application link to your remote JIRA or Confluence instance. (See [Using AppLinks to link to other applications](#) for details.) When creating the link:
 - a. During step 2 of the wizard, ensure you choose the option to create a link from the remote server back to your server.
 - b. During step 3 of the wizard, choose the **These servers fully trust each other** option. This will ensure that [incoming and outgoing authentication](#) is configured for the application link on each server to the other server.
3. If you configured a fully reciprocal application links between your JIRA instance and a Confluence instance, ensure that the Confluence instance's system administrator has enabled the **Remote API (XML-RPC & SOAP)** feature, since this Confluence feature is disabled by default. See [Enabling the Remote API](#) in the Confluence documentation for details.

 If you do not enable this feature, JIRA will not be able to communicate with Confluence. As a result, your users:

 - a. Will see **Failed to load** messages in the Confluence Wiki page links they create on JIRA issues.
 - b. Will not be able to search for Confluence pages using the **Find a Confluence page** dialog box.

 **Please Note:** You can create a one-way application link from your JIRA instance to a remote JIRA instance or Confluence instance. However, some loss of functionality will be experienced by your users when they create remote links. For instance, if your users create a link to a remote JIRA issue, they will find that the **Create reciprocal link** check box on the **Link** dialog box will not function correctly. Hence, it is recommended that you create fully reciprocal links instead.

Disabling issue linking

1. Choose



> **Issues.**

2. Select **Issue Features > Issue Linking** to open the Issue Linking page.
3. A status message indicates whether issue linking is enabled. If issue linking is enabled, click the **Deactivate** button. The **Issue Linking** page reloads, stating that linking is disabled.

Configuring the order of linked issues displayed on the 'view issue' page

JIRA system administrators can define the order in which linked issues are displayed in the Issue Links section on the 'view issue' page. This is done by editing the value of the `jira.view.issue.links.sort.order` property on JIRA's [Advanced settings](#) page.

Specify the fields by which to sort issues in the **Issue Links** section on the 'view issue' page by entering the appropriate 'value' for each field in a comma-separated list. This property behaves similarly to a list of values

specified after the ORDER BY keyword in JIRA Query Language (JQL), whereby sorting is conducted by the first and then subsequent fields specified in the list.

The `jira.view.issue.links.sort.order` property can accept the following individual field values: 'key', 'type', 'status', 'priority' and 'resolution'.

Configuring issue cloning

JIRA's issue cloning behavior can be modified by [JIRA system administrators](#).

Configuring cloned issue linking behavior

By default, when an issue is cloned, JIRA will automatically create a link between the original and cloned issue using the pre-existing link type name 'Cloners'.

You can change this default behavior by editing the `jira.clone.linktype.name` property of your `jira-config.properties` file.

 If this property does not exist in your `jira-config.properties` file, add it to the file.

- If this property has a value, JIRA will use the pre-existing link type whose name is the value specified for this property.
- If this property has no value, JIRA will not create links between original and cloned issues.

Configuring the cloned issue summary field prefix

By default, the 'Summary' field of a cloned issue is prefixed with the string 'CLONE - ' to indicate that the issue is a clone.

To change this prefix or prevent the addition of prefixes on cloned issues:

1. Access JIRA's Advanced Settings page. (See [Advanced JIRA configuration](#) for more information.)
2. Edit the value of the `jira.clone.prefix` property by clicking the existing value and specifying a different prefix for the 'Summary' field of cloned issues.
 -  Specifying no value prevents a prefix being added to the 'Summary' field of cloned issues.
3. Click the **Update** button to save the new value in the JIRA database.

Configuring the whitelist

JIRA administrators can choose to allow incoming and outgoing connections and content from specified sources by adding URLs to the whitelist.

JIRA will display an error if content has been added that is not from an allowed source, and prompt the user to add the URL to the whitelist.

[Application Links](#) are automatically added to the whitelist. You do need to manually add them.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** [global](#) permission.

Add allowed URLs to the whitelist

1. Choose  **> System.**
2. Select **Security > Whitelist** to open the Whitelist page.

Whitelist

You can set up whitelisting to restrict incoming and outgoing connections and content. [Learn more](#).

Application Links are automatically whitelisted. To add more, visit the [Application Links](#) page.

Test a URL:

Outgoing
Incoming

Expression	Type	Allow Incoming
<input type="text" value="{scheme}://{domain[:port]}"/>	Domain name	<input type="checkbox"/>
<input type="text" value="http://www.atlassian.com/*"/>	Wildcard expression	<input type="checkbox"/>

3. On the **Whitelist** page, enter the URL or expression you want to allow.
4. Choose the **Type** of expression (see *Expression Types* below for examples).
5. Choose **Allow Incoming** if you need to allow CORS requests (see [below](#)).
6. Choose **Add**.

Your URL or expression appears in the whitelist.

To test that your whitelisted URL is working as expected, you can enter a URL in the **Test a URL** field. Icons will indicate whether incoming or outgoing traffic is allowed for that URL.

Expression types

When adding a URL to the whitelist, you can choose from a number of expression types.

Type	Description	Example
Domain name	Allows all URLs from the specified domain.	<code>http://www.example.com</code>
Exact match	Allows only the specified URL.	<code>http://www.example.com/thispage</code>
Wildcard Expression	Allows all matching URLs. Use the wildcard <code>*</code> character to replace one or more characters.	<code>http://*example.com</code>
Regular Expression	Allows all URLs matching the regular expression.	<code>http(s)?://www\.example\.com</code>

Allow incoming

Allow Incoming enables [CORS](#) requests from the specified origin. The URL must match the format `scheme://host[:port]`, with no trailing slashes (`:port` is optional). So `http://example.com/` would not allow CORS requests from the domain `example.com`.

Disabling the whitelist

The whitelist is enabled by default. You can choose to disable the whitelist however this will allow all URLs, including malicious content, and is not recommended.

1. Choose  **> System**.
2. Select **Security > Whitelist** to open the Whitelist page.
3. On the **Whitelist** page, click the **Turn off whitelist** button.
4. Choose **Confirm**.

All URLs will now be allowed. Unless your instance is running in an environment without internet access, we do

not recommend disabling the whitelist.

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.

If sub-tasks are enabled and you have defined at least one sub-task issue type, your users will be able to:

- create sub-tasks
- convert issues to sub-tasks (and vice versa)

On this page:

- [Disabling sub-tasks](#)
- [Enabling sub-tasks](#)
- [Defining sub-task issue types](#)
- [Blocking issue workflows by sub-task status](#)
- [Configuring sub-task fields displayed on parent issues](#)

Disabling sub-tasks

Sub-tasks are enabled by default. However, this feature can be disabled from the Sub-Tasks administration page.

i *Sub-tasks will be disabled by default if your JIRA installation was upgraded from a version prior to 4.2 that had sub-tasks disabled.*

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose  **> Issues**. Select **Issue Types > Sub-Tasks** to open the Sub-Tasks page.
3. Click the **'Disable' Sub-Tasks** link. The page reloads and informs you that sub-tasks are now disabled.

i **Please note:** Sub-tasks cannot be disabled if one or more sub-tasks exists in the system. You must remove any existing sub-tasks (or convert them to standard issues) before you can disable this feature.

Enabling sub-tasks

Sub-tasks can be enabled from the Sub-Tasks administration screen.

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose  **> Issues**. Select **Issue Types > Sub-Tasks** to open the Sub-Tasks page.
3. Click the **'Enable' Sub-Tasks** link. The page will reload and inform you that the sub-tasks are now enabled.

i A default *sub-task issue type* is automatically available for use. You can edit it by clicking its **Edit** link in the Operations column.

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 at least one sub-task issue type must be defined in JIRA for users to be able to create sub-tasks.

Sub-task issue types can be customized on the Sub-Tasks administration page (described above). The

Sub-Tasks administration page also allows you to create, edit (i.e. the name, description or icon), and translate your sub-task issue types.

Creating a sub-task issue type

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



> **Issues**. Select **Issue Types > Sub-Tasks** to open the Sub-Tasks page.

Sub-Tasks

+ Add New Sub-Task Issue Type

Sub-Tasks are currently turned **ON**. You can manage your sub-tasks as part of standard issue types [here](#).

- [Disable Sub-Tasks](#)
- [Translate Sub-Tasks](#)
- [Manage Sub-Tasks](#)

Name	Description	Icon	Operation
Sub-task	The sub-task of the issue		Edit

3. Click **Add New Sub-task Issue Type** button to open the Add New Sub-task Issue Type dialog box.
4. Complete the following:

- **Name** — enter a short phrase that best describes your new sub-task issue type.
- **Description** — enter a sentence or two to describe when this sub-task issue type should be used.
- **Icon URL** — supply the path of a image from an accessible URL or an image that has been placed somewhere inside `<jira-application-dir>/images/icons` of your **JIRA** application installation directory.

Editing a sub-task issue type

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



> **Issues**. Select **Issue Types > Sub-Tasks** to open the Sub-Tasks page.

3. Click the **Edit** link (in the Operations column) for the sub-task issue type that you wish to edit.
4. Edit the **Name**, **Description**, and/or **Icon**, as described above in [Creating a sub-task issue type](#).

Deleting a sub-task issue type

You can only delete sub-task issue types through the Manage Issue Types page. For details, see [Deleting an issue type](#).

Blocking issue workflows by sub-task status

It is possible to restrict the progression of an issue through workflow depending on the status 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' status.

Configuring sub-task fields displayed on parent issues

JIRA system administrators can define which fields of sub-tasks are displayed in the Sub-tasks section on the 'view issue' page of a parent issue (which contains one or more sub-tasks). This is done by editing the value of the `jira.table.cols.subtasks` property on JIRA's [Configuring advanced settings](#) page.

Specify which fields you want to show in the **Sub-tasks** section of a parent issue's 'view issue' page by entering the appropriate 'value' for each field in a comma-separated list. The `jira.table.cols.subtasks` property can accept the values indicated in right-hand column of the `IssueFieldConstants` table on the [Constant Field Values](#) page (of JIRA's API documentation).

i Please note:

- The order of each value in this list determines the order of their representative fields in the Sub-tasks section of a parent issue's 'view issue' page.
- The `summary` field is a mandatory value which assumes first position in this property's value.

Managing shared filters

A **filter** is a saved issue search. JIRA users can create and manage their own filters and filter subscriptions.

A **shared filter** is a filter whose creator has shared that filter with other users. When a shared filter is created by a user, that user:

- Initially 'owns' the shared filter.
- Being the owner, can edit and modify the shared filter.

JIRA administrators can change the ownership of any user's shared filter, which allows the shared filter to be edited and modified by its new owner.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** and **JIRA Users** global permissions.

Changing the ownership of a shared filter

Before changing the ownership of a shared filter, ensure that you inform the shared filter's current owner of your intentions.

1. Choose



> **System.**

2. Select **Shared Filters** to open the Search Shared Filters page.

Search Shared Filters

Find and modify filters that are shared with any group or role.

Search

Searches in the filter's name and description.

Search

Owner

Start typing to get a list of possible matches.

Name +	Owner	Shared with	Popularity	
Bugs In Progress	Mia Culpa (mia)	• Shared with everyone	1	...
Experimental Filter	Jose Fonte (jose)	• Shared with everyone	1	...
Products Filter	Alana Smith (alana)	• Shared with everyone	1	...

3. Enter your search criteria into the 'Search' field and click the '**Search**' button. A list of shared filters matching your search criteria is shown below. Each shared filter indicates its:
 - Current owner — this is originally the user who created the shared filter
 - List of shares applied to the shared filter by its owner
 - Popularity — the number of users who have selected that shared filter as a 'favorite'.
4. Click the 'cog' icon to the right of the shared filter whose ownership you wish to change and select '**Change Owner**'.
5. In the 'Change Owner' dialog box, enter the username (or name) of the user who will become the new owner of the shared filter.
6. Select the appropriate user from the drop-down list and click the '**Change Owner**' button.

i Please note:

- A shared filter can only be edited by the shared filter's owner. The owner of a shared filter can only modify that filter's shares and search criteria too.
- You cannot change the ownership of a shared filter to a user who:
 - already has a shared filter with exactly the same name, or
 - does not have permission to view the shared filter.

On this page:

- Changing the ownership of a shared filter
- Deleting a shared filter

Deleting a shared filter

Before deleting a shared filter, then out of common courtesy, ensure that you inform the current owner of the shared filter of your intentions.

1. Choose



> **System.**

2. Select **Shared Filters** to open the Search Shared Filters page.

Search Shared Filters

Find and modify filters that are shared with any group or role.

Search

Searches in the filter's name and description.

Search

Owner

Start typing to get a list of possible matches.

1 - 20 of 596 ▶

Name ↕	Owner	Shared With	Popularity
0. JIRA Release Tracking	Paul [Redacted]	• Shared with all users	1
4.1 Todo	Nick [Redacted]	• Shared with all users	0
4.1 Triaged	Paul [Redacted]	• Shared with all users	3
4.1 Triaged NON-UI	Paul [Redacted]	• Shared with all users	2
4.1 Triaged UI	Paul [Redacted]	• Shared with all users	1

3. Enter your search criteria into the 'Search' field and click the '**Search**' button. A list of shared filters matching your search criteria is shown below. Each shared filter indicates its:
 - Current owner — this is originally the user who created the shared filter
 - List of shares applied to the shared filter by its owner
 - Popularity — the number of users who have marked that shared filter as a 'favorite'.
4. Click the 'cog' icon to the right of the shared filter you wish to delete and select '**Delete Filter**'. The 'Delete Filter' dialog box is shown.
 - The number of users who have marked the shared filter as a favorite is specified in this dialog box.
 - If any subscriptions are associated with this shared filter, a numbered link is provided leading to a page which indicates the shared filter's current subscribers.
5. If you are happy to proceed, click the '**Delete**' button to complete the action.

Managing shared dashboards

A **dashboard** is a customizable page that can display many different types of information, depending on your areas of interest. JIRA users can create and manage their own dashboards.

A **shared dashboard** is a dashboard whose creator has shared that dashboard with other users. When a shared dashboard is created by a user, that user:

- Initially 'owns' the shared dashboard.
- Being the owner, can edit and modify the shared dashboard.

JIRA administrators can change the ownership of any user's shared dashboard, which allows the shared dashboard to be edited and modified by its new owner.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Changing the ownership of a shared dashboard

Before changing the ownership of a shared dashboard, ensure that you inform the shared dashboard's current owner of your intentions.

1. Choose

On this page:

- [Changing the ownership of a shared dashboard](#)
- [Deleting a shared dashboard](#)



> **System.**

2. Select **Shared Dashboards** to open the Search Shared Dashboards page.

Search Shared Dashboards

Search

Searches in the dashboard's name and description.

Owner

Start typing to get a list of possible matches.

1 - 20 of 88 ▶

Name ↕	Owner	Shared With	Popularity
5.0 Bug Analysis Dashboard 5.0 Bug Analysis	Panna <small>(Panna Panna)</small>	• Shared with all users	0
5.0 bugs	Paul <small>(Paul Paul)</small>	• Shared with all users	5
5.0.x Wallboard	Simone <small>(Simone Simone)</small>	• Shared with all users	3
5.1 Bug Fix Wallboard	Mark <small>(Mark Mark)</small>	• Shared with all users	2
5.1 Bug Graphs	Mark <small>(Mark Mark)</small>	• Shared with all users	1
5.1 Bugs	Mark <small>(Mark Mark)</small>	• Shared with all users	1

3. Enter your search criteria into the 'Search' field and click the '**Search**' button. A list of shared dashboards matching your search criteria is shown below. Each shared dashboard indicates its:
 - Current owner — this is originally the user who created the shared dashboard
 - List of shares applied to the shared dashboard by its owner
 - Popularity — the number of users who have selected that shared dashboard as a 'favorite'.
4. Click the 'cog' icon to the right of the shared dashboard whose ownership you wish to change and select '**Change Owner**'.
5. In the 'Change Owner' dialog box, enter the username (or name) of the user who will become the new owner of the shared dashboard.
6. Select the appropriate user from the drop-down list and click the '**Change Owner**' button.

i Please note:

- A shared dashboard can only be edited by the shared dashboard's owner. The owner of a shared dashboard can only modify that dashboard's shares and gadgets too.
- You cannot change the ownership of a shared dashboard to a user who:
 - already has a shared dashboard with exactly the same name, or
 - does not have permission to view the shared dashboard.

Deleting a shared dashboard

Before deleting a shared dashboard, ensure that you inform the shared dashboard's current owner of your intentions.

1. Choose



> **System.**

2. Select **Shared Dashboards** to open the Search Shared Dashboards page.

Search Shared Dashboards			
Search	Owner		
<input type="text"/>	<input type="text"/>		
Searches in the dashboard's name and description.	Start typing to get a list of possible matches.		
<input type="button" value="Search"/>			
			1 - 20 of 88 ▶
Name +	Owner	Shared With	Popularity
5.0 Bug Analysis Dashboard 5.0 Bug Analysis	Panna	• Shared with all users	0
5.0 bugs	Paul	• Shared with all users	5
5.0.x Wallboard	Simone	• Shared with all users	3
5.1 Bug Fix Wallboard	Mark	• Shared with all users	2
5.1 Bug Graphs	Mark	• Shared with all users	1
5.1 Bugs	Mark	• Shared with all users	1

- Enter your search criteria into the 'Search' field and click the '**Search**' button. A list of shared dashboards matching your search criteria is shown below. Each shared dashboard indicates its:
 - Current owner — this is originally the user who created the shared dashboard
 - List of shares applied to the shared dashboard by its owner
 - Popularity — the number of users who have marked that shared dashboard as a 'favorite'.
- Click the 'cog' icon to the right of the shared dashboard you wish to delete and select '**Delete Dashboard**'. The 'Delete Dashboard' confirmation message box is shown.
 - The number of users who have marked the shared dashboard as a favorite is specified in this message box.
- If you are happy to proceed, click the '**Delete**' button to complete the action.

Enabling logout confirmation

Administrators can configure JIRA to prompt users with a confirmation before logging them out.

⚠ Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators global permission**.

By default, JIRA will not prompt users to confirm logging out. To change this:

- Choose



> **System.**

- Select **General configuration** to open the Administration page.
- Locate the 'Options' section.

By default, JIRA will not prompt users to confirm logging out. To change this, click the **Edit Settings** button at the top of the page, and then enable or disable logout confirmation.

The **Never** and **Always** settings are self-explanatory. When set to **Cookie**, your JIRA users will only be prompted if they have logged in using a cookie (i.e. by selecting the '**Remember my login on this computer**' checkbox before they click the '**Log In**' button).

Rich text editing

Users expect a What You See Is What You Get (WYSIWYG) editing experience, and the rich text editor lets your users choose between **Text** mode, which is wiki markdown, or **Visual** mode, which is a WYSIWYG editor. The rich text editor is available on description fields, comment fields, and all Text field (multi line) custom fields that use the [wiki renderer](#).

Rich text editing is on by default, but you can disable it by:

- Choose



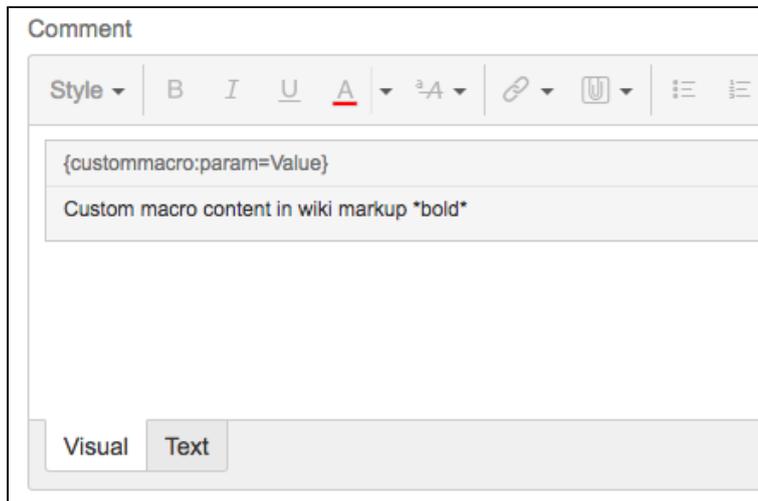
> **System.**

- Select **Rich text editor** in the **User Interface** section of the left hand menu.
- Click the **Enable rich text editing for users** toggle to disable or enable the editor. Hovering over the toggle will let you know if the editor is on or off.

Currently, the editor does not support:

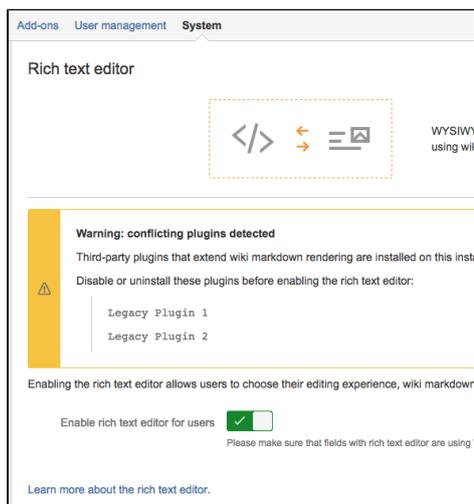
- Nested tables.
- Pasting rich text (plain text is fine) that contains complex formatting.

Third party macros provided by add-ons that aren't compatible with JIRA 7.8 are displayed in legacy mode:



The Macro header is not editable in Visual mode, and content within the macro is presented in text mode (wiki markup).

You can check the status of your add-ons on the **Rich text editor** configuration page:



Server optimization

This section of the documentation includes information on how to optimize your JIRA installation, such as performance testing and using the configuration tool. While not included in this section of the documentation, you may also be interested in our [Tuning database connections](#) page in the [Installation](#) section.

- [Configuring secure administrator sessions](#)
- [JIRA application cookies](#)
- [Preventing security attacks](#)
- [Using the JIRA application configuration tool](#)
- [Running JIRA applications as a Windows service](#)
- [Tuning garbage collection \(GC\)](#)

If you're looking for very specific information regarding your setup, and can't find it in the documentation, you may also wish to check out the [JIRA Knowledge Base](#), and [Atlassian Answers](#).

Configuring secure administrator sessions

JIRA protects access to its administrative functions by requiring a secure administration session in order to use the JIRA administration screens. (This

is also known as websudo.) When a JIRA administrator (who is logged into JIRA) attempts to access an administration function, they are prompted to log in again. This logs the administrator into a temporary secure session that grants access to the JIRA administration screens.

The temporary secure session has a rolling timeout (defaulted to 10 minutes). If there is no activity by the administrator in the JIRA administration screens for a period of time that exceeds the timeout, then the administrator will be logged out of the secure administrator session (note that they will remain logged into JIRA). If the administrator does click an administration function, the timeout will reset.

Note that Project Administration functions (as defined by the '[Project Administrator](#)' permission) do not require a secure administration session.

On this page:

- [Manually ending a secure administrator session](#)
- [Disabling secure administrator sessions](#)
- [Changing the timeout](#)
- [Developer notes](#)

Manually ending a secure administrator session

An administrator can choose to manually end their secure session by clicking the '**drop access**' link in the banner displayed at the top of their screen.

Disabling secure administrator sessions

Secure administrator sessions (i.e. password confirmation before accessing administration functions) are enabled by default. If this causes issues for your JIRA instance (e.g. if you are using a custom authentication mechanism), you can disable this feature by specifying the following line in your `jira-config.properties` file:

```
jira.websudo.is.disabled = true
```

i You will need to restart your JIRA server for this setting to take effect.

Changing the timeout

To change the number of minutes of inactivity after which a secure administrator session will time out, specify the `jira.websudo.timeout` property (in your `jira-config.properties` file) whose value is the number of minutes of inactivity required before a secure administration session times out.

For example, the following line in your `jira-config.properties` file will end a secure administration session in 10 minutes:

```
jira.websudo.timeout = 10
```

i You will need to restart your JIRA server for this setting to take effect.

Developer notes

If you have written a plugin that has webwork actions in the JIRA Administration section, those actions should have the `@WebSudoRequired` annotation added to the class (not the method or the package, unlike Confluence).

Please also see [How do I develop against JIRA with Secure Administrator Sessions?](#) and [Adding WebSudo Support to your Plugin](#).

JIRA application cookies

This page lists cookies stored in JIRA application users' browsers which are generated by JIRA itself. This page does not list cookies that may originate from 3rd-party JIRA plugins.

On this page:

- Authentication cookies
- Other JIRA cookies

Authentication cookies

JIRA uses [Seraph](#), an open source framework, for HTTP cookie authentication. JIRA uses two types of cookies for user authentication:

- The JSESSIONID cookie is created by the application server and used for session tracking purposes. This cookie contains a random string and the cookie expires at the end of every session or when the browser is closed.
- The 'remember my login' cookie (aka the 'remember me' cookie), `seraph.rememberme.cookie`, is generated by JIRA when the user selects the **Remember my login on this computer** checkbox on the login page.

 You can read about cookies on the [Wikipedia page about HTTP cookies](#).

The 'remember my login' cookie

The 'remember my login' cookie, `seraph.rememberme.cookie`, is a long-lived HTTP cookie. This cookie can be used to authenticate an unauthenticated session. JIRA generates this cookie when the user selects the **Remember my login on this computer** checkbox on the login page.

Cookie key and contents

By default, the cookie key is `seraph.rememberme.cookie`, which is defined by the `login.cookie.key` parameter in the `<jira-application-dir>/WEB-INF/classes/seraph-config.xml` file of your [JIRA installation directory](#).

The cookie contains a unique identifier plus a securely-generated random string (i.e. token). This token is generated by JIRA and is also stored for the user in the JIRA database.

Use of cookie for authentication

When a user requests a web page, if the request is not already authenticated via session-based authentication or otherwise, JIRA will match the 'remember my login' cookie (if present) against the token (also if present), which is stored for the user in the JIRA database.

If the token in the cookie matches the token stored in the database and the cookie has not expired, the user is authenticated.

Life of 'remember my login' cookies

You can configure the maximum age of the cookie. To do that you will need to modify the `<jira-application-dir>/WEB-INF/classes/seraph-config.xml` file of your [JIRA installation directory](#) and insert the following lines below the other `init-param` elements:

```
<init-param>
  <param-name>autologin.cookie.age</param-name>
  <param-value>2592000</param-value> <!-- The value of 30 days in
seconds -->
</init-param>
```

Other JIRA cookies

There are several cookies that JIRA uses for a variety of other purposes, such as to enhance JIRA's security

and to store basic presentation and browser capability states, including the type of search view that was last used and various other presentation states. JIRA users' authentication details are not stored by these cookies.

Cookie key	Purpose	Cookie contents	Expiry
atlassian.xsrf.token	Helps prevent XSRF attacks. Ensures that during a user's session, browser requests sent to a JIRA server originated from that JIRA server. For more information about XSRF checking by JIRA, see Form Token Checking on the Atlassian Developers site.	Your JIRA server's Server ID, a securely-generated random string (i.e. token) and a flag indicating whether or not the user was logged in at the time the token was generated.	At the end of every session or when the browser is closed.
jira.issue.navigator.type	Tracks which type of search view was last used (i.e. simple or advanced searching).	A string indicating the state of your last search view.	Approximately 10 years from the date it is set or was last updated.
AJS.conglomerate.cookie	Tracks which general tabs were last used (e.g. in JIRA's plugin manager) or expansion elements were last opened or closed.	One or more key-value strings which indicate the states of your last general tab views or expansion elements.	One year from the date it is set or was last updated.
UNSUPPORTED_BROWSER_WARNING	Acknowledges that the user has read a message displayed by JIRA indicating that the user's browser is not supported by JIRA.	A string which indicates that the user has clicked a button acknowledging they have read the message stating they are using an unsupported browser.	At the end of every session or when the browser is closed.
AJS.thisPage	Indicates that the user's browser does not support local storage. This relates to a mechanism used by JIRA to store field information in search views when the user clicks their browser's back button.	A string which indicates that the user's browser does not support local storage.	At the end of every session or when the browser is closed.

Preventing security attacks

This page provides guidelines which, to the best of our knowledge, will help prevent security attacks on your JIRA installation.

Use strong passwords

Administrators should use strong passwords

All your JIRA administrators, JIRA system administrators and administrators of all Atlassian applications should have strong passwords. Ask your administrators to update their passwords to strong passwords.

Do not use passwords that are dictionary words. Use mixed-case letters, numbers and symbols for your administrator passwords and make sure they are sufficiently long (e.g. 14 characters). We encourage you to refer to the [Strong Password Generator](#) for guidelines on selecting passwords.

Using strong passwords greatly increases the time required by an attacker to retrieve your passwords by brute force, making such an attack impractical.

On this page:

- Use strong passwords
- Apply JIRA security patches
- Protect against brute force attack
- Restrict network access to administrative sections of applications
- Restrict file system access by the application server

Administrators should have different passwords for different systems

As well as choosing a strong password, administrators should have *different* strong passwords for different systems. This will reduce the impact the attacker can have if they do manage to obtain administrator credentials on one of your systems.

Apply JIRA security patches

Apply the patches found in any security advisories that we release for your version of JIRA. These patches protect JIRA from recently detected privilege escalation and XSS vulnerabilities.

Protect against brute force attack

You can also actively protect your systems against repeated unsuccessful login attempts, known as "brute force" login attacks.

Enable brute force login protection on your Web server

It is possible to also enable brute force login protection on your web server by detecting repeated authentication failures in application logs. Once repeated login failures have been detected, you can set up an automated system to ban access to your web server from that particular IP address.

For more information on how to configure an automated approach to this kind of login prevention, refer to [Using Fail2Ban to limit login attempts](#).

Restrict network access to administrative sections of applications

An Atlassian application's administration interface is a critical part of the application; anyone with access to it can potentially compromise not only the application instance but the entire machine. As well as limiting access to only users who really need it, and using strong passwords, you should consider limiting access to it to certain machines on the network.

For more information on how to implement Apache blocking rules to restrict access to administrative or sensitive actions in:

- JIRA, refer to [Using Apache to limit access to the JIRA administration interface](#)
- Confluence, refer to [Using Apache to limit access to the Confluence administration interface](#)

You can use a similar approach to protecting all Atlassian applications.

Restrict file system access by the application server

The application server (e.g. Tomcat) runs as a process on the system. This process is run by a particular user and inherits the file system rights of that particular user. By restricting the directories that can be written to by the application server user, you can limit unnecessary exposure of your file system to the application.

For example, ensure that only the following directories can be written to by JIRA's application server:

- The following subdirectories of your [JIRA installation directory](#):
 - logs
 - temp
 - work
- Your [JIRA home directory](#)

Using the JIRA application configuration tool

The JIRA application **configuration tool** is an application that offers server-level JIRA configuration through a convenient GUI. This tool allows you to do the following:

- [Configure your JIRA home directory](#)
- [Configure your database connection](#)
- [Tune your database connection](#)
- [Configure the webserver](#), including the TCP ports that JIRA runs through and SSL configuration.

On this page:

- [Starting the JIRA configuration tool](#)
- [Configuring the JIRA home directory](#)
- [Configuring the database connection](#)
- [Configuring JIRA's web server](#)
- [Tuning JIRA's database connections](#)

Please note:

- The JIRA configuration tool requires a Java platform to be installed and configured on your operating system. If you need to install a Java platform to run this tool, we recommend using a [Java platform supported by JIRA](#) — refer to [JIRA requirements](#) for details.
- If you have a console-only connection to your JIRA server, you will need to perform these server-level configurations manually.
- Whenever you configure or reconfigure JIRA's server-level settings using this tool, JIRA **must be restarted** so it can recognize these changes.

Starting the JIRA configuration tool

The `JAVA_HOME` environment variable must be set to use the JIRA configuration tool. If it has not been set already, follow the instructions in [Installing Java](#) to set it.

- **Windows:** Open a command prompt and run `config.bat` in the `bin` sub-directory of the [JIRA installation directory](#).
- **Linux/Unix:** Open a console and execute `config.sh` in the `bin` sub-directory of the [JIRA installation directory](#).
 -  This may fail with the error as described in our [Unable to Start JIRA applications Config Tool due to No X11 DISPLAY variable was set error KB](#) article. Please refer to it for the workaround.

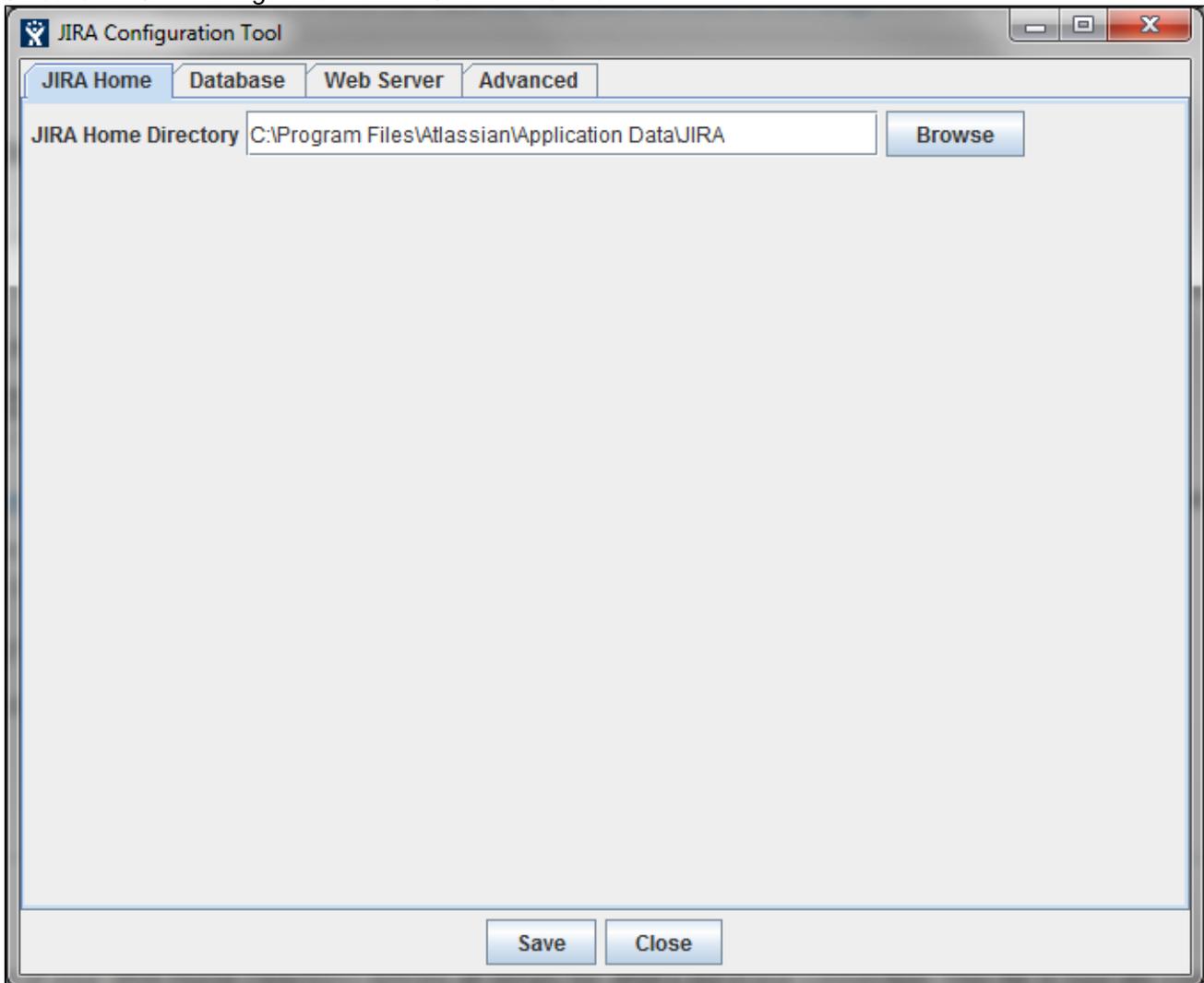
The JIRA configuration tool can be run with a graphical user interface or via a command-line interface using the `-c` or `--console` argument. The following sections show the graphical user interface, but the functionality is the same regardless of the interface.

Configuring the JIRA home directory

Your [JIRA home directory](#) allows you to set the folder that JIRA uses to store its various data files.

1. Click the JIRA **home** tab.
2. In the JIRA home directory field, type the full file path into the text field, or click the **Browse** button to browse for the location of your [JIRA home directory](#).
3. Click the '**Save**' button. Your changes are saved to the `jira-application.properties` file located in the `<jira-application-dir>` subdirectory of your [JIRA application installation directory](#). For more information, please see [Setting your JIRA home directory](#).

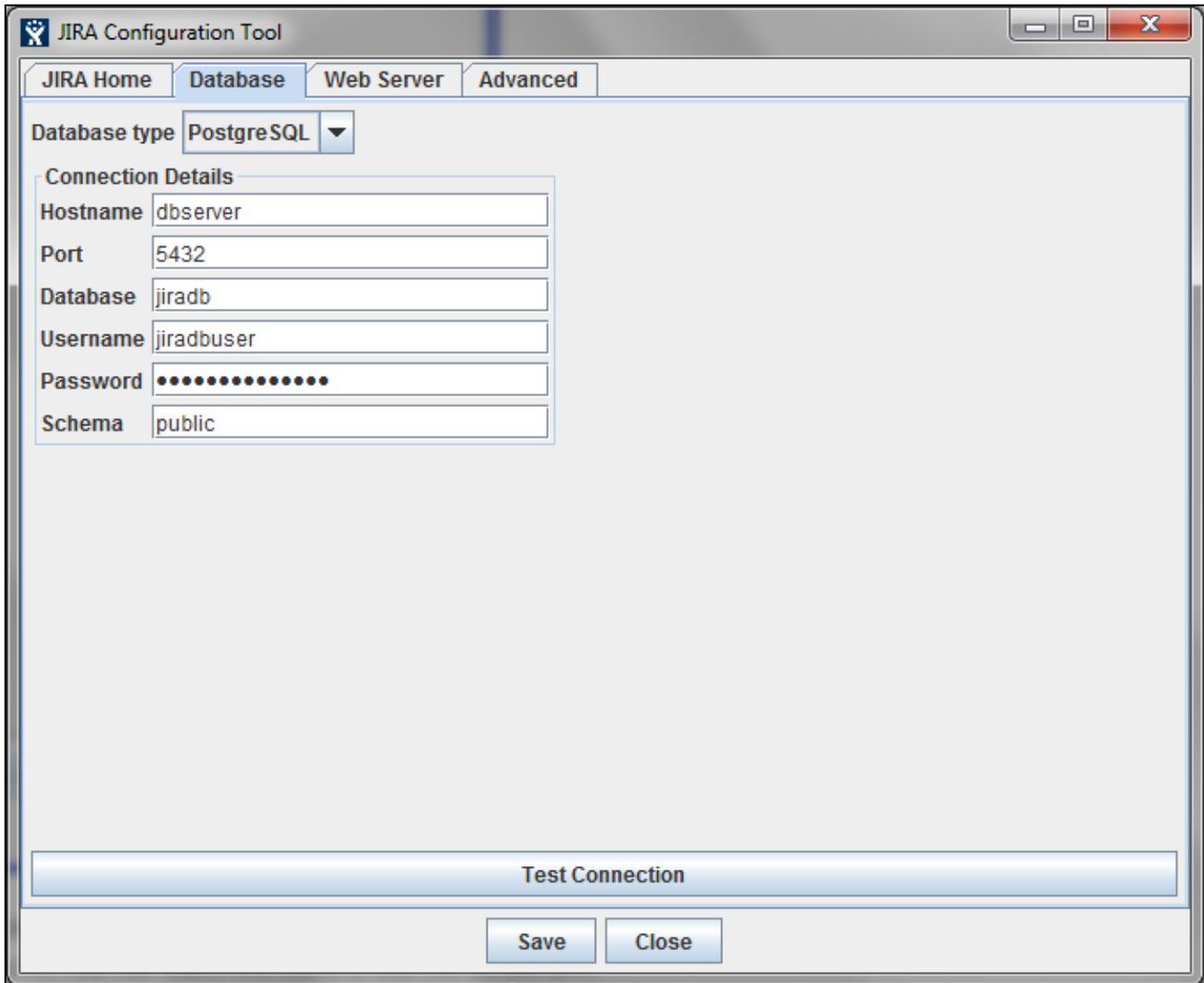
Screenshot: *JIRA configuration tool — 'JIRA Home' tab*



Configuring the database connection

To configure JIRA's database connection using the JIRA configuration tool, follow the appropriate procedure for your [database type](#):

Screenshot: *JIRA configuration tool — 'Database' tab*

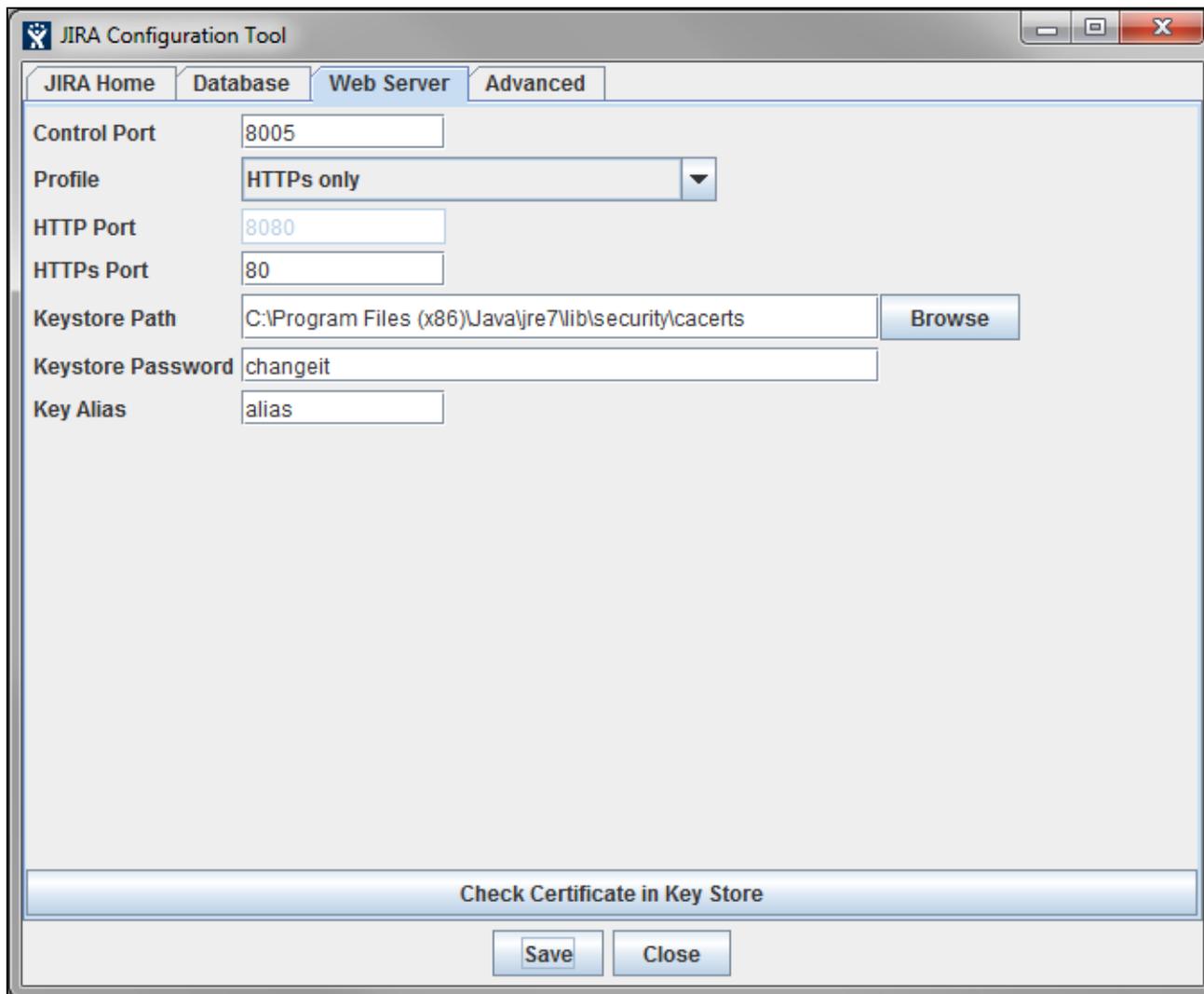


Configuring JIRA's web server

The JIRA configuration tool can also be used to configure JIRA's web server, specifically the TCP ports and the SSL configuration. Follow the relevant instructions linked below:

- [Changing JIRA's TCP ports](#)
- [Running JIRA applications over SSL or HTTPS](#)

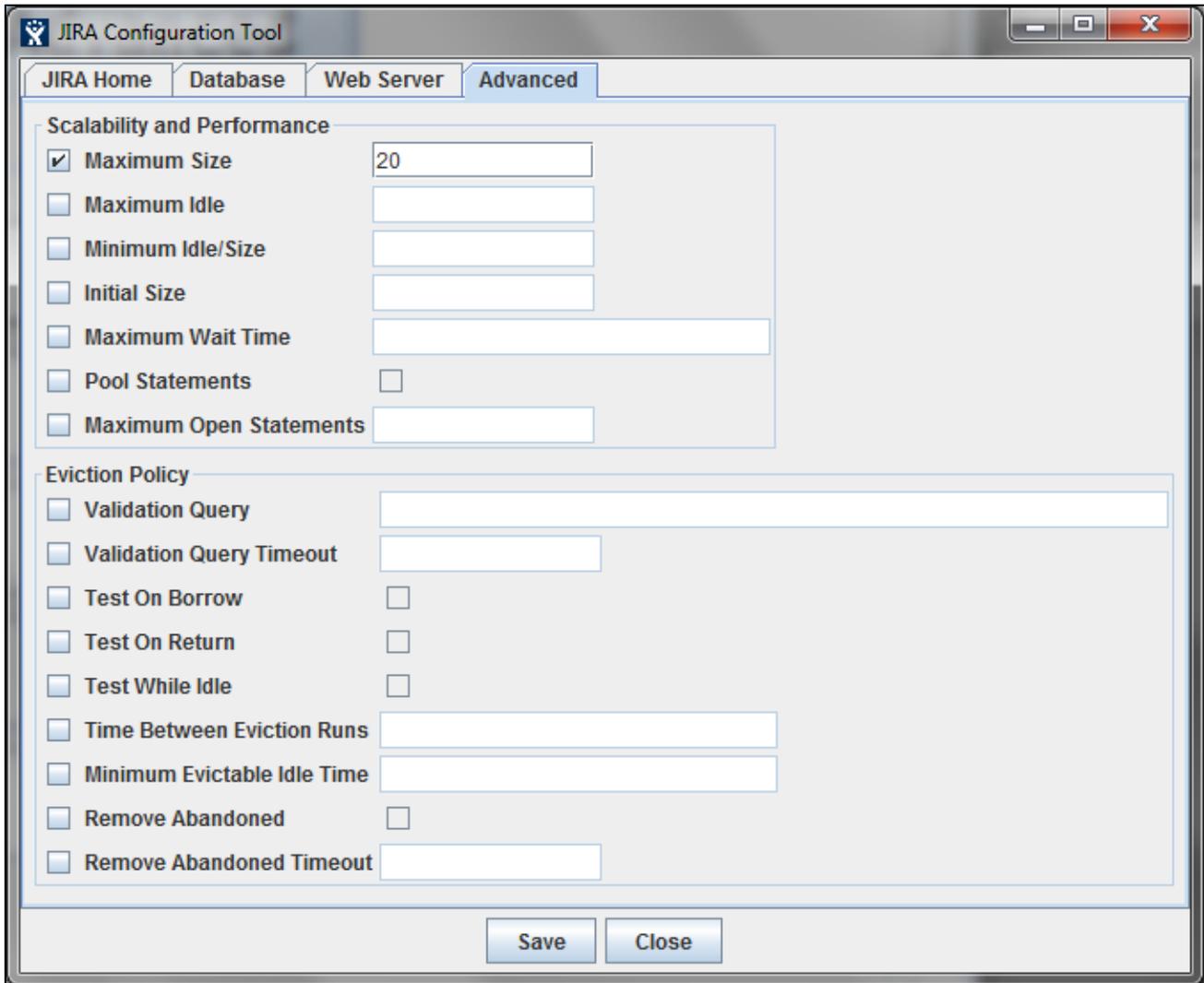
Screenshot: JIRA configuration tool — 'Web Server' tab



Tuning JIRA's database connections

For more information about the functionality of the **Advanced** tab, see [Tuning database connections](#).

Screenshot: JIRA configuration tool — 'Advanced' tab



Running JIRA applications as a Windows service

For long-term use, JIRA should be configured to automatically restart when the operating system restarts. For Windows servers, this means configuring JIRA to run as a **Windows service**.

Running JIRA as a Windows service has other advantages. When started manually, a console window opens, and there is a risk of someone accidentally shutting down JIRA by closing this window. Also, the JIRA logs are properly managed by the Windows service (found in `logs\stdout*.log` in your JIRA installation directory, and rotated daily).

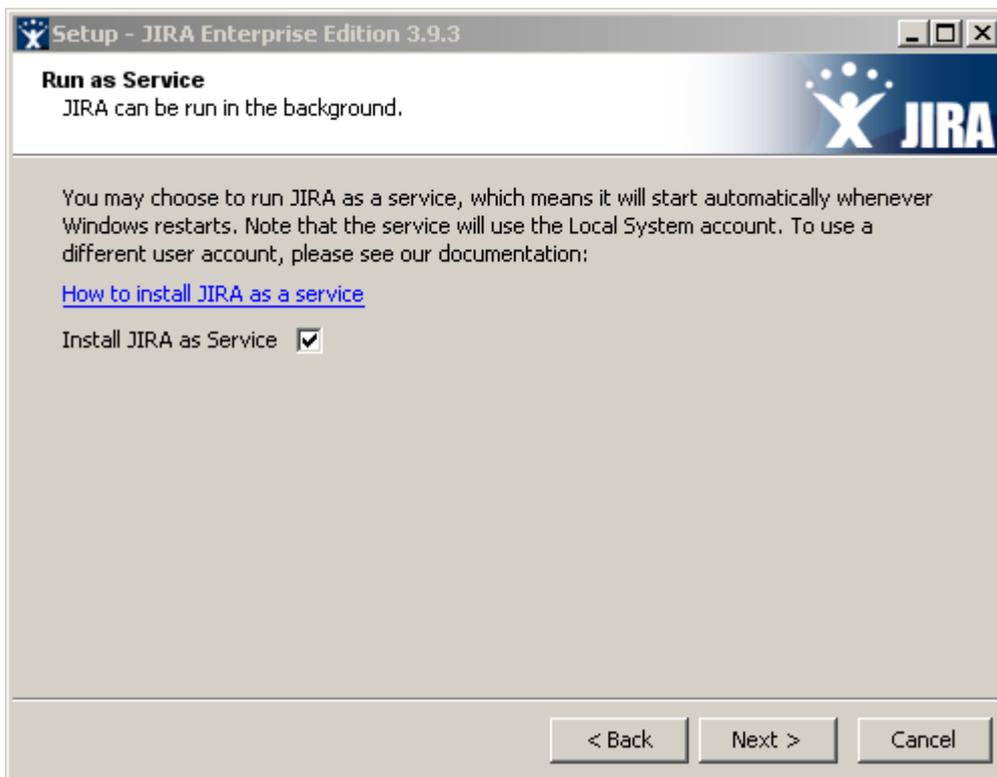
There are two ways to install JIRA as a service: via the installer, and manually.

On this page:

- Installing as a service with the installer
- Removing the JIRA service
- Changing the Windows user that the JIRA service uses
- Specifying the startup order of multiple services
- Locating the name of a service
- Troubleshooting

Installing as a service with the installer

The easiest way to get JIRA installed as a Windows service is by clicking the **'Install JIRA as Service'** checkbox when running the [Windows Installer](#):



You will need full Administrator rights on your Windows operating system for this installation process

to complete successfully.

Manually setting up JIRA to run as a service

You can still set up JIRA to run as a service, if any of the following situations apply to you:

- You did not use the [Windows Installer](#).
- You used the Windows Installer, but did not initially install JIRA as a service.

i Please note:

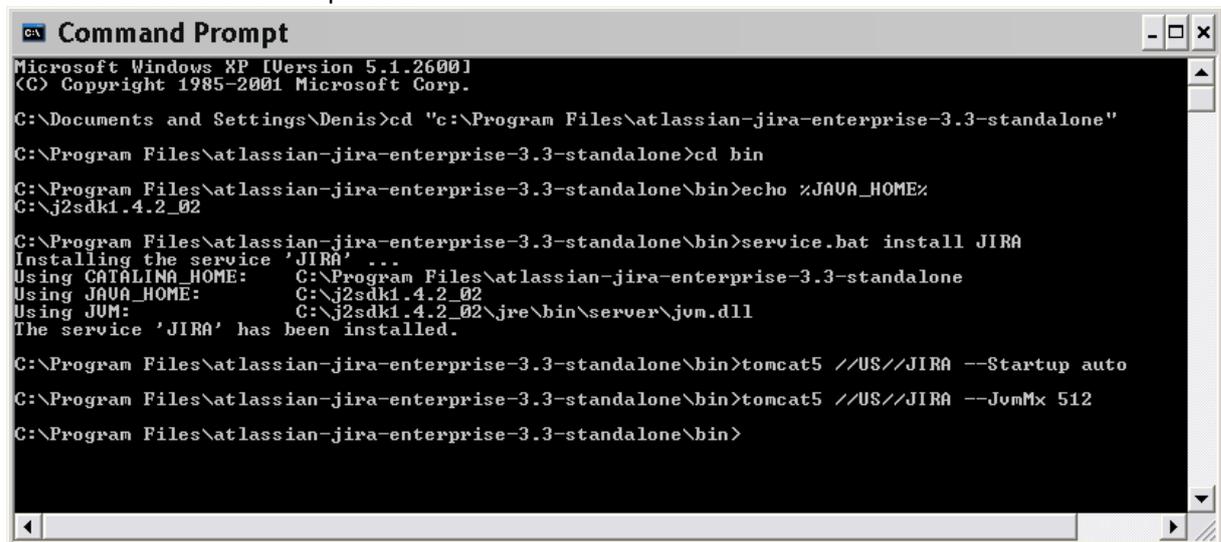
- On any Windows operating system with User Account Control (UAC), such as Windows Vista or Windows 7, you must either disable UAC or run 'cmd.exe' as an administrator (e.g. by right-clicking on 'cmd.exe' and selecting "Run as administrator") in order to execute the script in the procedure below. If UAC is enabled, simply logging in to Windows with an Administrator account will not be sufficient.

To set up JIRA to run as a service:

1. Open a Command Prompt.
2. Change directory ('cd') to the [JIRA application installation directory](#) and then into this directory's 'bin' subdirectory.
 - ⚠ If a directory in the path has spaces (e.g. 'C:\Program Files\..'), please convert it to its eight-character equivalent (e.g. 'C:\Progra~1\..').
3. Ensure the **JAVA_HOME** variable is set to the root of your Java platform's installation directory.
 - i** To find out the current value of the **JAVA_HOME** variable, enter `echo %JAVA_HOME%` at the command prompt.
4. Run the following command:

```
service.bat install JIRA
```

Here is a screenshot of the process:



```

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Denis>cd "c:\Program Files\atlassian-jira-enterprise-3.3-standalone"
C:\Program Files\atlassian-jira-enterprise-3.3-standalone>cd bin
C:\Program Files\atlassian-jira-enterprise-3.3-standalone\bin>echo %JAVA_HOME%
C:\j2sdk1.4.2_02
C:\Program Files\atlassian-jira-enterprise-3.3-standalone\bin>service.bat install JIRA
Installing the service 'JIRA' ...
Using CATALINA_HOME:      C:\Program Files\atlassian-jira-enterprise-3.3-standalone
Using JAVA_HOME:         C:\j2sdk1.4.2_02
Using JVM:                C:\j2sdk1.4.2_02\jre\bin\server\jvm.dll
The service 'JIRA' has been installed.
C:\Program Files\atlassian-jira-enterprise-3.3-standalone\bin>tomcat5 //US//JIRA --Startup auto
C:\Program Files\atlassian-jira-enterprise-3.3-standalone\bin>tomcat5 //US//JIRA --JvmMx 512
C:\Program Files\atlassian-jira-enterprise-3.3-standalone\bin>

```

JIRA should now be set up to run as a service.

5. In addition, to have the JIRA service start automatically when the operating system starts, run:

```
tomcat8 //US//JIRA --Startup auto
```

The JIRA service will automatically start up the next time the operating system reboots. The JIRA service can be manually started with the command '**net start JIRA**' and stopped with '**net stop JIRA**'.

To see what parameters the JIRA Core service is starting with, go to **Start -> Run** and run 're

gedt32.exe' and then:

* For Windows 32 bit edition navigate to HKEY_LOCAL_MACHINE -> SOFTWARE -> Apache Software Foundation -> Procrun 2.0 -> JIRA<time stamp>

* For Windows 64 bit edition navigate to HKEY_LOCAL_MACHINE -> SOFTWARE -> Wow6432Node -> Apache Software Foundation -> Procrun 2.0 -> JIRA<time stamp>

6. Additional JIRA setup options (optional):

- To increase the maximum memory JIRA can use (the default will already be 256MB), run:

```
tomcat8 //US//service_name --JvMx 512
```

where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

- If you are running JIRA and Confluence in the same JVM, increase the MaxPermSize size to 128 MB:

```
tomcat8 //US//service_name
++JvMOptions="-XX:MaxPermSize=128m"
```

where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

- Occasionally, it may be useful to view JIRA's Garbage Collection information. This is especially true when investigating memory issues. To turn on the Verbose GC (garbage collection) logging, execute the following command in the command prompt:

```
tomcat8 //US//service_name
++JvMOptions="-Xloggc:path\to\logs\atlassian-gc.log"
```

where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

The path (denoted by **path\to**) refers to the directory in which JIRA is currently installed. For example:

```
tomcat8 //US//service_name
++JvMOptions="-Xloggc:c:\jira\logs\atlassian-gc.log"
```

where **service_name** is the name of your JIRA service, e.g. JIRA123487934298.

See the [Tomcat documentation](#) for further service options.

Removing the JIRA service

If JIRA was installed through the Windows installer, go to the '**Control Panel**' in Windows, click '**Add or Remove Programs**' and remove JIRA. This will remove the service too.

If you installed the service manually (see above) it can be uninstalled with:

```
service.bat remove JIRA
```

Alternatively, if the above does not work, use `tomcat8 //DS//JIRA`.

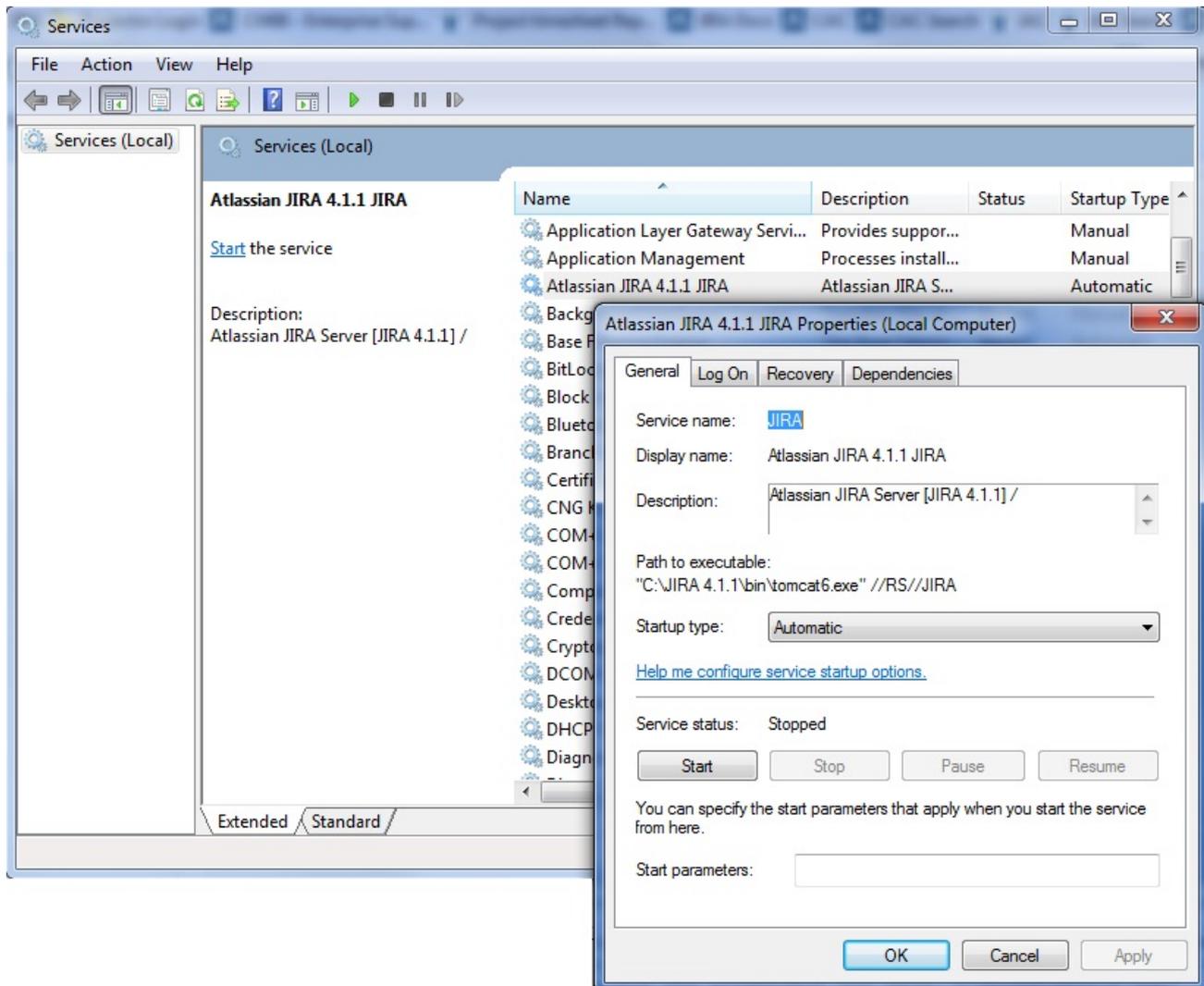
Changing the Windows user that the JIRA service uses

If you are using mapped network drives for JIRA's backup directory, attachments directory, index directory or

the %CATALINA_HOME%* directory, you need to ensure that JIRA can write to these drives. That is, these directories all need to be writeable by the user which the JIRA service is running as. This may mean that you need to change the Windows user that the JIRA server uses.

i Note that you must also specify these network drives by UNC and not letter mappings, e.g. `\\backupserv er\jira` not `z:\jira`

To change the Windows user that the JIRA service uses, navigate to the service in Windows, i.e. **'Control Panel' -> 'Administrative Tools' -> 'Services'**. Locate the 'Atlassian JIRA' service, right-click and view the 'Preferences'.



Go to the **'Log On'** tab and change the user as desired.

Specifying the startup order of multiple services

If you have services that depend on each other, it is important that they are started in the correct order. Common examples include:

- If you are running both JIRA and **Crowd**, it is important to start Crowd first, so that Crowd is running before people try to login to JIRA.
- If the database JIRA connects to is hosted on the same server as JIRA, and is started via a Windows service, the JIRA service will only start successfully if the database service has already started first.

To set up start up dependency rules, open a command prompt and enter the following command:

```
C:\Documents and Settings\Developer>sc config [JIRA service] depend=[database service]
```

Please note the space character after 'depend='.

- **[JIRA service]** is the name of the JIRA service you are running, e.g. JIRA051007111904.

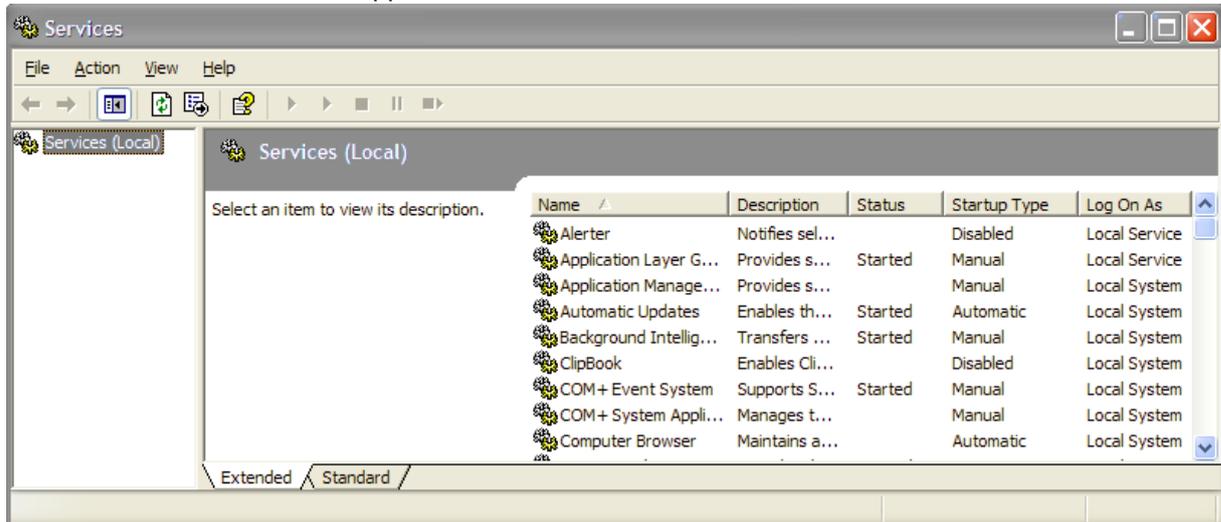
- **[database service]** is the name of the database service you are running, e.g. MSSQLSERVER.

If you wish, you can also set up dependency rules by editing the system registry. Please see <http://support.microsoft.com/kb/193888> for details on how to do this.

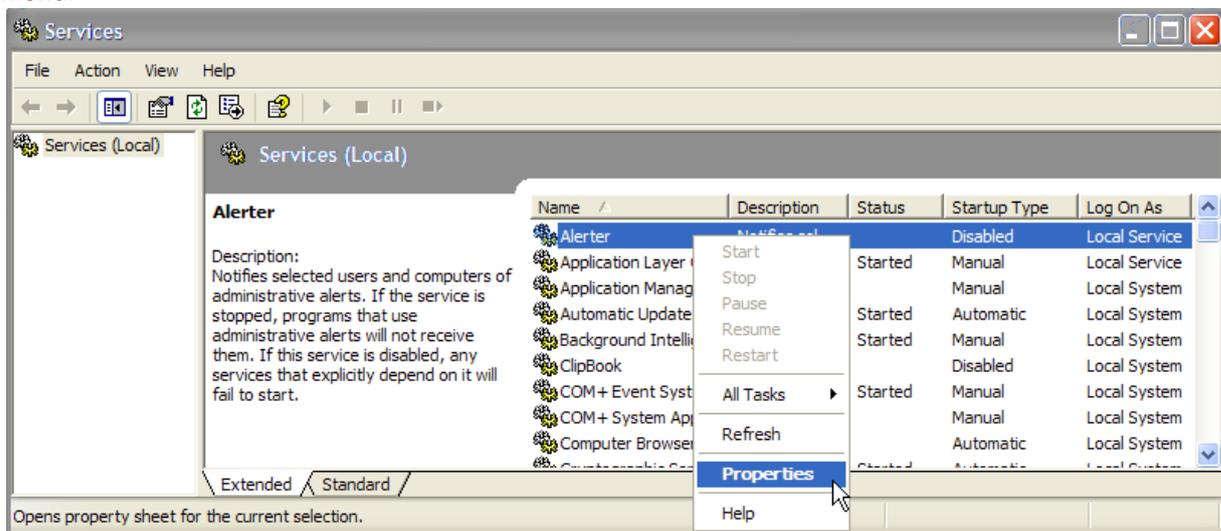
Locating the name of a service

If you do not know the exact name of your JIRA service or your database service, you can find out what they are by following the steps below:

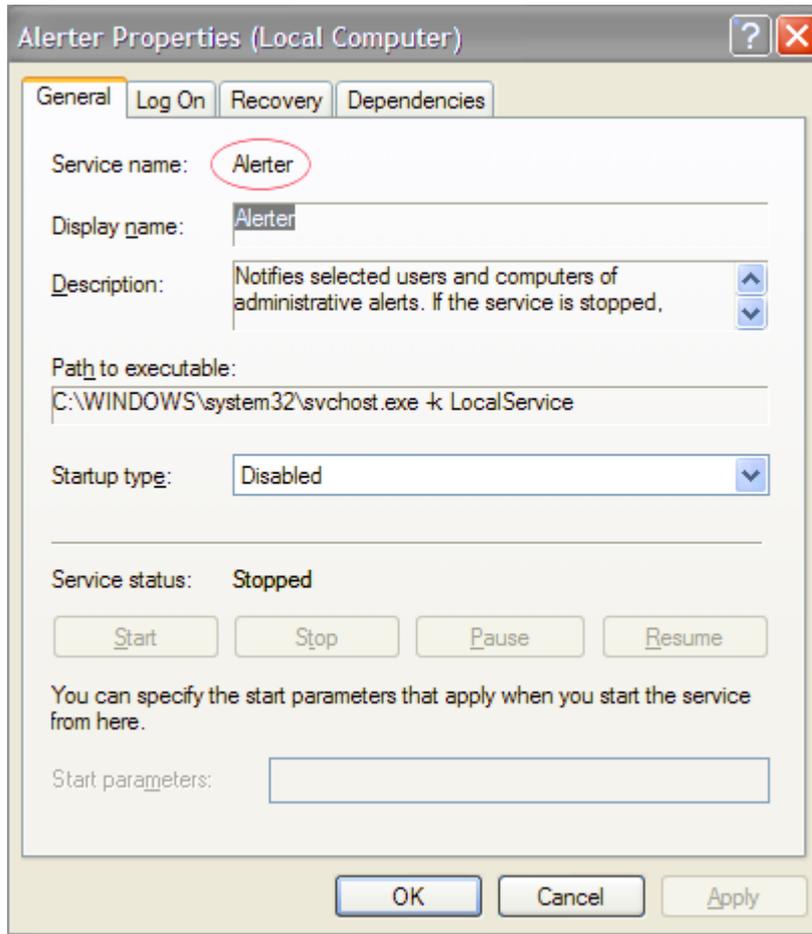
1. Navigate to '**Control Panel**' > '**Administrative Tools**' > '**Services**'.
2. The 'Services' window should appear:



3. Right-click on the service you wish to find out the name of, and select '**Properties**' from the popup menu:



4. The 'Service name' should appear in the 'General' tab:



Troubleshooting

- Java 6 is not supported by JIRA 6.0 and later. Problems may occur when trying to setup JIRA to run as a Windows service with JDK 1.6. The problem is due to failure to locate "MSVCR71.DLL", which can be found in %JAVA_HOME%/bin. There are two options to resolve this problem:
 - Add %JAVA_HOME/bin to PATH, then restart the JIRA server.
 - Copy MSVCR71.DLL to system path, C:\WINDOWS\SYSTEM32 or C:\WINNT\SYSTEM32
- Take note of the username that the service is running as, and be sure to modify the /temp and /work directories in your install directory so that this user has read and write permissions.
- You cannot run JIRA as a service on a 64-bit operating system if you require allocating more than 1.5GB of memory, due to 32-bit JDK memory limitations and 64-bit JDK/Tomcat service issues.

Tuning garbage collection (GC)

JIRA applications are robust applications that rarely require in-depth garbage collection (GC) tuning. However, on large-scale installations, GC tuning can improve the performance of JIRA applications. Analysis of GC logs can also assist in troubleshooting performance problems. This page lists basic recommendations that improve garbage collection, and is an entry point to the advanced GC Tuning Guide. The recommendations are based on Support's successful experiences with customers with large JIRA instances.

You should get yourself familiar with garbage collection if:

- JIRA has high memory or CPU consumption, or
- JIRA is performing slowly or has occasional outages

Tuning garbage collection

Checking the GC performance

The first step towards improving GC is actually measuring the GC performance of the JVM to see the size of the problem. You can do it by viewing and analyzing the GC logs. The logs indicate when the JVM is collecting garbage, how long this process takes, and how much memory has been freed. Starting from JIRA 7.4, GC logs are generated automatically, and you can find them in <installation-directory>/logs.

GCViewer is a great tool for analyzing GC logs. To enable garbage collection in earlier JIRA versions, see [Using garbage collection logs](#).

Garbage Collectors

Don't use the Concurrent Mark Sweep (CMS) Collector unless otherwise advised by Atlassian Support. It requires extensive manual tuning and testing, and is likely to result in degraded performance. Instead, we recommend that you use Garbage First Garbage Collector (G1GC), especially if your heap is larger than 6 GB.

Heap size

Start with the smallest heap possible, and increase it in 512 MB or 1 GB allotments when you experience OutOfMemory errors or run into any memory-related problems. Do not increase the heap further than required, as this will result in longer garbage collection. After garbage collection, the memory used should be 1/3 the size of the total heap, and that's your goal.

Old tuning parameters

On every full garbage collection, the JVM will resize the allocations of Eden, Survivor, etc., based on the throughput it is actually seeing. It will tune itself based on the real world data of the objects that are being created and collected. Most of the time simply allowing JVM to tune itself will give you better performance.

If you have added JVM parameters in the past and are experiencing difficulties with GC now, we'd recommend you remove all GC related parameters, unless you added them to solve a specific problem, and they did in fact solve that problem. You should also consider re-benchmarking now to ensure that they are still solving that problem, and are not causing you any other issues.

VM resources

If you run JIRA on a VM, check that it's not using the swap file. If it does, during the garbage collection, the JVM has to load the objects from the swap file into memory to clean them, and this can cause significantly longer GC pauses. Instead of using swapping, ballooning and bursting, allocate adequate memory to the VM.

Manual tuning

If you find you are still experiencing difficulties with GC after following these recommendations and you would like to see if you can tune the JVM better to improve performance, we recommend following the instructions in our [Garbage Collection \(GC\) Tuning Guide](#). This document will take you through the process of choosing performance goals (throughput/footprint/latency), and how to tune for these goals.

Getting help

How can we help you?

We have a number of [help resources](#) available. You can get your problem resolved faster by using the appropriate resource(s).

I don't know how to do something

Something isn't working

I don't like the way something works

Something else?

I don't know how to do something

1. Search [Atlassian Answers](#).
2. Raise a [support request](#)*.

Something isn't working

1. Check the JIRA knowledge base at <https://confluence.atlassian.com/display/JIRAKB>.
2. Search [Atlassian Answers](#).

3. Raise a [support request](#)*.

If you've identified a bug but don't need further assistance, raise a [bug report](#).

I don't like the way something works

Raise a [suggestion](#).

Something else?

If you need help with something else, raise a [support request](#)*.

** If you are the **JIRA system administrator**, you have a number of additional support tools available. See [Raising support requests as an administrator](#) for details.*

About our help resources

Atlassian Support

Our support team handles support requests that are raised in our [support system](#). You need to log in using your [Atlassian account](#) before you can raise support requests.

For information on our general support policies, including support availability, SLAs, bugfixes, and more, see [Atlassian Support Offerings](#). Note, you'll find anything security-related at [Security @ Atlassian](#).

Atlassian Answers

[Atlassian Answers](#) is our official application forum. Atlassian staff and Atlassian users contribute questions and answers to this site.

You may be able to find an answer immediately on Atlassian Answers, instead of having to raise a support request. This is also your best avenue for help if:

- you are using an unsupported JIRA instance or an unsupported JIRA platform,
- you are trying to perform an unsupported operation, or
- you are developing an add-on for JIRA Software.

You can also have a look at the [most popular JIRA answers](#).

JIRA knowledge base

If there are known issues with a version after it has been released, the problems will be documented as articles in our [knowledge base](#).

Atlassian issue tracker

Our official [issue tracker](#) records our backlog of bugs, suggestions, and other changes. This is open for the public to see. If you log in with your Atlassian account, you will be able to create issues, comment on issues, vote on issues, watch issues, and more.

Tip: Before you create an issue, search the existing issues to see if a similar issue has already been created.