Documentation for JIRA Administrators 7.1

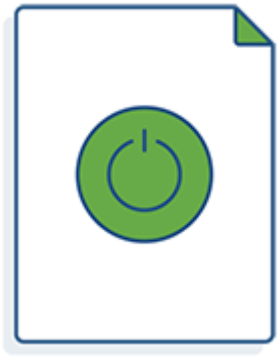
Contents

Administering JIRA applications 7.1

For JIRA application administrators

The JIRA Application Admin documentation is your one stop shop for administering all of your JIRA applications, from JIRA Core to JIRA Software and JIRA Service Desk. This space will help you get acquainted with the tools and techniques for administering your JIRA applications, projects and users.

If you're looking for information specific to a particular JIRA Application, select it below to visit the documentation.



Getting started

Installing JIRA Applications

Licensing and application access

How to

Define a project

Configure issues

Customize your JIRA Apps

Users and groups

Managing users

Managing application access

Integrations

Integrate development tools

Integrate collaboration tools

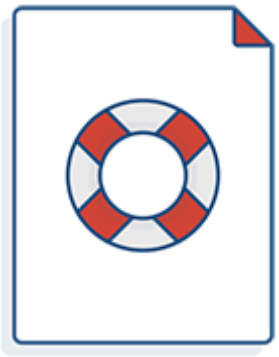
Integrate using other tools

User documentation

JIRA Software

JIRA Service Desk

JIRA Core



Support and reference info

Release notes

Knowledge base

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.

Please note that Internet Explorer 9 is **not** supported.

Installing a JIRA application

Follow the instructions for your operating system:

- Installing JIRA applications on Windows
- Installing JIRA applications on Linux
- Installing JIRA on Solaris

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

JIRA is a 'web application', meaning it runs centrally on a server, and users interact with it through web browsers from any computer.

Please read the Supported Platforms page for JIRA applications, which lists the required server and client software supported by JIRA applications

- 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 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	• 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 2GB of available RAM and a reasonably fast hard drive (7200 rpm or faster) should cater for your needs.
	• 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:
	  • 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.

	• 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:
	  • 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.

Database	Using the embedded H2 database is not supported in production. It's strongly recommended to connect them 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.

Next Steps

Installing JIRA applications

Installing Java

On this page:

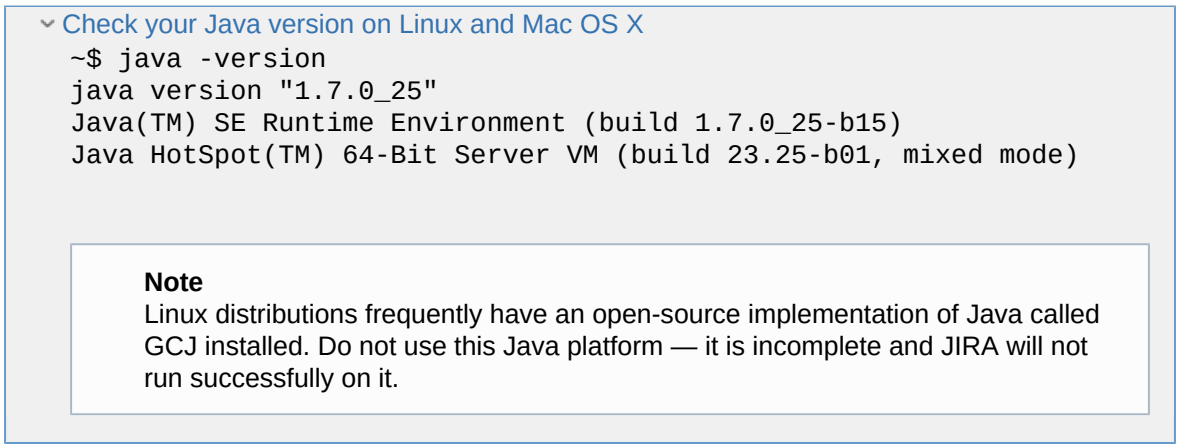
Before you begin

Please skip these instructions if you intend to use or have used the Windows Installer or Linux Installer to install JIRA, since these executable files will install and configure their own JRE to run JIRA. If you are trying to use a different Java/JRE instead of the version bundled with JIRA, please use the How to Use System JRE Instead of Embedded JRE guide.

Check out your Java version

Before you proceed to install Java check whether it's already installed by following these instructions:

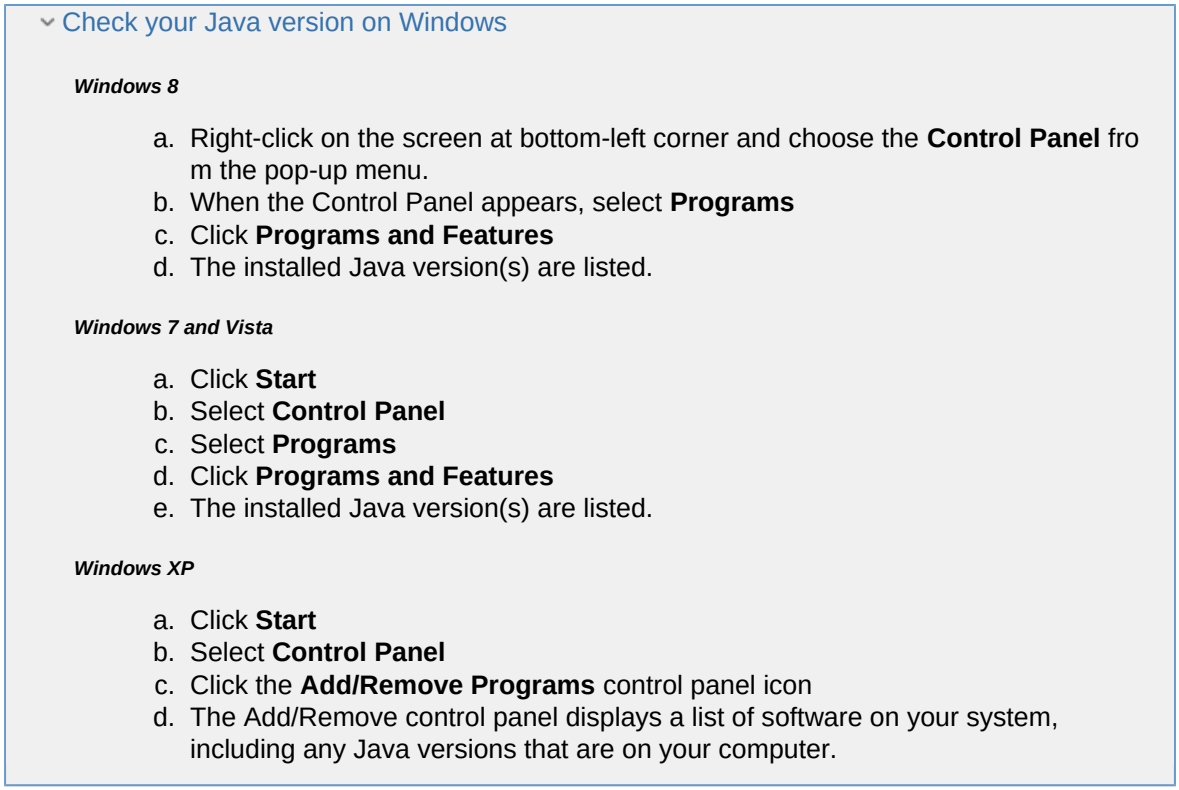
1. Check out the list of Java versions supported by JIRA here.
2. Download Oracle's JDK/JRE from Oracle's website.



Check your Java version on Linux and Mac OS X

```
~$ java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
```

Note
Linux distributions frequently have an open-source implementation of Java called GCJ installed. Do not use this Java platform — it is incomplete and JIRA will not run successfully on it.



Check your Java version on Windows

Windows 8

a. Right-click on the screen at bottom-left corner and choose the **Control Panel** fro m the pop-up menu.
b. When the Control Panel appears, select **Programs**
c. Click **Programs and Features**
d. The installed Java version(s) are listed.

Windows 7 and Vista

a. Click **Start**
b. Select **Control Panel**
c. Select **Programs**
d. Click **Programs and Features**
e. The installed Java version(s) are listed.

Windows XP

a. Click **Start**
b. Select **Control Panel**
c. Click the **Add/Remove Programs** control panel icon
d. The Add/Remove control panel displays a list of software on your system, including any Java versions that are on your computer.

Setting the JAVA_HOME

Follow these instructions if you have to install Java on your computer:

1. Once the JDK or JRE is installed, set the `JAVA_HOME` environment variable, whose value is the root directory of the JDK/JRE. Some JDK/JRE installers set this automatically.
2. Type '`echo %JAVA_HOME%`' in a Windows command prompt, or '`echo $JAVA_HOME`' in a Linux/UNIX console).

Installing JAVA_HOME on Linux

Linux-based computers

On many Linux-based computers, the `JAVA_HOME` environment variable is set in the `/etc/environment` file.

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

- `echo JAVA_HOME="path/to/JAVA_HOME" >> /etc/environment`

If, however, `JAVA_HOME` is already defined in this file, open the `/etc/environment` file in a text editor and modify its value to the appropriate `path/to/JAVA_HOME` — that is:

- `JAVA_HOME="path/to/JAVA_HOME"`

Installing JAVA_HOME on Mac OS X

Mac OS X

`On many Linux-based computers, the JAVA_HOME environment variable is set in the ~/.bash_profile file.`

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

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

If, however, `JAVA_HOME` is already defined in this file, open the ~/.bash_profile file in a text editor and modify its value to the appropriate `path/to/JAVA_HOME` — that is:

- `JAVA_HOME="path/to/JAVA_HOME"`

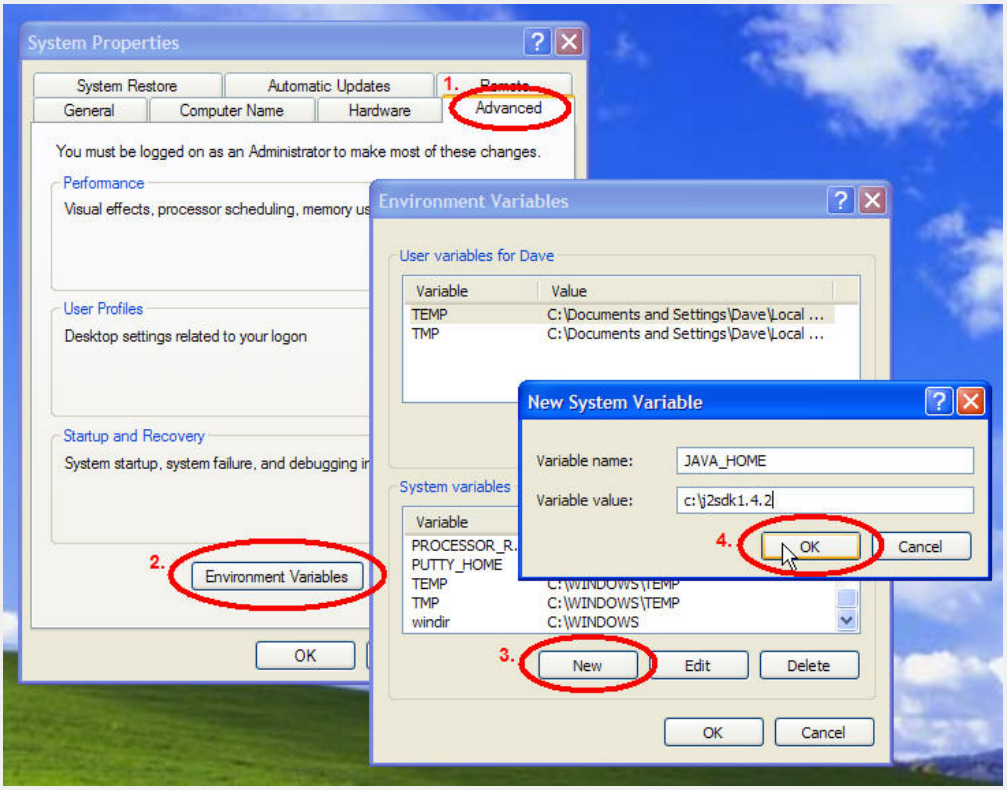
Installing JAVA_HOME on Windows

Windows-based computers

If this environment variable is not set on a Windows-based computer, you can set it in the Control Panel using the following procedure:

a. Open the Windows 'Advanced' system properties dialog box:
   - On Windows XP-based operating systems, right-click on the **My Computer** ic on on your desktop (or via the **Start** menu), select '**Properties**' and click the ' **Advanced**' tab.
   - On Windows 7-based operating systems, right-click the **Computer** icon on your desktop (or via the **Start** menu), select '**Properties**', click '**Advanced system settings**', select '**Properties**' and click the '**Advanced**' tab.

b. Click the **Environment Variables** button.
c. Click one of the **New** buttons (to define a new environment variable for your user account, or if available, system-wide).
d. Type `JAVA_HOME` as the variable name and the directory where you installed Java.



The default path for the bundled JRE with JIRA is `C:\Program Files\Atlassian\JIRA\jre`. If using a 32-bit install in 64-bit system, this would be `C:\Program Files (x86)\Atlassian\JIRA\jre`.

e. After clicking the required '**OK**' buttons to save your changes, your `JAVA_HOME` envi ronment variable should be available in a *new* command prompt window. If not or if necessary, restart your computer.

Confirming that Java Works

1. Check if Java has been correctly installed by following these instructions:

Check your Java version on Linux and Mac OS X

```
~$ java -version
java version "1.7.0_25"
Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
```

Check your Java version on Windows

Windows 8

a. Right-click on the screen at bottom-left corner and choose the **Control Panel** fro m the pop-up menu.
b. When the Control Panel appears, select **Programs**
c. Click **Programs and Features**
d. The installed Java version(s) are listed.

Windows 7 and Vista

a. Click **Start**
b. Select **Control Panel**
c. Select **Programs**
d. Click **Programs and Features**
e. The installed Java version(s) are listed.

Windows XP

a. Click **Start**
b. Select **Control Panel**
c. Click the **Add/Remove Programs** control panel icon
d. The Add/Remove control panel displays a list of software on your system, including any Java versions that are on your computer.

If you subsequently start JIRA and you receive an error like **Windows cannot find '-Xms128m'**, then you may not have correctly set `JAVA_HOME`. Please verify step 2 of the procedure above.

Next Step

Installing JIRA from an Archive File on Windows, Linux or Solaris

Supported platforms

This page lists the supported platforms for **JIRA 7.1** only. If a particular platform or a particular platform's version is not noted on this page, then *we do not support it* for JIRA 7.1.

Not using JIRA 7.1? The information below does not apply to you. See the following pages instead:

- Supported platforms for JIRA 7.0
- Supported platforms for JIRA 6.4
- Supported platforms for JIRA 6.3
- Supported platforms for JIRA 6.2
- Supported platforms for JIRA 6.1
- Supported platforms for JIRA 6.0
- Documentation for older JIRA versions

Further information:

- Please ensure you have read JIRA requirements, since not all the platforms listed below may be required for your specific JIRA setup.

Supported platforms for JIRA 7.1

	Supported platform(s)	Supported version(s)	Notes
Java platforms	Oracle JDK / JRE (formerly Sun JDK / JRE)	• 1.8	
Operating systems	Microsoft Windows		• JIRA is a pure Java-based application and should run on any supported operating system, provided that the JDK / JRE requirements are satisfied. • Please see Anti-Virus in JIRA for more information.
	Linux / Solaris		• JIRA is a pure Java-based application and should run on any supported operating system, provided that the JDK / JRE requirements are satisfied. • Atlassian only officially supports JIRA running on x86 hardware and 64-bit derivatives of x86 hardware. If you are installing JIRA from an archive, you should create a dedicated user account on the operating system to run JIRA, since JIRA runs as the user it is invoked under, and therefore can potentially be abused. • Although the JIRA Linux Installer is designed to install successfully on all 'flavors' of Linux, **we only test the JIRA Linux Installer on** CentOS Linux.If you encounter problems with the JIRA Linux Installer on your particular flavor of Linux, we recommend installing JIRA on Linux from an archive file. • NFS mounts are **not supported** due to Lucene requirements. Please see the IndexWriter docs for more information.
Virtualization	VMware		• Please read our Virtualizing JIRA (JIRA on VMware) guide for information on the required configuration of VMWare. • We are **unable to provide any support** for VMWare itself. • All of the operating systems listed in the 'Operating systems' rows above are supported for VMware.
Application servers	Apache Tomcat	• 6.0.32 • 7.0.29 • 8.0.x*	• Deploying multiple Atlassian applications in a single Tomcat container is **not supported**. • *JIRA 7.0 ships with Tomcat 8.0.17.
Mail servers			• SMTP servers must be able to support the multipart content type.
Databases (additional notes)	Oracle	• 12C	• Using Advanced Compression Option (ACO) is not supported. • Supported Oracle databases use the latest driver listed here.
	MySQL	• 5.1 - 5.4 • 5.5 • 5.6	• Neither MariaDB nor PerconaDB are supported. Both are proven to cause problems with JIRA applications. • We recommend running MySQL in strict mode. • JIRA does not support 4 byte characters, regardless of MySQL version. • If you must use 4 byte characters, we recommend you use PostgreSQL. • Supported MySQL databases use the latest JDBC driver listed here.

	PostgreSQL	• 9.4 • 9.3 • 9.2 • 9.1 • 9.0	• Supported PostgreSQL databases use the latest JDBC driver listed here.
	Microsoft SQL Server	• 2014 • 2012 • 2008 • 2008 R2	• Express Editions are **not supported**. • Supported Microsoft SQL Server databases use the latest JTDS JDBC driver listed here.
	H2	• **Supported for evaluation use only**	• JIRA ships with a built-in database (H2). While this database is suitable for evaluation purposes, we **strongly recommend** that you configure JIRA to use an external database.
Web Browsers	Chrome	• Latest stable version supported	• Minimum screen resolution of 1024 x 768 (when browsers are maximized).
	Microsoft Internet Explorer	• 10.0 • 11.0	• Minimum screen resolution of 1024 x 768 (when browsers are maximized). • 'Compatibility View' is **not supported**. • Edge support is due very soon!
	Mozilla Firefox	• Latest stable version supported	• Minimum screen resolution of 1024 x 768 (when browsers are maximized).
	Safari	• Latest stable version supported on Mac OS X only	• Minimum screen resolution of 1024 x 768 (when browsers are maximized).
	Mobile	• Mobile Safari (iOS, iPod touch and iPhone only) — Latest stable version • Android — The default browser on Android 4.0.3 (Ice Cream Sandwich)	Mobile devices are only supported on the Mobile views.

What database should I use for JIRA Server?

We've long been supporting the most commonly requested & used databases. A typical question often asked by customers is which database they should use when self-hosting their JIRA instance. Up until now we've remained database agnostic when asked this question.

Having reviewed this position recently, we believe there are benefits for customers in using PostgreSQL. In

addition to being tested to the same standard as all of our databases PostgreSQL is used;

- by Atlassian internally for all of it's JIRA instances
- for all Atlassian's Cloud JIRA instances - that's tens of thousands of instances used by our customers

This means we do more dogfooding and see more real world usage of PostrgreSQL than any of our other supported databases. That's why we now recommend you choose PostgreSQL for your JIRA server instance.

End of support announcements

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

The table below summarises the end of support announcements for **upcoming** JIRA releases.

Platform/Functionality	JIRA end of support
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)

> **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.

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`.

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`.

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`.

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).

Previous announcements

Platform/Functionality	JIRA end of support
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 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 utilise 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 at atlassian dot 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.

Installing JIRA applications on Windows

This guide describes how to install a new JIRA application installation on Windows using the automated Windows installer. If you are upgrading JIRA, please refer to the Upgrading JIRA applications guide. Before beginning your installation, you should review both the supported platforms and JIRA requirements pages for more information on what you need to run a JIRA application successfully. Note the Internet Explorer 9 is **not** supported.

You can also install JIRA from a 'zip' archive — see Installing JIRA from an archive file on Windows, Linux or Solaris for details. This is useful if you want JIRA to use a pre-existing supported Java platform, since the Windows installer installs its own JRE to run JIRA.

On this page:

Before you begin

- We strongly recommend that you install the latest production version available. If you install an older version of JIRA, it may contain security vulnerabilities that were patched or fixed in subsequent versions. If you have to install an older version, check if there are any security advisories that apply to the version.
- Note, 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.

Installing JIRA

There are two ways to install JIRA using the Windows installer:

- Using the installation wizard
- Performing an unattended installation

Using the installation wizard

Use the installation wizard if you are installing JIRA on your server for the first time or you wish to specify your installation options.

If you have previously installed JIRA using the installation wizard and wish to re-install JIRA again with the same installation options, you can re-install JIRA in 'unattended mode' without any user input required (see below for details).

1. Download and run the JIRA application 'Windows installer'

To install JIRA as a service, the Windows installer must be run using a Windows administrator account. While you can run the Windows installer with a non-administrator account, your installation options will be much more limited.

1. Download the JIRA application's **'Windows installer'** (.exe) file from the JIRA download page.
2. Run the '.exe' file to start the installation wizard.
   If a Windows 7 (or Vista) 'User Account Control' dialog box asks if you want to allow the installation wizard to make changes to your computer, specify '**Yes**'. If you do not, the installation wizard will have

restricted access to your operating system and any subsequent installation options will be limited.

3. At the 'Upgrading JIRA?' step, choose between the '**Express Install**' or '**Custom Install**' options:
    - **Express Install** — If you choose this option, JIRA will be installed with default settings, which are shown in the next step of the installation wizard. If you want to customize any of these options, click the '**Back**' button, and choose the '**Custom Install**' option instead.
    - **Custom Install** — If you choose this option, JIRA will prompt you to specify the following options, which are presented during subsequent steps of the installation wizard and pre-populated with default values:
        - The 'Destination Directory' in which to install JIRA.
        - The JIRA home directory, which must be unique for each JIRA installation.
        - The Windows 'Start' menu folder options.
        - The *TCP ports* (i.e. an HTTP and a Control port) that JIRA will run through.
        - If you are running the installer using an administrator account, you will be prompted to 'Install JIRA as a service' (recommended). You can also do this manually later, as described in Running JIRA as a Windows service.
          If you installed JIRA as a service, you must start JIRA through the Windows 'Start' menu, since JIRA will not start if you run `start-jira.bat` at the Windows Command Prompt.
4. The installation wizard will install JIRA onto your operating system and will start JIRA automatically when the wizard finishes. JIRA will also be launched automatically in your browser window if you chose this option.

Please note:

- If you chose to install JIRA as a service, the JIRA service will be run as the Windows 'SYSTEM' user account. To change this user account, see Changing the Windows user that the JIRA service uses.
- If you do not install JIRA as a service, then once started, JIRA will be run as the Windows user account under which JIRA was installed.
- If you use JIRA running on a Windows server in production, we strongly recommend creating a dedicated user account (e.g. with username 'jira') for running JIRA.
    - For more information about creating a dedicated user account and defining which directories this account should have write access to, refer to our guidelines.
      If your Windows server is operating under a Microsoft Active Directory, ask your Active Directory administrator to create a dedicated user account that you can use to run JIRA (with no prior privileges).
    - If JIRA is installed as a service, do not forget to change the user account that runs the JIRA service to your dedicated user account for running JIRA.

2. Starting JIRA

If JIRA is not already started, you can start JIRA using the appropriate Windows 'Start' menu shortcut or command prompt option.

Once JIRA is started, you can access JIRA from the appropriate Windows 'Start' menu shortcut or a browser on any computer with network access to your JIRA server.

2.1 Windows 'Start' menu shortcuts

The installer will have created the following Windows 'Start' menu shortcuts:

- **Access JIRA** — opens a web browser window to access your JIRA application.
  Your JIRA server must have been started for this shortcut to work.
- **Start JIRA server** — starts up the Apache Tomcat application server, which runs your JIRA installation so that you can access JIRA through your web browser.
- **Stop JIRA server** — stops the Apache Tomcat application server, which runs your JIRA installation. You will not be able to access JIRA through your web browser after choosing this shortcut.
- **Uninstall JIRA** — uninstalls JIRA from your Windows operating system.

2.2 Starting and stopping JIRA from a command prompt

Enter the `bin` subdirectory of your JIRA installation directory and run the appropriate file:

- `start-jira.bat` (to start JIRA)
- `stop-jira.bat` (to stop JIRA)

If you followed our guidelines for running JIRA with a dedicated user account, then to run JIRA as this user account (e.g. 'jira'), use the `runas` command to execute `start-jira.bat`. For example:

- `> runas /env /user:<DOMAIN>\jira start-jira.bat`
  (where `<DOMAIN>` is your Windows domain or computer name.)

2.3 Accessing JIRA from a browser

You can access JIRA from any computer with network access to your JIRA server by opening a supported web browser on the computer and visiting this URL:

- `http://<computer_name_or_IP_address>:<HTTP_port_number>`

where:

- `<computer_name_or_IP_address>` is the name or IP address of the computer on which JIRA is installed and
- `<HTTP_port_number>` is the HTTP port number specified when you installed JIRA (above).

If JIRA does not appear in your web browser, you may need to change the port that JIRA runs on.

3. Run the setup wizard

See Running the setup wizard.

4. Next steps

- See Configuring projects for more information on defining projects, configuring issues, permissions, and workflows, configuring project schemes and screens, etc.
- See User management for more information on managing users and groups, managing user access to JIRA applications, configuring user directories, etc.
- If you did not install JIRA as a service, you will need to start JIRA manually every time you restart your computer. To change your JIRA installation to run as a service, please see Running JIRA as a Windows service.

Performing an unattended installation

If you have previously installed JIRA using the installation wizard (above), you can use a configuration file from this JIRA installation (called `response.varfile`) to re-install 'unattended mode' without any user input required.

Installing JIRA in unattended mode saves you time if your previous JIRA installation was used for testing purposes, and you need to install JIRA on multiple server machines based on the same configuration.

Please note:

- The `response.varfile` file contains the options specified during the installation wizard steps of your previous JIRA installation. Hence, do not uninstall your previous JIRA installation just yet.
- If you intend to modify the `response.varfile` file, please ensure all directory paths specified are absolute; for example, `sys.installationDir=C\:\\Program Files\\Atlassian\\JIRA` Unattended installations will fail if any relative directory paths have been specified in this file.

Download and run the JIRA application 'Windows installer' in unattended mode

1. Download the JIRA application's **'Windows installer'** (.exe) file from the JIRA download page to a suitable location.
2. Open the Windows command prompt, and perform the remaining steps in the command prompt.

3. copy the `response.varfile` file located in the `.install4j` subdirectory of your previous JIRA installation directory, to the same location as the downloaded 'Windows installer' file.
   You can uninstall your previous JIRA installation after this step. Save your `response.varfile` if you need to install JIRA on multiple machines.
4. Change directory (`cd`) to the location of the 'Windows installer' file and run the following command:

   ```
   atlassian-jira-X.Y.exe -q -varfile response.varfile
   ```

   Where:
   - `X.Y` — refers to the version of JIRA you are about to install.
   - `-q` — instructs the installer to operate in unattended mode (i.e. 'quietly').
   - `-varfile response.varfile` — specifies the configuration file containing the configuration options used by the installer. The location and name of the configuration file should be specified after the `-varfile`option.
5. JIRA will start automatically when the silent installation finishes. Continue from step 2 Starting JIRA (a bove).

Uninstalling JIRA applications from Windows

This page describes the procedure for uninstalling JIRA, which had been installed using the Windows Installer.

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.

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 Linux

This guide describes how to install a new JIRA installation on Linux using the automated 'Linux installer'. If you are upgrading JIRA, please refer to the Upgrading JIRA applications guide. Before beginning your installation, you should review both the supported platforms and JIRA requirements pages for more information on what you need to run a JIRA application successfully. Note the Internet Explorer 9 is **not** supported.

You can also install JIRA from a 'zip' archive — see Installing JIRA from an archive file on Windows, Linux or Solaris for details. This is useful if you want JIRA to use a pre-existing supported Java platform, since the Linux installer installs its own JRE to run JIRA.

Before you begin

- We strongly recommend that you install the latest production version available. If you install an older version of JIRA, it may contain security vulnerabilities that were patched or fixed in subsequent versions. If you have to install an older version, check if there are any security advisories that apply to the version.
- Note, it is possible that any anti-virus or other Internet security tools installed on your Linux operating system 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.

Installing JIRA

There are two ways to install JIRA using the Linux installer:

- Using the console wizard
- Performing an unattended installation

Using the console wizard

Use the console wizard if you are installing JIRA on your server for the first time or you wish to specify your installation options.

If you have previously installed JIRA using the installation wizard and wish to re-install JIRA again with the same installation options, you can re-install JIRA in 'unattended mode' without any user input required (see below for details).

1. Download and install the JIRA application 'Linux installer'

If you execute the Linux installer with 'root' user privileges, the installer will create and run JIRA using a dedicated user account. You can also execute the Linux installer without 'root' user privileges, although your installation options will be much more limited, and a dedicated user account (to run JIRA) will not be created. To run JIRA as a service, the Linux installer must be executed with 'root' user privileges.

1. Download the appropriate JIRA application's **'Linux 64-bit / 32-bit installer'** (.bin) file from the JIRA download page.
   Please note:
   - To access the 32-bit installer, you may need to click the '**All JIRA download options**' link on

the JIRA download page to access the other installation packages.
- The difference between the 64-bit / 32-bit .bin installers relates to their bundled Java platforms that run JIRA. Bear in mind that a JIRA installation installed using the 64-bit installer may require additional memory (to run at a similar level of performance) to a JIRA installation installed using the 32-bit installer. This is because a 64-bit Java platform's object references are twice the size as those for a 32-bit Java platform.

2. Open a Linux console, and change directory (`cd`) to the '.bin' file's directory.
   ⚠ If the '.bin' file is not executable after downloading it, make it executable, for example:
   `chmod a+x atlassian-jira-X.Y.bin`
   (where X.Y represents your version of JIRA)
3. Execute the '.bin' file to start the console wizard.
4. When prompted to choose from '**Express Install**', '**Custom Install**', or '**Upgrade an existing JIRA installation**', choose either the '**Express Install**' or '**Custom Install**' options:
   - **Express Install**— If you choose this option, JIRA will be installed with default settings, which are shown in the next step of the console wizard.
     Please note:
     - If you are running the installer with 'root' user privileges, JIRA will be installed as a service.
     - If you want to customize any of these options:
       i. Enter '**e**' to exit the console wizard.
       ii. Execute the console wizard again (step 3 above).
       iii. Choose the '**Custom Install**' option instead.
   - **Custom Install** — If you choose this option, JIRA will prompt you to specify the following options, which are presented during subsequent steps of the console wizard and pre-populated with default values:
     - The 'Destination Directory' in which to install JIRA.
     - The JIRA home directory (which must be unique for each JIRA installation).
     - The *TCP ports* (i.e. an HTTP and a Control port) that JIRA will run through.
     - If you are running the installer with 'root' user privileges, you will be prompted to 'Run JIRA as a service' (recommended). You can also do this manually later.
5. The console wizard will install JIRA onto your operating system and will start JIRA automatically when the wizard finishes.

Please note:

- If you executed the Linux installer with 'root' user privileges, the Linux installer creates a dedicated Linux user account with username 'jira' and no password, which is used to run JIRA. This account has only:
  - Full write access to your JIRA home directory.
  - Limited write access to your JIRA installation directory.
- The bundled installer expects the 'root' user to have have the default umask (`0022` or `002`). If this is not set, it can cause problems with the install, as per
  JRA-32435 - JIRA Linux Installer does not set files with correct permissions when a non-default umask is used
  VERIFIED
  .
  Please ensure this is set prior to installation.
- If you executed the Linux installer without 'root' user privileges, be aware that JIRA can still be run with 'root' privileges. However, to protect the security of your operating system, *this is not recommended*.
- Depending on how big the JIRA instance may become, you may need to increase the maximum number of files available on the operating system. This is further covered in our Loss of Functionality due to Too Many Open Files Error KB - please review it for further information.

2. Start JIRA

If JIRA is not already started, you can start JIRA using the appropriate command at the Linux console.

Once JIRA is started, you can access JIRA from a browser on any computer with network access to your JIRA server.

2.1 Starting and stopping JIRA manually

In the Linux console, enter the `bin` subdirectory of your JIRA installation directory, and execute the appropriate file:

- `start-jira.sh` (to start JIRA)
- `stop-jira.sh` (to stop JIRA)

JIRA will be ready to access (from a browser window) when the following message appears in the application's log file:

```
*************************************************
 ... You can now access JIRA through your web browser.
*************************************************
```

To start JIRA using the service, you can execute the /etc/init.d/jira script.

2.2 Accessing JIRA from a browser

You can access JIRA from any computer with network access to your JIRA server by opening a supported web browser on the computer and visiting this URL:

- `http://<computer_name_or_IP_address>:<HTTP_port_number>`

where:

- `<computer_name_or_IP_address>` is the name or IP address of the computer on which JIRA is installed and
- `<HTTP_port_number>` is the HTTP port number specified when you installed JIRA (above).

Please Note:

- If JIRA does not appear, you may need to change the port that JIRA runs on.
- Application server logs (i.e. for Apache Tomcat) will be written to the `logs/catalina-YYYY-MM-DD.log` file within the JIRA application installation directory.

3. Run the setup wizard

See Running the setup wizard.

4. Next steps

- See Configuring projects for more information on defining projects, configuring issues, permissions, and workflows, configuring project schemes and screens, etc.
- See User management for more information on managing users and groups, managing user access to JIRA applications, configuring user directories, etc.
- If you did not install JIRA to run as a service, you will need to start JIRA manually every time you restart your computer.

Performing an unattended installation

If you have previously installed JIRA using the console wizard (above), you can use a configuration file from this JIRA installation (called `response.varfile`) to re-install JIRA in 'unattended mode' without any user input required.

Installing JIRA in unattended mode saves you time if your previous JIRA installation was used for testing purposes, and you need to install JIRA on multiple server machines based on the same configuration.

Please note:

- The `response.varfile` file contains the options specified during the installation wizard steps of your previous JIRA installation. Hence, do not uninstall your previous JIRA installation just yet.
- If you intend to modify the `response.varfile` file, please ensure all directory paths specified are absolute, for example, `sys.installationDir=/opt/atlassian/jira`

Unattended installations will fail if any relative directory paths have been specified in this file.

Download and run the JIRA application 'Linux installer' in unattended mode

1. Download the JIRA application's **'Linux installer'** (.bin) file from the JIRA download page to a suitable location.
2. Open a Linux console.
3. Copy (`cp`) the file `.install4j/response.varfile` located in your previous JIRA installation directory, to the same location as the downloaded 'Linux installer' file.
   You can uninstall your previous JIRA installation after this step. Save your `response.varfile` if you need to install JIRA on multiple machines.
4. Change directory (`cd`) to the location of the 'Linux installer' file and execute the following command:

   ```
   atlassian-jira-X.Y.bin -q -varfile response.varfile
   ```

   Where:
   - `X.Y` — refers to the version of JIRA you are about to install.
   - `-q` — instructs the installer to operate in unattended mode (i.e. 'quietly').
   - `-varfile response.varfile` — specifies the configuration file containing the configuration options used by the installer. The location and name of the configuration file should be specified after the `-varfile`option.
5. JIRA will start automatically when the silent installation finishes. Continue from step 2 Starting JIRA (a bove).

Uninstalling JIRA applications from Linux

This page describes the procedure for uninstalling JIRA, which had been installed using the Linux installer.

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`
   This command must be executed as the same user account that was used to install JIRA with the Linu x installer.
4. Follow the prompts to uninstall JIRA from your computer.

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 from an archive file on Windows, Linux or Solaris

To install JIRA on Windows from a 'zip' archive file or Linux/Solaris from a 'tar.gz' archive file, follow the instructions on this page.

Before you begin

Please ensure that you have installed Java and set JAVA_HOME. Also refer to the Supported platforms page for details about which Java (as well as

other) platforms are supported by JIRA.

⚠ Linux distributions frequently have an open-source implementation of Java called GCJ installed. Do not use this Java platform as it is incomplete, and JIRA will not run successfully on it.

On this page:

1. Download and extract the JIRA archive file

1. Download the appropriate JIRA application's archive file for your operating system ('zip' for Windows or 'tar.gz' for Linux/Solaris), from the JIRA download page.

   > After selecting the appropriate operating system tab on the 'JIRA download' page, you may need to click the '**All JIRA download options**' link to access the required installation package.

2. Extract the downloaded file. On Windows, we recommend using a file extraction tool, such as 7-Zip. On Solaris, use **GNU tar** to extract JIRA instead of the Solaris' default tar utility, as GNU tar handles long filenames better.

2. Set the JIRA home directory in JIRA

To set this, do one of the following:

- 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`.
  ℹ If you define an UNC path in Microsoft Windows, be sure to double escape the leading backslash: `\\\\machinename\\path\\to\\JIRA\\home`
  ℹ 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.

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 independent 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).

3. Create a dedicated user account on the operating system to run JIRA

ℹ This step is optional if you are evaluating JIRA, but should be mandatory for JIRA installations used in production.

A dedicated user should be created to run JIRA, as JIRA runs as the user it is invoked under, and therefore can potentially be abused. For example:

- If your operating system is *nix-based (for example, Linux or Solaris), type the following in a console:
  ```
  $ sudo /usr/sbin/useradd --create-home --comment "Account for running JIRA"
  --shell /bin/bash jira
  ```
- If your operating system is Windows:
  1. Create the dedicated user account by either:
     - Typing the following at the Windows command line:
       ```
       > net user jira mypassword /add /comment:"Account for running
       JIRA"
       ```
       (This creates a user account with user name 'jira' and password 'mypassword'. You should choose your own password.)
     - Opening the Windows 'Computer Management' console to add your 'jira' user with its own password.
  2. *(Optional)* Use the Windows 'Computer Management' console to remove the 'jira' user's membership of all unnecessary Windows groups, such as the default 'Users' group.
     ✔ If Windows is operating under a Microsoft Active Directory, ask your Active Directory administrator to create your 'jira' account (with no prior privileges).

Ensure that *only the following directories* can be written to by this dedicated user account (e.g. 'jira'):

- The following subdirectories of your JIRA installation directory for recommended JIRA distributions:
  - `logs`
  - `temp`
  - `work`
- Your JIRA home directory.

⚠ Do not make the JIRA installation directory itself writeable by the dedicated user account.

4. Start JIRA

Enter the `bin` subdirectory of your JIRA installation directory, and execute the appropriate file to start

running JIRA:

- `start-jira.sh` (on Linux/Solaris)
- `start-jira.bat` (on Windows)

To run JIRA as the dedicated user account (e.g. 'jira') created above:

- **On Windows**, use the `runas` command to run `start-jira.bat`. For example,
  `runas /env /user:<DOMAIN>\jira start-jira.bat`
  (where `<DOMAIN>` is your Windows domain or computer name.)
- **On Linux**, switch to the 'jira' account using the `su` command before running `start-jira.sh` (or use `su` to run `start-jira.sh` as the 'jira' account).

Wait until the following message appears in the application's log file:

```
*************************************************
 ... You can now access JIRA through your web browser.
*************************************************
```

You can access JIRA from any computer with network access to your JIRA server by opening a supported web browser on the computer and visiting this URL:

- `http://<computer_name_or_IP_address>:<HTTP_port_number>`

where:

- `<computer_name_or_IP_address>` is the name or IP address of the computer on which JIRA is installed and
- `<HTTP_port_number>` is the HTTP port number (8080 by default).

If JIRA does not appear in your web browser, you may need to change the port that JIRA runs on.

Logs will be written to `logs/catalina.out`.

> If something goes wrong, please verify that Java is installed correctly. If the problem persists, please contact us — we're happy to help.

5. Run the setup wizard

See Running the setup wizard.

Next steps

- To set JIRA to run automatically on restart in Windows, see Running JIRA as a Windows service.
- By default, JIRA installed from an archive uses the standard Tomcat port (i.e. 8080). If you need another application to run on that port, either now or in the future, please see Changing JIRA's TCP ports.

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.

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 2008
- Connecting JIRA applications to SQL Server 2012
- Connecting JIRA applications to SQL Server 2014

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.

On this page:

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.

1. Create and configure the PostgreSQL database

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.

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.

1. In the first screen, 'Configure Language and Database', set **Database Connection** to **My own database**.
2. Set **Database Type** to **PostgreSQL**.
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 **PostgreSQL**.
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	Description

Hostname	Located in the <url> tag (bold text in example below): <url>jdbc:postgresql://**dbserver**:5432/jiradb</url>	The name or IP address of the machine that the PostgreSQL server is installed on.
Port	Located in the <url> tag (bold text in example below): <url>jdbc:postgresql://dbserver:**5432**/jiradb</url>	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 <url> tag (bold text in example below): <url>jdbc:postgresql://dbserver:5432/**jiradb**</url>	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 <username> tag (see bold text in example below): <username>**jiradbuser**</username>	The user that JIRA uses to connect to the PostgreSQL server. You should have created this in Step 1 above.
Password	Located in the <password> tag (see bold text in example below): <password>**jiradbuser**</password>	The user's password — used to authenticate with the PostgreSQL server.
Schema	Located in the <schema-name> tag (see bold text in example below): <schema-name>**public**</schema-name>	The name of the schema that your PostgreSQL database uses. PostgreSQL 7.2 and later require a schema to be specified in the <schema-name/> element. If your PostgreSQL database uses the default 'public' schema, this should be specified in the <schema-name/> 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-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 MySQL database.

Before you begin

- Check whether your version of MySQL 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.
- If you plan to set up Confluence and JIRA on the same MySQL server, please read the Confluence MySQL setup guide, and configure your MySQL server to suit Confluence, as well as JIRA. Note that the 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:

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:

   ```
   GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,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>;
```

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 <url> tag (bold text in example below): <url>jdbc:mysql://**dbserver**:3306/jiradb?useUnicode=true&characterE torage_engine=InnoDB</url>
Port	Located in the <url> tag (bold text in example below): <url>jdbc:mysql://dbserver:**3306**/jiradb?useUnicode=true&characterE storage_engine=InnoDB</url>
Database	Located in the <url> tag (bold text in example below): <url>jdbc:mysql://dbserver:3306/**jiradb**?useUnicode=true&characterEr orage_engine=InnoDB</url>
Username	Located in the <username> tag (see bold text in example below): <username>**jiradbuser**</username>
Password	Located in the <password> tag (see bold text in example below): <password>**jiradbuser**</password>

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-que

ry>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=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&amp;characterEncoding=UTF8
&amp;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>
    <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>
    <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 and troubleshooting

- **Hostnames in permissions are compared as strings** - If you grant permission in MySQL to a hostname such as `localhost`, then you must use the *same* string for the connecting to the database from JIRA. So using 127.0.0.1 won't work, even though it resolves to the same place. This mistake produces warnings about not finding tables because the JDBC connection did not have permission to create the new tables when JIRA was set up.
- **Connection closures** — If you are using a MySQL database with any of the following, you may experience problems with your connections dropping out (see JRA-15731 for details). Please read Surviving connection closures for information on how to address this.
  - JIRA 3.13 or above,
  - version 5.5.25 or higher of Tomcat 5,
  - version 6.0.13 or higher of Tomcat 6,
- **Special characters for database password** — JIRA is not able to interpret special characters for database password.

- **Using the InnoDB storage engine** — The default storage engine used by MySQL Server versions prior to 5.5 is MyISAM. Hence, a JIRA database running on a default configuration of a MySQL Server earlier than version 5.5, could experience table creation problems (JRA-24124), which may result in data corruption in JIRA. We strongly recommend specifying the `sessionVariables=storage_engine=InnoDB` parameter in your database URL (as stated above). Doing so ensures that tables written to JIRA's MySQL database will use the InnoDB storage engine, which supports 'database transactions' required by JIRA.
- **Binary logging** — Be aware that JIRA uses the 'READ-COMMITTED' transaction isolation level with MySQL, which currently only supports row-based binary logging. If you require MySQL's binary logging features, you must configure MySQL's binary logging format to be 'row-based'. If not, you may encounter problems when creating issues in JIRA.
- **4 byte characters** — Please note that JIRA does not support using MySQL with 4 byte characters.

Connecting JIRA applications to Oracle

These instructions will help you connect JIRA to an Oracle database.

On this page:

Before you begin

- Check whether your version of Oracle 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.

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 permissions:

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

   ⚠ If the incorrect permissions are applied it's possible the JIRA instance will not work properly, as described in JIRA XML Backup and Restore error KB. Please use these permissions only.
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. 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

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.

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 **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 `<url>` tag (bold text in example below):`<url>jdbc:oracle:thin:@`**`dbserver`**`:1521:ORCL</url>`	The name or IP address of the machine that the Oracle server is installed on.
Port	Located in the `<url>` tag (bold text in example below): `<url>jdbc:oracle:thin:@dbserver:`**`1521`**`:ORCL</url>`	The TCP/IP port that the Oracle server is listening on. The default port number for Oracle is '1521'.
SID	Located in the `<url>` tag (bold text in example below):`<url>jdbc:oracle:thin:@dbserver:1521:`**`ORCL`**`</url>`	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 <username> tag (see bold text in example below): <username>**jiradbuser**</username>	The user that JIRA uses to connect to the Oracle server. You should have created this in Step 1 above.
Password	Located in the <password> tag (see bold text in example below): <password>**jiradbuser**</password>	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>
  </jdbc-datasource>
</jira-database-config>
```

3. 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 and troubleshooting

- 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. Adding this connection property may overcome these problems. Be aware that you will need to restart JIRA for this setting to take effect.

Connecting JIRA applications to SQL Server 2008

These instructions will help you connect JIRA to a Microsoft SQL Server 2008 database.

Before you begin

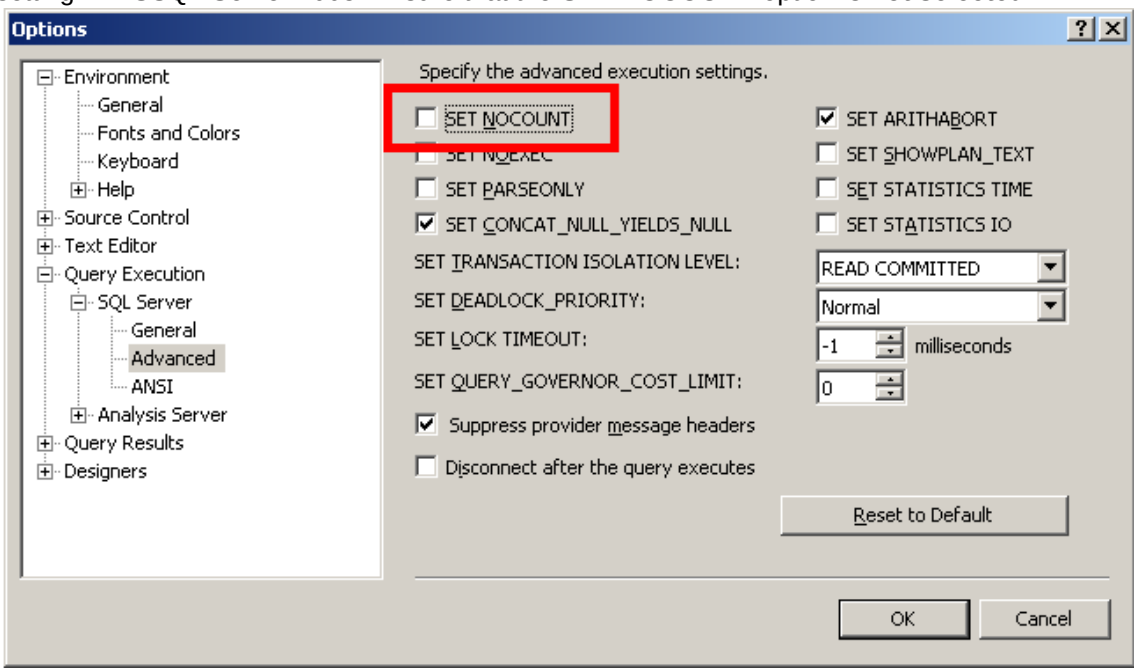
- Check whether your version of SQL Server is supported. See Supported platforms.
  Note, SQL Server Express is *not supported*, however, it is possible to set up JIRA to work with this database.
- If you are migrating JIRA to another server, create an export of your data as an XML backup. You will then 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:

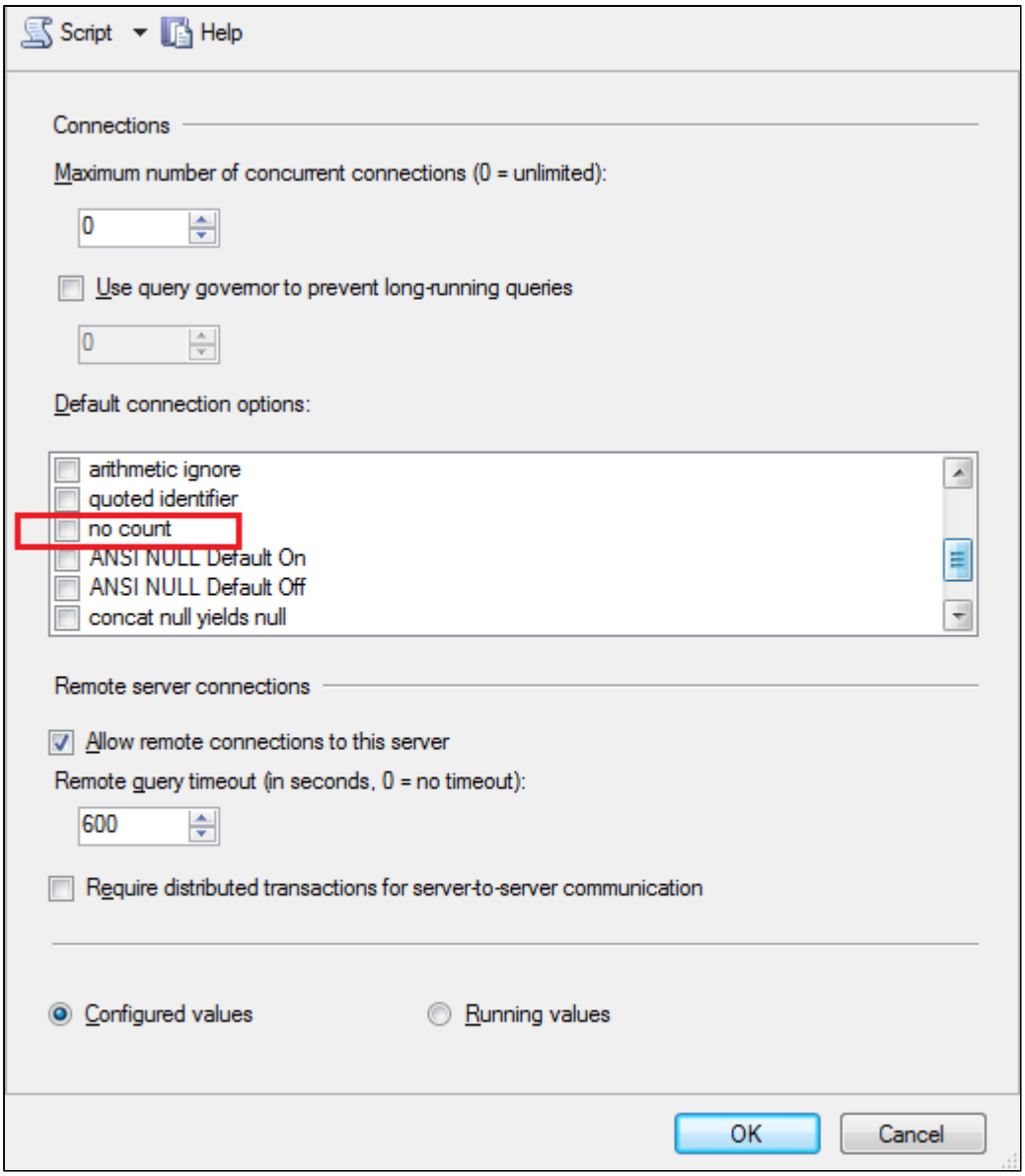
1. Create and Configure the SQL Server Database

1. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
   Remember your database name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - Collation type must be **case-insensitive**. 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`**).
   Remember your database user name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - This database user should *not* be the database owner, but *should* be in the `db_owner` role.
3. Create an empty 'schema' in the database (e.g. **`jiraschema`**) for the JIRA tables.
   Remember this database schema name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - A 'schema' in SQL Server 2008 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 2008 schemas in the relevant Microsoft documentation.
4. Ensure that the database user has permission to connect to the database, and create and populate tables in the newly-created schema.
5. Ensure that TCP/IP is enabled on SQL Server and listening on the correct port (which is 1433 for a default SQL Server installation).
   Remember this port number, as it will be used to configure JIRA's connection to this database in subsequent steps. JIRA does not support dynamic port assignment.
   - Read the Microsoft documentation for information on how to enable a network protocol (TCP/IP) and how to configure SQL server to listen on a specific port.
6. Ensure that SQL Server is operating in the appropriate authentication mode.
   - 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. To turn off SET NOCOUNT:
   - Open **SQL Server Management Studio** and navigate to **Tools > Options > Query Execution**

> SQL Server > Advanced. The following screenshot displays the configuration panel for this setting in MSSQL Server 2008. Ensure that the **SET NOCOUNT** option is **not selected:**



- You will also need to access the **Server > Properties > Connections > Default Connections** properties box and clear the **no count** option.



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

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

2. Configure Your JIRA Server to Connect to Your SQL Server 2008 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 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**

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.

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.

- **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

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the `<url>` tag (bold text in example below):`<url>jdbc:jtds:sqlserver://`**dbserver**`:1433/jiradb</url>`	The name or IP address of the machine that the SQL Server server is installed on.
Port	Located in the `<url>` tag (bold text in example below): `<url>jdbc:jtds:sqlserver://dbserver:`**1433**`/jiradb</url>`	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 `<url>` tag (bold text in example below): `<url>jdbc:jtds:sqlserver://dbserver:1433/`**jiradb**`</url>`	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 `<username>` tag (see bold text in example below): `<username>` **jiradbuser** `</username>`	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 `<password>` tag (see bold text in example below): `<password>` **jiradbuser** `</password>`	The user's password — used to authenticate with the SQL Server server.
Schema	Located in the `<schema-name>` tag (see bold text in example below): `<schema-name>` **dbo** `</schema-name>`	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:jtds:sqlserver://dbserver:1433/jiradb</url>
  <driver-class>net.sourceforge.jtds.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>
</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!

Congratulations, you now have JIRA connected to your SQL Server database.

Connecting JIRA applications to SQL Server 2012

These instructions will help you connect JIRA to a Microsoft SQL Server 2012 database.

Before you begin

- Check whether your version of SQL Server is supported. See Supported platforms.
  Note, SQL Server Express is *not supported*, however, it is possible to set up JIRA to work with this database.
- If you are 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.
- Shut down JIRA before you begin, unless you are running the Setup Wizard.

On this page:

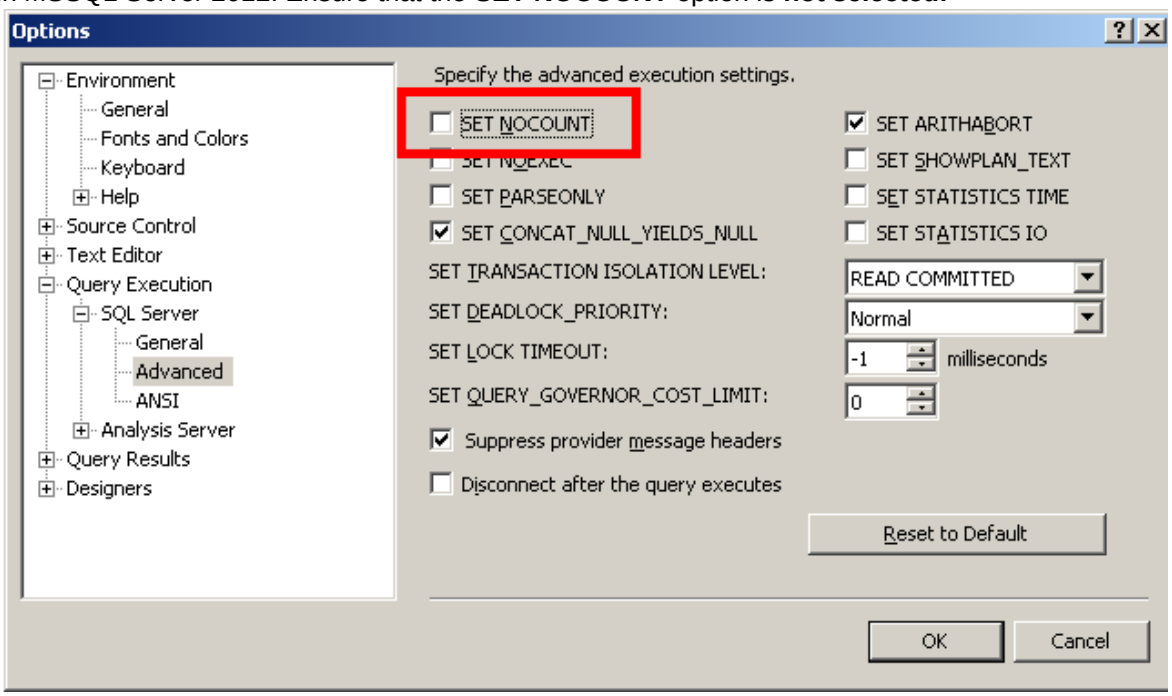
1. Create and Configure the SQL Server Database

1. Create a database for JIRA to store issues in (e.g. **`jiradb`**).

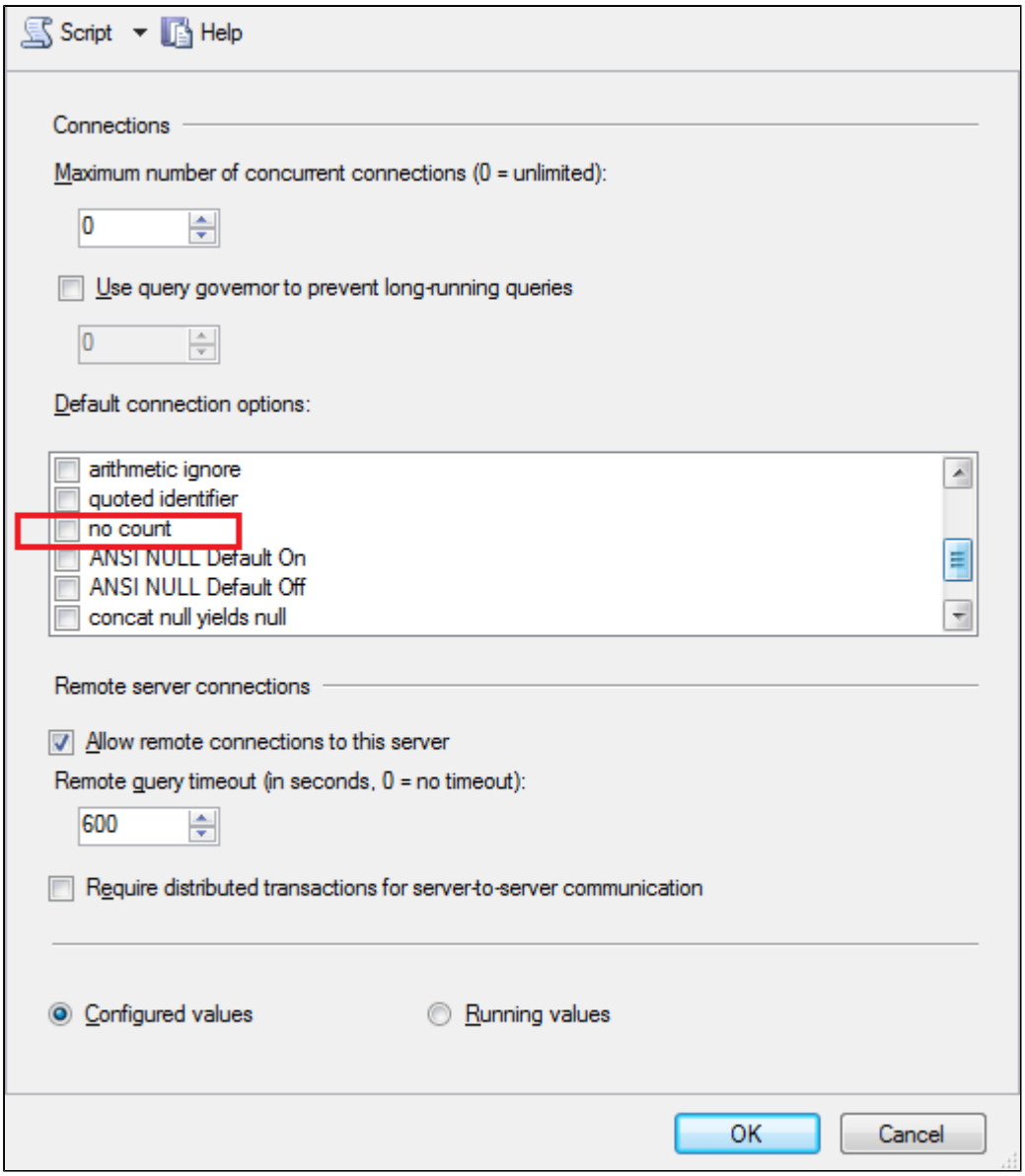
Remember your database name, as it will be used to configure JIRA's connection to this database in subsequent steps.

- Collation type must be **case-insensitive**. 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`**).
   Remember your database user name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - This database user should *not* be the database owner, but *should* be in the `db_owner` role.
3. Create an empty 'schema' in the database (e.g. **`jiraschema`**) for the JIRA tables.
   Remember this database schema name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - A 'schema' in SQL Server 2012 is a distinct namespace used to contain objects and is *different* fro m 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. Ensure that the database user has permission to connect to the database, and create and populate tables in the newly-created schema.
5. Ensure that TCP/IP is enabled on SQL Server and listening on the correct port (which is 1433 for a default SQL Server installation).
   Remember this port number, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - Read the Microsoft documentation for information on how to enable a network protocol (TCP/IP) and how to configure SQL server to listen on a specific port.
6. Ensure that SQL Server is operating in the appropriate authentication mode.
   - 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. To turn off SET NOCOUNT:
   - Open **SQL Server Management Studio** and navigate to **Tools > Options > Query Execution > SQL Server > Advanced**. The following screenshot displays the configuration panel for this setting in MSSQL Server 2012. Ensure that the **SET NOCOUNT** option is **not selected:**



   - You will also need to access the **Server > Properties > Connections > Default Connections** pro perties box and clear the **no count** option.



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

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

2. Configure Your JIRA Server to Connect to Your SQL Server 2012 Database

There are three 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.
- **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.

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 **SQL Server**.
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 **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

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the `<url>` tag (bold text in example below):`<url>jdbc:jtds:sqlserver://`**dbserver**`:1433/jiradb</url>`	The name or IP address of the machine that the SQL Server server is installed on.
Port	Located in the `<url>` tag (bold text in example below): `<url>jdbc:jtds:sqlserver://dbserver:`**1433**`/jiradb</url>`	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 `<url>` tag (bold text in example below): `<url>jdbc:jtds:sqlserver://dbserver:1433/`**jiradb**`</url>`	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 `<username>` tag (see bold text in example below): `<username>` **jiradbuser** `</username>`	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 `<password>` tag (see bold text in example below): `<password>` **jiradbuser** `</password>`	The user's password — used to authenticate with the SQL Server server.

Schema	Located in the `<schema-name>` tag (see bold text in example below): `<schema-name>` **dbo** `</schema-name>`	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:jtds:sqlserver://dbserver:1433/jiradb</url>
  <driver-class>net.sourceforge.jtds.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>
</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!

Congratulations, you now have JIRA connected to your SQL Server database.

Connecting JIRA applications to SQL Server 2014

These instructions will help you connect JIRA to a Microsoft SQL Server 2014 database.

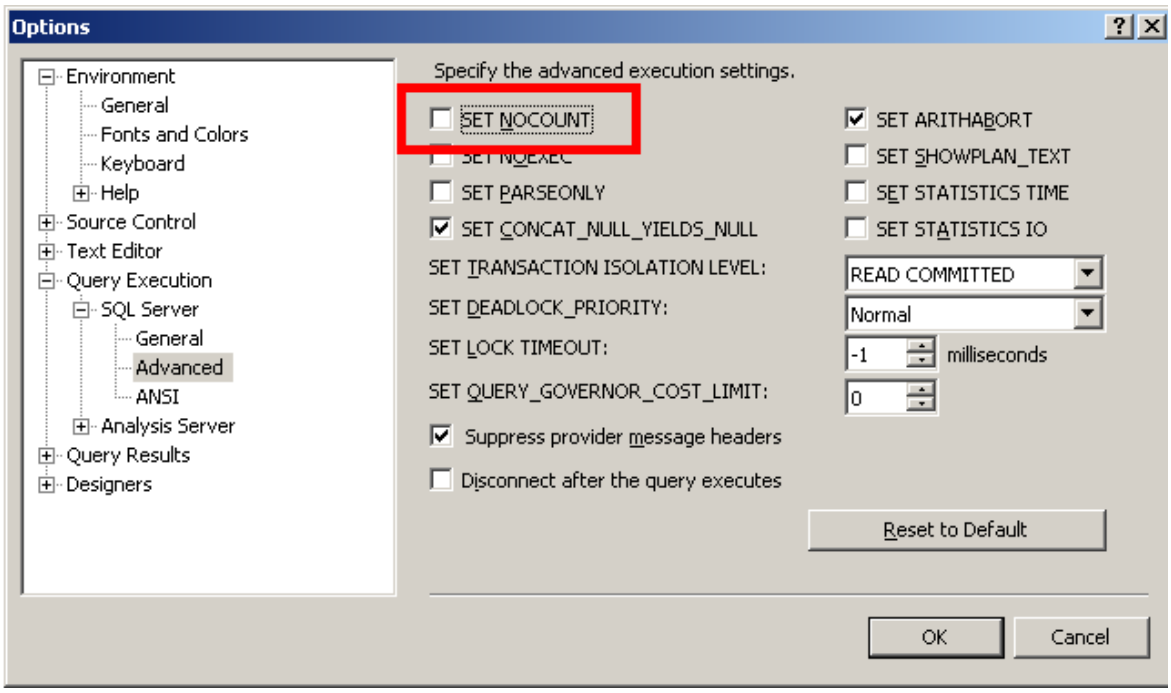
Before you begin

- Check whether your version of SQL Server is supported. See Supported platforms.
  Note, SQL Server Express is *not supported*, however, it is possible to set up JIRA to work with this database.
- If you are 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.
- Shut down JIRA before you begin, unless you are running the Setup Wizard.

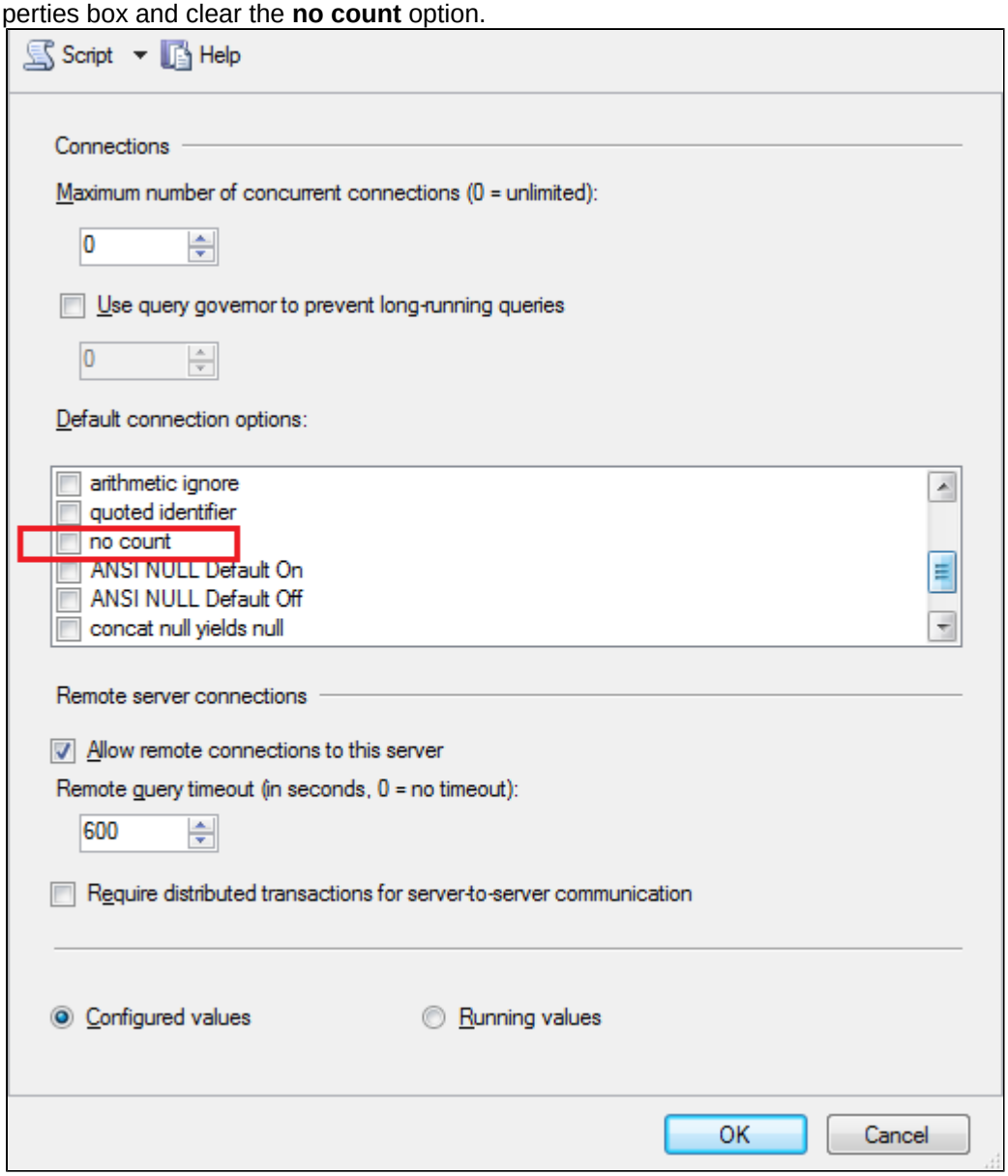
On this page:

1. Create and Configure the SQL Server Database

1. Create a database for JIRA to store issues in (e.g. **`jiradb`**).
   Remember your database name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - Collation type must be **case-insensitive**. 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`**).
   Remember your database user name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - This database user should *not* be the database owner, but *should* be in the `db_owner` role.
3. Create an empty 'schema' in the database (e.g. **`jiraschema`**) for the JIRA tables.
   Remember this database schema name, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - A 'schema' in SQL Server 2014 is a distinct namespace used to contain objects and is *different* fro m 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 2014 schemas in the relevant Microsoft documentation.
4. Ensure that the database user has permission to connect to the database, and create and populate tables in the newly-created schema.
5. Ensure that TCP/IP is enabled on SQL Server and listening on the correct port (which is 1433 for a default SQL Server installation).
   Remember this port number, as it will be used to configure JIRA's connection to this database in subsequent steps.
   - Read the Microsoft documentation for information on how to enable a network protocol (TCP/IP) and how to configure SQL server to listen on a specific port.
6. Ensure that SQL Server is operating in the appropriate authentication mode.
   - 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. To turn off SET NOCOUNT:
   - Open **SQL Server Management Studio** and navigate to **Tools > Options > Query Execution > SQL Server > Advanced**. The following screenshot displays the configuration panel for this setting in MSSQL Server 2014. Ensure that the **SET NOCOUNT** option is **not selected:**



- You will also need to access the **Server > Properties > Connections > Default Connections** properties box and clear the **no count** option.



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

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

2. Configure Your JIRA Server to Connect to Your SQL Server 2014 Database

There are three 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.
- **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.

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 **SQL Server**.
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 **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

Setup Wizard / Configuration Tool	dbconfig.xml	Description
Hostname	Located in the <url> tag (bold text in example below):<url>jdbc:jtds:sqlserver://**dbserver**:1433/jiradb</url>	The name or IP address of the machine that the SQL Server server is installed on.
Port	Located in the <url> tag (bold text in example below): <url>jdbc:jtds:sqlserver://dbserver:**1433**/jiradb</url>	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 <url> tag (bold text in example below): <url>jdbc:jtds:sqlserver://dbserver:1433/**jiradb**</url>	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 <username> tag (see bold text in example below): <username> **jiradbuser** </username>	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 <password> tag (see bold text in example below): <password> **jiradbuser** </password>	The user's password — used to authenticate with the SQL Server server.
Schema	Located in the <schema-name> tag (see bold text in example below): <schema-name> **dbo** </schema-name>	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:jtds:sqlserver://dbserver:1433/jiradb</url>
  <driver-class>net.sourceforge.jtds.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>
</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!

Congratulations, you now have JIRA connected to your SQL Server database.

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

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.

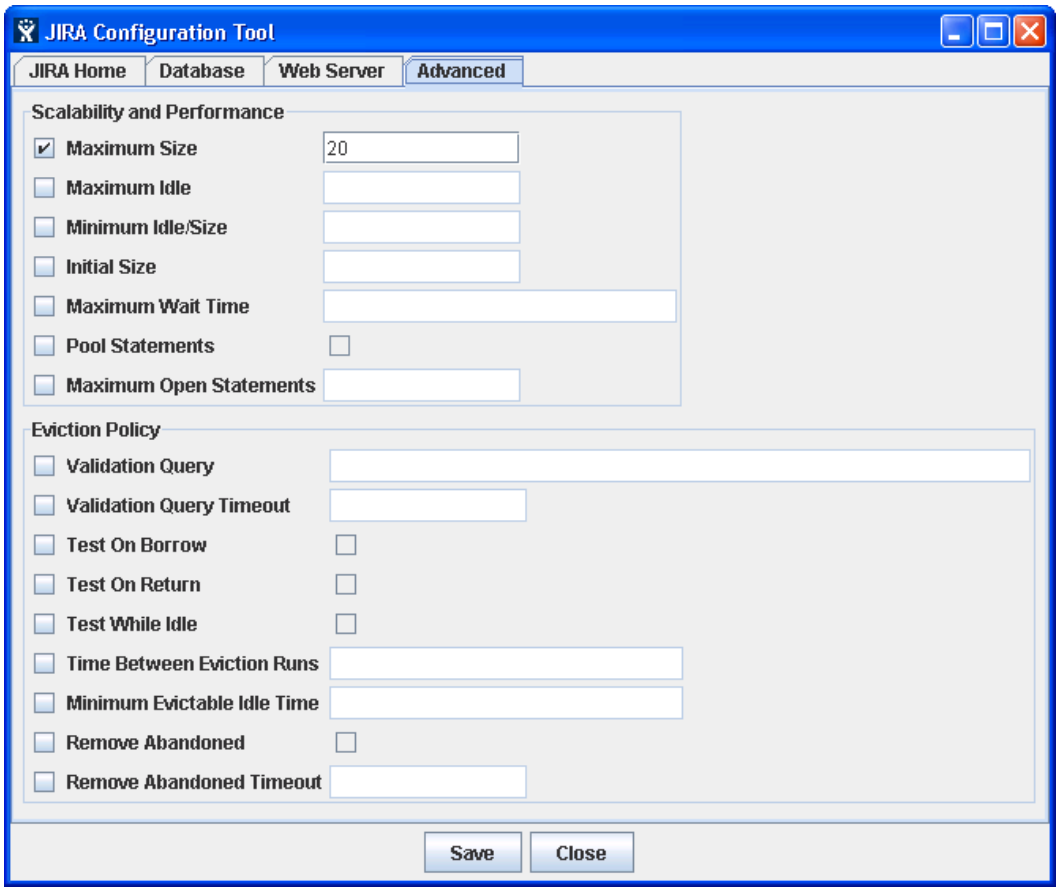
Conversely, if the frequency of JIRA's database access requests begin to drop below the number of available database connections in the pool, connections can be automatically closed to release resources back to the system.

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.
              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.
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 `db config.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 v

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 **um 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 **um 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 nc specifiec `onfig.x`

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 (whilst 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 required.				
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 **JRA-44908** - DBPC configuration pool-prepared-statements leads to Statement Leak RESOLVED	false (when nc specifiec `onfig.x`
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 nc specifiec `onfig.x`
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 f or more information.	select 1 (for MyS (otherwis specifiec `onfig.x`

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.	The length of time should be quite short as the validation query should be designed to do a minimum amount of work. 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.	3 (for MyS (otherwis specifiec `onfig.x`
Test On Borrow	`pool-test-on-borrow`	Tests if the database connection is valid when it is borrowed from the database connection pool by JIRA. If the database connection is broken, it is removed from the pool.	This value should always be 'false' as JIRA borrows a connection for each database operation. 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.	false (when nc specifiec `onfig.x`

Test On Return	`pool-test-on-return`	Tests if the database connection is valid when it is returned to the database connection pool by JIRA. If the database connection is broken, it is removed from the pool.	This value should always be 'false' as JIRA returns borrowed connections for each database operation.	false (when nc specifiec `onfig.x`
Test While Idle	`pool-test-while-idle`	Periodically tests if the database connection is valid when it is idle. If the database connection is broken, it is removed from the pool.	This should be set to 'true' for MySQL. 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. **Test While Idle** only needs to be specified if you have specified a **Validatio n Query** above.	true (for MyS false (when nc specifiec `onfig.x`

Time Between Eviction Runs	`time-between-eviction-runs-millis`	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. The eviction thread will remove idle database connections when the number of idle connections exceeds **Minimum Idle/Size** (above).	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). If you continue to have problems with database connections closing, try setting this option to a lower value.	300000 (for MyS 5000 (for HSQ (otherwis specifiec `onfig.x`
Minimum Evictable Idle Time	`min-evictable-idle-time-millis`	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).		60000 (for MyS 4000 (for HSQ (otherwis specifiec `onfig.x`

Remove Abandoned	`pool-remove-abandoned`	Flag to remove abandoned database connections if they exceed the **Removed Abandoned Timeout** (be low). 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.	This value should be set to 'true'. This will allow the pool to recover any abandoned connections and prevent this affecting system performance.	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&amp;characterEnco
ding=UTF8&amp;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-mi
llis>

    <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>
    <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 `dbconfig.xml`
Test While Idle	`pool-test-while-idle`
Time Between Eviction Runs	`time-between-eviction-runs-millis`
Minimum Evictable Idle Time	`min-evictable-idle-time-millis`

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	`select 1`	`3`
Microsoft SQL Server	`select 1`	`N/A`
Oracle	`select 1 from dual`	`N/A`
PostgreSQL	`select version();`	`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.

     ℹ **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.

Installing JIRA Data Center

This page:

Before you start:

Before you install JIRA Data Center, please review this pre-requisite information:

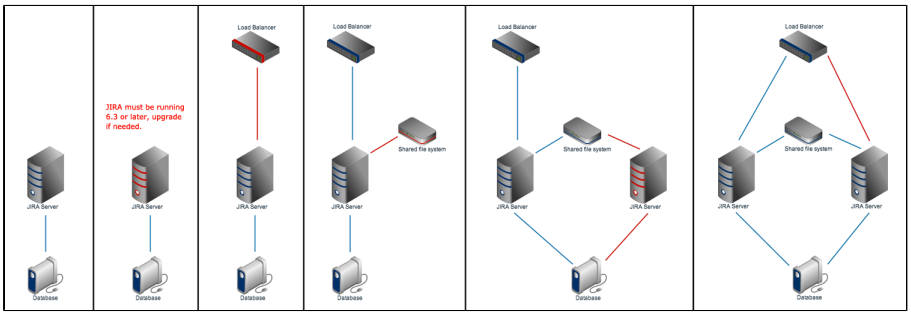
- Understand how JIRA Data Center works.
- Understand the node requirements:
    - Each JIRA node must run on its own machine (physical or virtual), with a separate machine for the shared services. The shared services machine must be accessible by each node.
    - Normal JIRA supported platforms and requirements apply to each node.
    - Each node does not need to be identical, but for consistent performance, we recommend they are as close as possible.
    - Nodes must run the exact same JIRA version and must be located in the same data center.
    - Nodes must be configured with the same timezone and keep the current time synchronized. Using ntpd or some similar service is a good way to arrange this.
- Install and configure a load balancer of your choice:
    - The load balancer must support "cookie based session affinity", (also known as "sticky sessions").

You can optionally cluster the load balancer, database, and shared file systems.

After you install JIRA Data Center or add a new node to your environment, use the health check tools to check that your instance is configured and operating correctly.

Installing JIRA Data Center

This illustration shows the general method of installing a JIRA clustered instance:



This install guide assumes that you already have a JIRA instance, already have a load balancer, and are able to set up a network file share system.

> Before upgrading from an earlier version of JIRA, back up your data. Refer to Automating JIRA backups.

1. Upgrade your JIRA instance to 7.0 or later

See JIRA applications installation guide.

> You must purchase a JIRA Data Center license to use the clustering functionality of JIRA Data Center. Please contact our sales team for information about purchasing a JIRA Data Center license.

2. Set up the JIRA file storage location on shared storage

In this step, you need to set up a shared home directory that is writable by the JIRA instance and any future nodes.

There are multiple ways to do this, but the simplest is to use an NFS share. The mechanics of setting one is unique from installation to installation, and is outside the scope of this document.

Assuming that the final mount point for this shared storage location is */data/jira/sharedhome* :

- Ensure that directory can be read and written by other potential nodes
- Copy the following directories into /data/jira/sharedhome: (some of them may be empty)
  - *data*
  - *plugins*
  - *logos*
  - *import*
  - *export*

```
$ cp -R /path/to/jira-local-home/{data,plugins,logos,import,export}
/data/jira/sharedhome
```

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

Set up the following on your existing JIRA instance:

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

Expand for example

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
```

See `Cluster.properties file parameters` for more information.

- If using the Apache load balancer, set the Apache node name by appending the following setting to the same variable (replacing `node1` with the node name used in the load balancer configuration):
  - `-DjvmRoute=node1`

4. Add the first node to your load balancer

JIRA Data Center relies on a load balancer to balance traffic between the nodes. Many larger installations of JIRA already have a reverse proxy configured, and many reverse proxies have the ability to perform load balancing as well. We've provided a sample Apache httpd configuration to serve as an example, but please check with your proxy vendor for specific information.

Sample httpd configuration with mod_balancer

```
<VirtualHost *:80>
        ProxyRequests off

        ServerName MyCompanyServer

        <Proxy balancer://jiracluster>
                # JIRA node 1
                BalancerMember http://jira1.internal.atlassian.com:8080
route=node1
                # JIRA node 2  Commented pending node installation
                # BalancerMember http://jira2.internal.atlassian.com:8080
route=node2

                # Security "we aren't blocking anyone but this the place to
make those changes
                Order Deny,Allow
                Deny from none
                Allow from all

                # Load Balancer Settings
                # We are not really balancing anything in this setup, but need
to configure this
                ProxySet lbmethod=byrequests
                ProxySet stickysession=JSESSIONID
        </Proxy>

        # Here's how to enable the load balancer's management UI if desired
        <Location /balancer-manager>
                SetHandler balancer-manager

                # You SHOULD CHANGE THIS to only allow trusted ips to use the
manager
                Order deny,allow
                Allow from all
        </Location>

        # Don't reverse-proxy requests to the management UI
        ProxyPass /balancer-manager !
        # Reverse proxy all other requests to the JIRA cluster
        ProxyPass / balancer://jiracluster/
        ProxyPreserveHost on
</VirtualHost>
```

After adding JIRA to the load balancer, ensure that basic functionality is working after restarting the JIRA instance by navigating to the instance, logging in, and noting any broken links or malfunctioning JIRA functionality.

Be sure to check that the base server URL is configured properly (to the load balancer public URL).

5. Add a new JIRA node to the cluster

1. Copy the JIRA installation directory to a new host. Atlassian recommends that your configuration deviates from the first installation as little as possible to ease the burden of documentation and deployment (e.g. Installation paths, users, file permissions, etc).
2. Ensure that the new host can access the shared home directory (e.g. ensure that you can read the contents of the shared JIRA directory and have write access to it)
3. Copy the local home directory from the first node to this new node.
4. Alter the cluster.properties file to reference the new node id. All node ids must be unique among nodes.

5. Start the new node and monitor for startup problems.
6. Ensure that issue creation, search, attachments, and customizations work as expected.

6. Connect this new node to the load balancer

Verify that the new node is in the cluster and receiving requests by checking the logs on each node to ensure both are receiving traffic and also check that updates done on one node are visible on the other.

Repeat steps 5 and 6 for each node.

Cluster.properties file parameters

You can set the following parameters in the `cluster.properties` file:

Parameter	Required	Description/value
`jira.node.id`	Yes	This unique ID must match the username and the `BalancerMember` entry in the Apache config
`jira.shared.home`	Yes	The location of the shared home directory for all JIRA nodes
`ehcache.peer.discovery`	No	Describes how nodes find each other: `default` - JIRA will automatically discover nodes. Recommended `automatic` - Will use EhCache's multicast discovery. This is the historical default method used by ehCache, but can be problematic for customers to configure and is no longer recommended by Atlassian for use with JIRA clustering
`ehcache.listener.hostName`	No	The hostname of the current node for cache communication. JIRA Data Center will resolve this this internally if the parameter isn't set. If you have problems resolving the hostname of the network you can set this parameter.
`ehcache.listener.port`	No	The port the node is going to be listening to (default = 40001) if multiple nodes are on the same host or this port is not available, you might need to set this manually.
`ehcache.listener.socketTimeoutMillis`	No	By default this is set to the Ehcache default

If you set `ehcache.peer.discovery` = `automatic` then you need to set the following parameters:

- `ehcache.multicast.address`
- `ehcache.multicast.port`
- `ehcache.multicast.timeToLive`
- `ehcache.multicast.hostName`

Refer to the Ehcache documentation for more information on these parameters.

Running the setup wizard

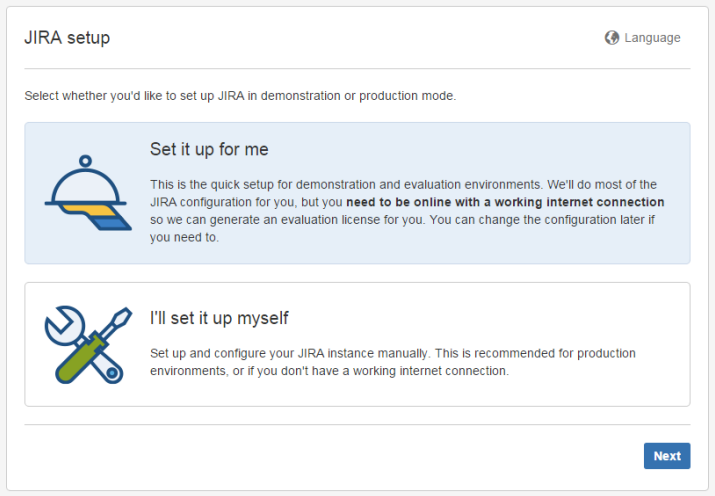
The JIRA setup wizard allows you to either set up a JIRA application for evaluation and demonstration purposes, or for production and testing.

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`.

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.*

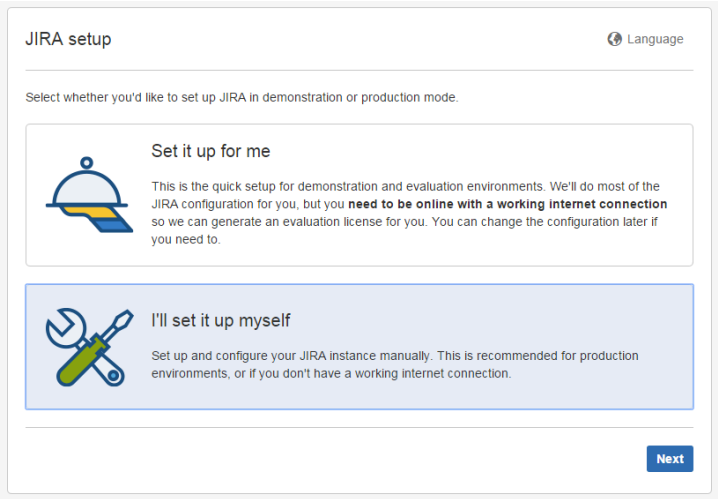
Evaluation and demonstration setup



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



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.	• 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	1. Choose a database. See our list of supported databases first. 2. 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.	• 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.	• 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.	• 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.	• 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.

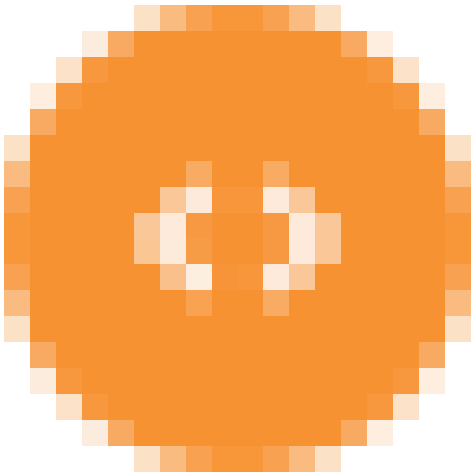
JIRA applications and project types overview

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.

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.

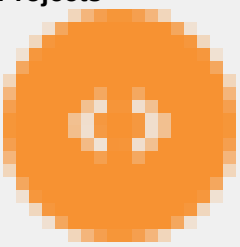
Application	Project type	Application-specific feature set
JIRA Core	 Business projects	• Available to all licensed users of JIRA
JIRA Software	 Software projects	• Integration with development tools • Agile boards • Release hub for software version release
JIRA Service Desk	 Service Desk projects	• Service Level Agreements (SLAs) • A customisable 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.

Note that 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.

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
Business Projects 	Project level	View	✓	✓	✓
	Issue level	Create	✓	✓	✓
		View	✓	✓	✓
		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	✗	✗	✓

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

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-user**
JIRA Software	**jira-software-user**
JIRA Service Desk	**jira-servicedesk-agent**

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.

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	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: • Run arbitrary Java when a certain transition occurs, via post-functions. • Limit visibility of transitions to certain users, via conditions. • Validate input on transition screens (eg. in comments), via validators. See the Working with workflows for details on workflow post-functions, conditions, and validators. Once written, these can be packaged as plugins and reused.
Issues and projects	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. 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.
Listeners (Note this is not configurable in JIRA Cloud applications)	JIRA has a complete event subsystem, which fires events whenever anything happens. For example, an `ISSUE_CREATED` event is fired whenever an issue is created. A listener is just a class that implements a `JiraListener` interface, and is called whenever events occur in JIRA. Using those events, you can then perform any action you want. For example the email sent by JIRA is driven by the `MailListener`. This is useful when you want to drive or affect external systems from events, which occur within JIRA — usually used to *push* data into outside systems.For more information, read the listeners documentation.

Services	Services are classes that implement the `JiraService` interface.When installed, you specify an update period, and JIRA will call the `run()` method of your service periodically. A sample service is `POPCommentService`. This service checks a particular POP mailbox periodically, and if it finds messages, tries to extract an issue key from the subject. If the subject contains a key, the body of the mail is added as a comment to the message. Services are useful when you want to periodically *pull* data into JIRA from outside systems. For more information, see the services guide.

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.

On this page:

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.

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 apps from Cloud.

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.

On this page:

Integration features

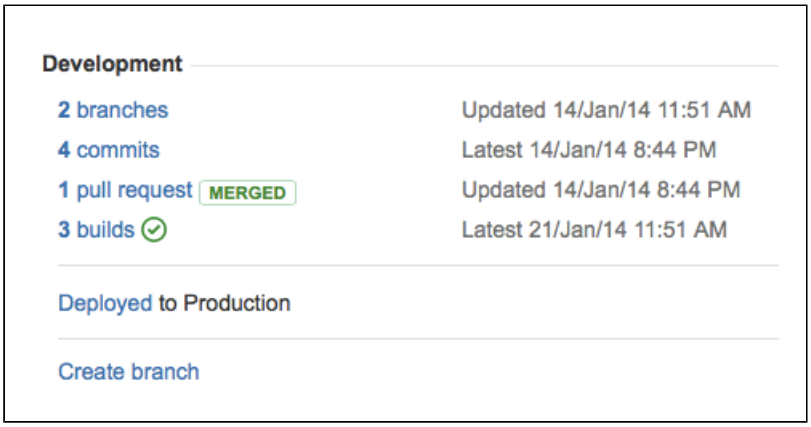
These are the features that become available when you connect JIRA Software to the development tools listed below. We recommend that you use the latest version of each application – if you're using earlier versions, see the version matrix to find out which features are available.

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.



Workflow triggers

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

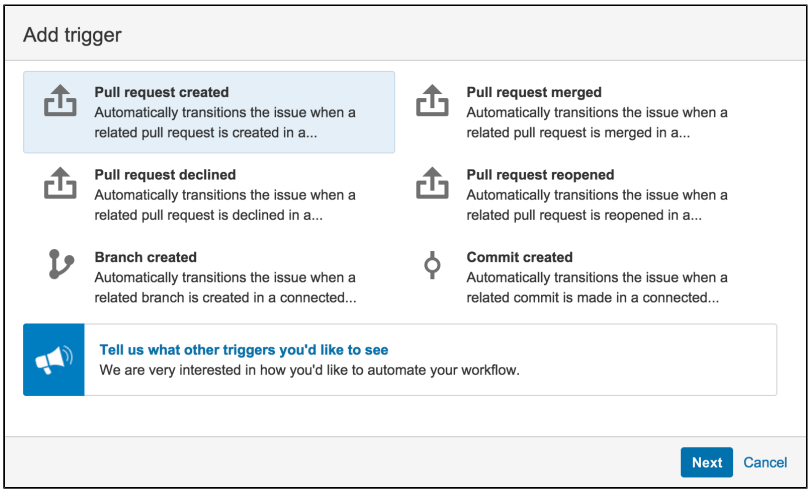
Instead of relying on developers to manually update the status of an issue after committing code, completing reviews, or creating branches, you can configure triggers in your workflow to automatically transition issues when these events occur in your development tools. For example, you could configure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

See Configuring workflow triggers f or 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

DCON-432 - Implement 'Create Branch' feature in DVCS connector plugin for Github integration **CLOSED**

— please keep this in mind when configuring trigger events.

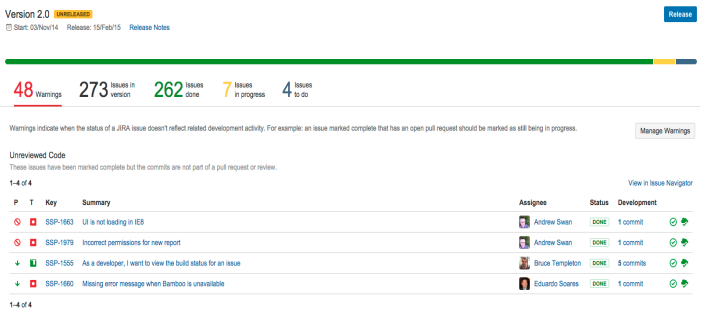


Release Hub

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



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: • View issues and developr information (from Bamboo Bitbucket, or FishEye/Crucible) for a version. • See warnings that help yo reconcile what is happeni development with JIRA da • Release a version, manag 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 • Transition JIRA issues fro within Bitbucket Server ar FishEye/Crucible
6.2+	3.3+/3.3+	5.4+	Current	Bitbucket Server 4.0+ (Stash 2.10+)	Current	11.10.290+	Development panel: • Bitbucket: create and viev branches and pull reques from an issue, view comm • FishEye/Crucible*(Git/Sub on/Perforce/CVS)*: brows search repos, view comm and branches, create and view reviews • Bamboo: view the status builds and deployments fr the related issues
6.1.+	*N/A*	*N/A*	Current	Bitbucket Server 4.0+ (Stash 2.8.x)	*N/A*	*N/A*	Development panel: • Create branches in Bitbuc

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 avoid 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

Connect to Bitbucket Cloud and GitHub

Set up the connection to Bitbucket Cloud (or other services such as GitHub and GitHub Enterprise) using the DVCS Connector that is bundled with JIRA, following the instructions in Linking Bitbucket Cloud and GitHub accounts to JIRA Software.

Note that you won't see the **Commits** tab at the bottom of the JIRA View Issue screen any more.

Troubleshooting

JIRA Application Development panel displays error - Couldn't read data

Administering Bitbucket and GitHub with JIRA applications

Using Bitbucket and/or GitHub with your JIRA Software application allows you to work more effectively and efficiently with your development tools. If required, you can link multiple JIRA instances to one Bitbucket instance, and you can link multiple Bitbucket instances to one JIRA instance.

When JIRA applications are connected to Bitbucket or GitHub accounts:

- You can link all, or a subset, of your existing repositories (both public and private) to JIRA Software.
- New repositories are automatically linked to JIRA Software.
- Create new branches in a repository from a JIRA Software issue, and see those branches in the issue.
- Reference JIRA Software issues in a commit message and have those commits appear in the Development panel of JIRA Software issues.
- Create new pull requests from a JIRA Software issue, and see all the related PRs in an issue.
- Transition issues through a JIRA Software Cloud workflow (for example, Close, Reopen, and so forth) using Smart Commit messages. See Processing JIRA Software issues with Smart Commit messages.

Read more about integrating JIRA Software with Atlassian development tools: Streamlining your development with JIRA Cloud.

You can connect Bitbucket Cloud or GitHub accounts with JIRA applications using the free JIRA DVCS

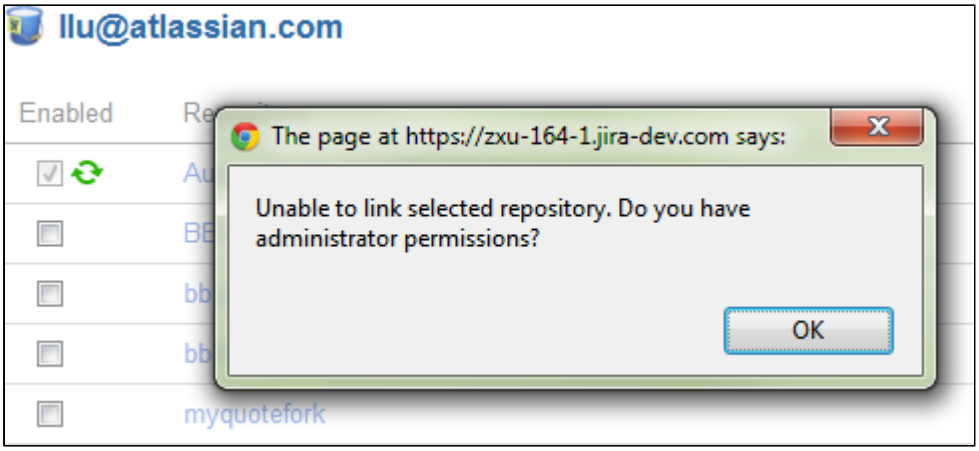
Connector add-on. This add-on comes bundled with JIRA Software. For JIRA Server versions earlier than 7.0, this is a system add-on that you can install. For more details, see Linking Bitbucket Cloud and GitHub accounts to JIRA Software.

Notes and Tips
• The Bitbucket links feature offers a similar functionality for JIRA Software and other services. • If you want to report a new issue, provide feedback, or require help, please raise a request in the issue tracker for JIRA DVCS Connector project.

Current limitations of this feature

The following limitations exist for this feature:

- There is no way to link user accounts on JIRA Software with user accounts on Bitbucket. This functionality will come with Atlassian account in the future.
  However, when you create JIRA Software users, emails will be automatically sent to the new users, inviting them to join Bitbucket. For details, see Managing Users.
- If you use email addresses to connect Cloud with Bitbucket, repositories cannot be linked and you will see an error similar to the one shown in the screenshot below. To work around this, use your Bitbucket account name instead of email address.



Using this feature

See the following for details about using this feature:

- Linking a Bitbucket or GitHub repository with JIRA
- Enabling DVCS Smart Commits
- Getting started with Bitbucket and JIRA Cloud

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. 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. SeeStreamlining your development with JIRA.

This page explains how to link a Bitbucket Cloud or GitHub account to JIRA.

- Make sure you understand how JIRA Software connects to your DVCS account
- Link a DVCS account to JIRA Software
- 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 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

When you link a Bitbucket or GitHub account to JIRA Software, 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 Cloud

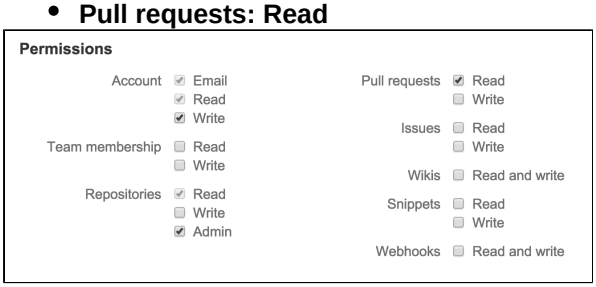
Log in as a user with administrative rights on the account.

1. Choose ***avatar*** **> 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**



   These are the minimum permissions required by the JIRA DVCS connector.
   Selecting additional permissions will have no adverse affects on the integration.
7. Click **Save**.
8. Click the name of your new consumer to see the OAuth **Key** and **Secret** values.
9. 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:

1. Choose **Edit Your Profile**.
2. Select **Applications**.
3. Choose **Register new application**.
4. Enter `JIRA DVCS` for the **Application Name**.
5. 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

   DCON-467 - 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.

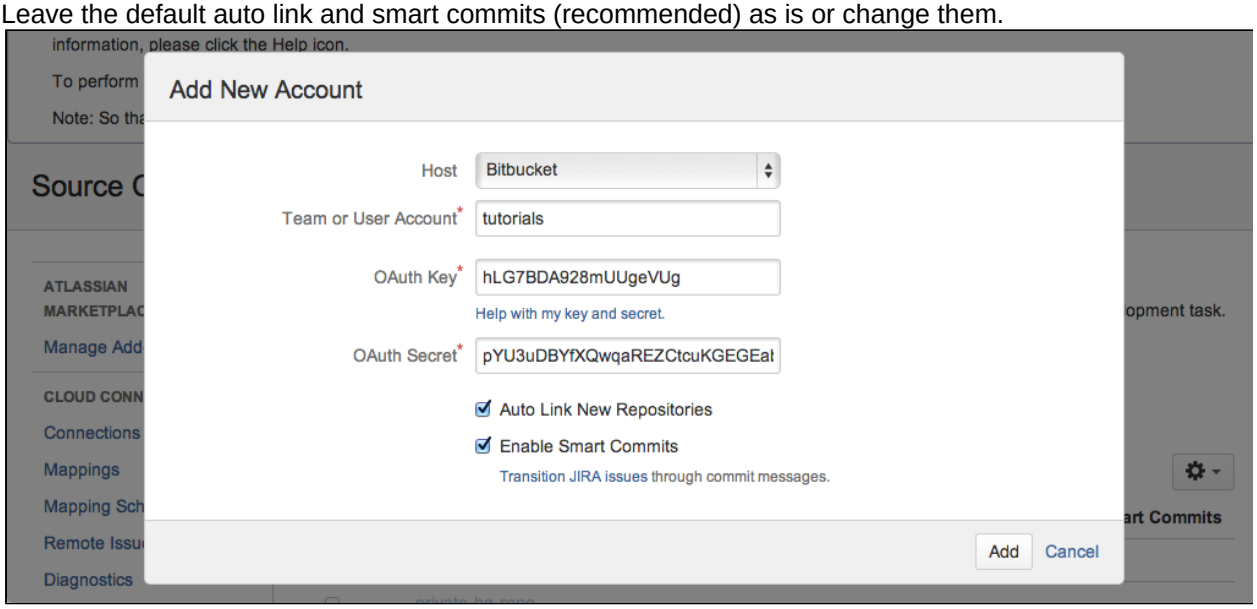
6. Keep your browser open to your DVCS and go to the next step.

Step 2. Link the account on JIRA Software

Do the following to complete the link between your DVCS and JIRA Software:

1. Log in to JIRA Software as a user with administrative rights.
2. From the JIRA Software dashboard click the
   
   (settings) icon.
3. Choose **Applications**.
4. From the **Integrations** section on the left, choose **DVCS accounts**.
5. Click **Link Bitbucket Cloud or GitHub account**.
6. Choose **Bitbucket Cloud** as your **Host** value.
7. Enter a **Team or User Account** name.

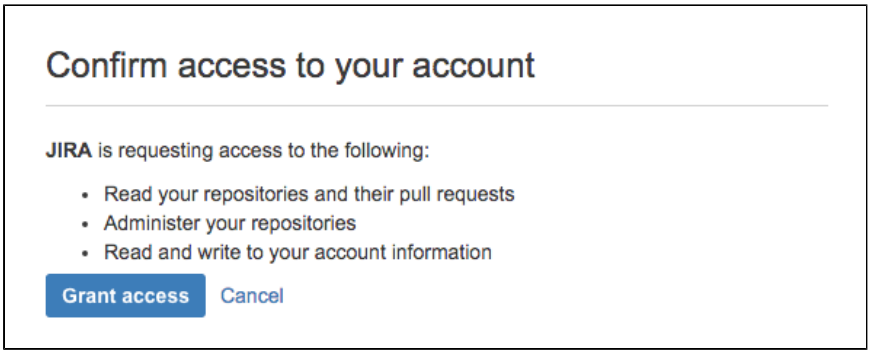
   For example, if you want to link the account that owns the `https://bitbucket.org/tutorials/markdowndemo` repository, you would enter `tutorials` for the **Team or User Account** value. Linking the `tutorials` account links all of that account's repositories, not only the `markdowndemo` repository.
8. 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:



12. When JIRA connects successfully, you'll see your account on the 'DVCS accounts' page.

The account you just connected and all of its repositories appears in the 'Managed DVCS Accounts' page. The initial synchronisation starts automatically. After that, the system continues to sync your repository automatically on a regular basis.

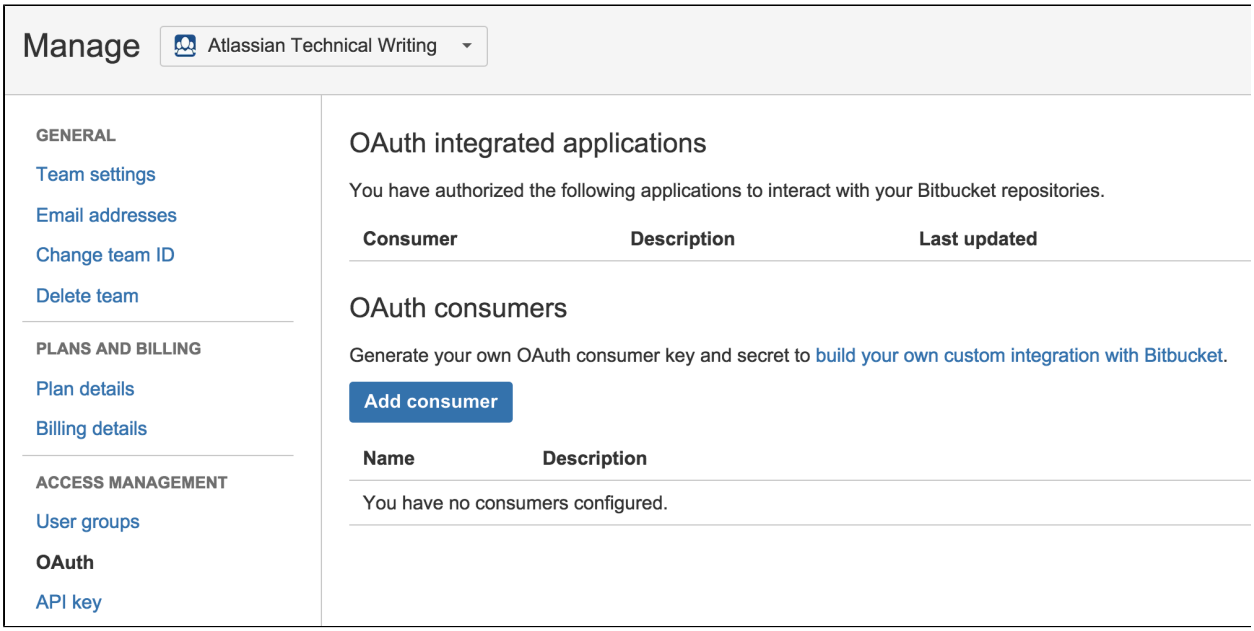
Automatic synchronisation and temporarily disabling a link

After you link an account, JIRA Software automatically starts looking for commits that reference issue keys. The summary shows the synchronisation results and errors, if any. A synchronisation of commit data from the DVCS repository to JIRA Software can take some time. As the synchronisation 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 Cloud

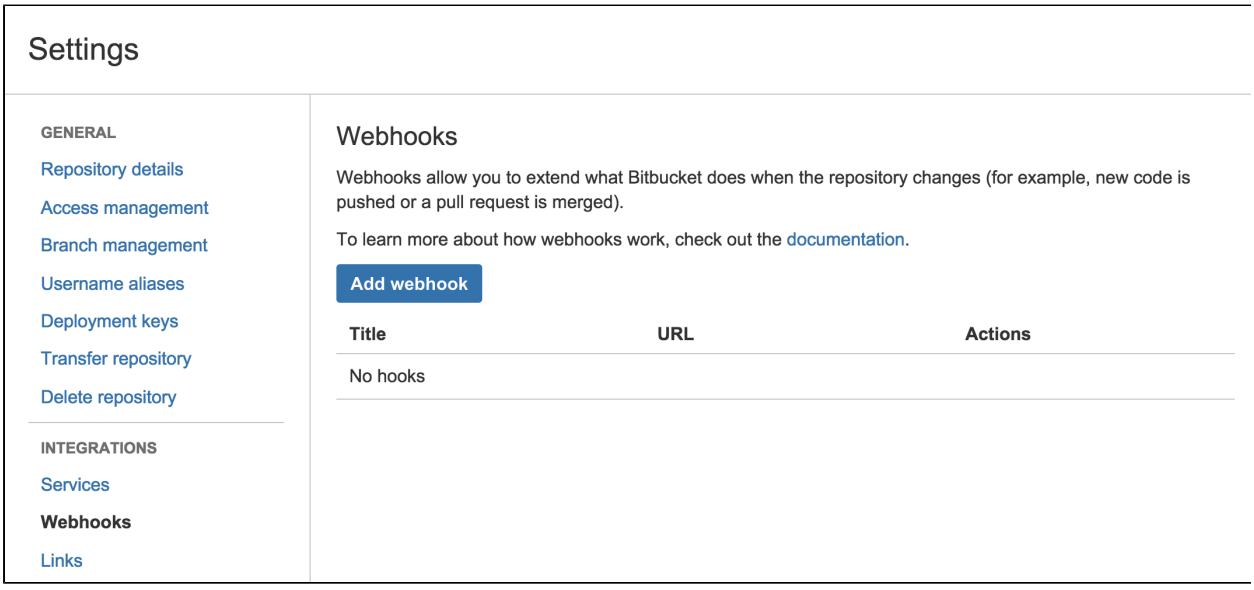
The DVCS Connector does two things:

- 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:



- 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:



The DVCS Connector uses its link to check for new repositories on the account, then adds this service to

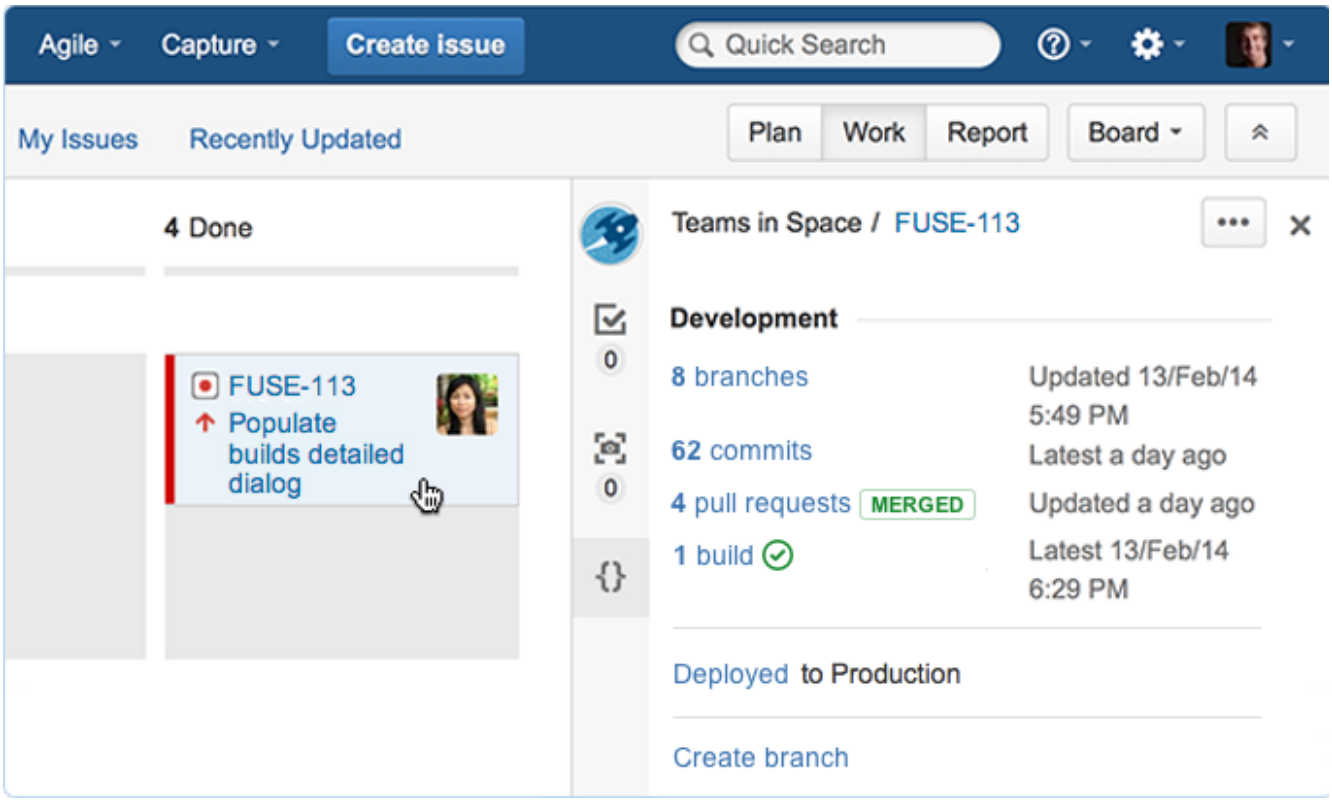
those as well. You see the result of all this on the 'Services' page.

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:



See "Integrating with Development tools" (JIRA Server / JIRA Cloud) 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.

Related pages

Managing linked Bitbucket and GitHub accounts

Processing JIRA Software issues with Smart Commit messages

Enabling DVCS 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

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:
Bitbucket Cloud or GitHub

Connect using the 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 add-ons for more information.

A JIRA administrator with access to the Bitbucket Cloud or GitHub account must set up OAuth authentication with JIRA Software. See Linking Bitbucket Cloud and GitHub accounts to JIRA Software for details.
Bitbucket Server, FishEye or Crucible

Connect using an application link.

See Using AppLinks to link to other applications.

Then enable Smart Commits in JIRA Software

Smart Commits must be enabled in JIRA Software:
Bitbucket Cloud or GitHub

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.

Control Smart Commits in JIRA Software...

Control whether Smart Commits are enabled for new repositories:

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**.

Enable or disable Smart Commits on individual repositories:

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 or Crucible

A JIRA administrator can control Smart Commits for each account in the connected application (Bitbucket Server, FishEye or Crucible).

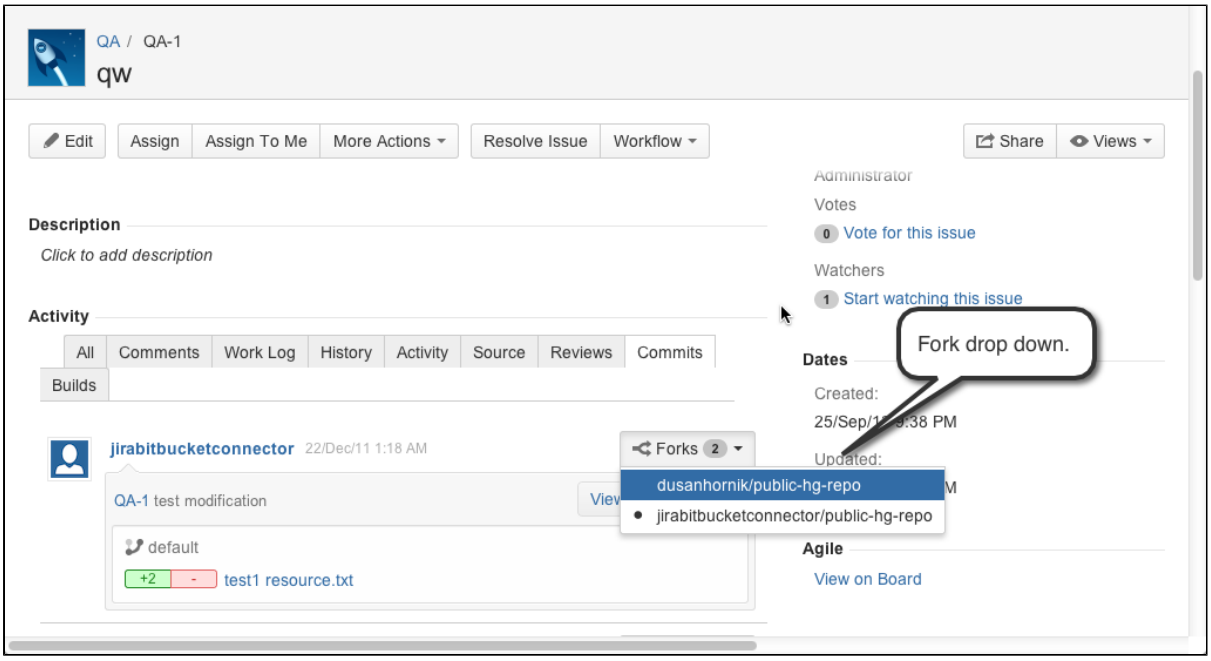
Control Smart Commits in JIRA Software...

Enable or disable Smart Commits on individual accounts:

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.

Forks and Smart Commits

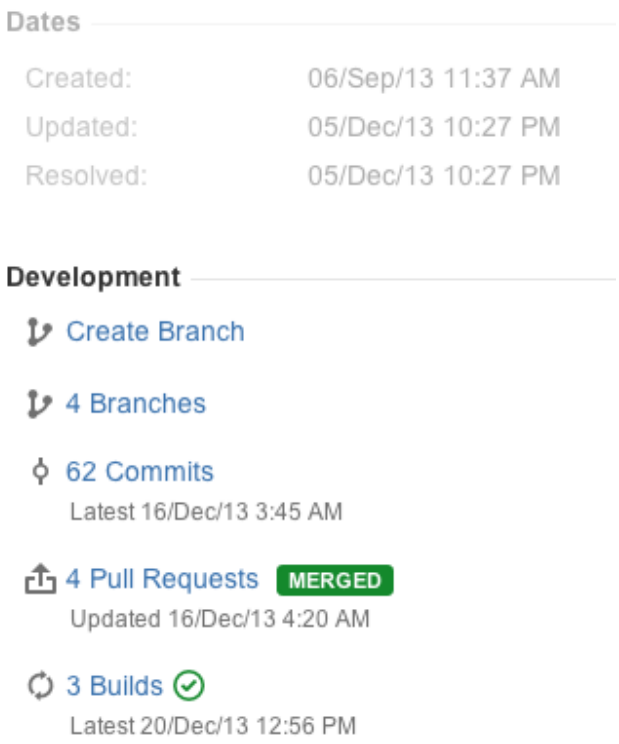
If you use forks in your workflow, the DVCS Connector records each repository that contains a Smart Commit message. It actually processes the Smart Commit message only the first time it encounters it. When you view the commit tab in JIRA Software, you can see which forks include that particular commit:



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.



The builds reference in the preceding figure will not be present unless you Integrate JIRA with Bamboo.

On this page:

Before you begin make sure you understand...

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 **`atlassian_tutorial`** has a corresponding Bitbucket URL of `https://bitbucket.org/`**`atlassian_tutorial`**.
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.

Prerequisite: **You have an individual Bitbucket account**	

Create a team
Two user groups, **Administrator** and **Developer**, are created by default when you create a team.
To create a team:

1. Select **Teams > Create team**.
2. Fill in the available fields:
   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.
   Note: you are already a member of the team and the administrator by default.
4. Click **Create**.

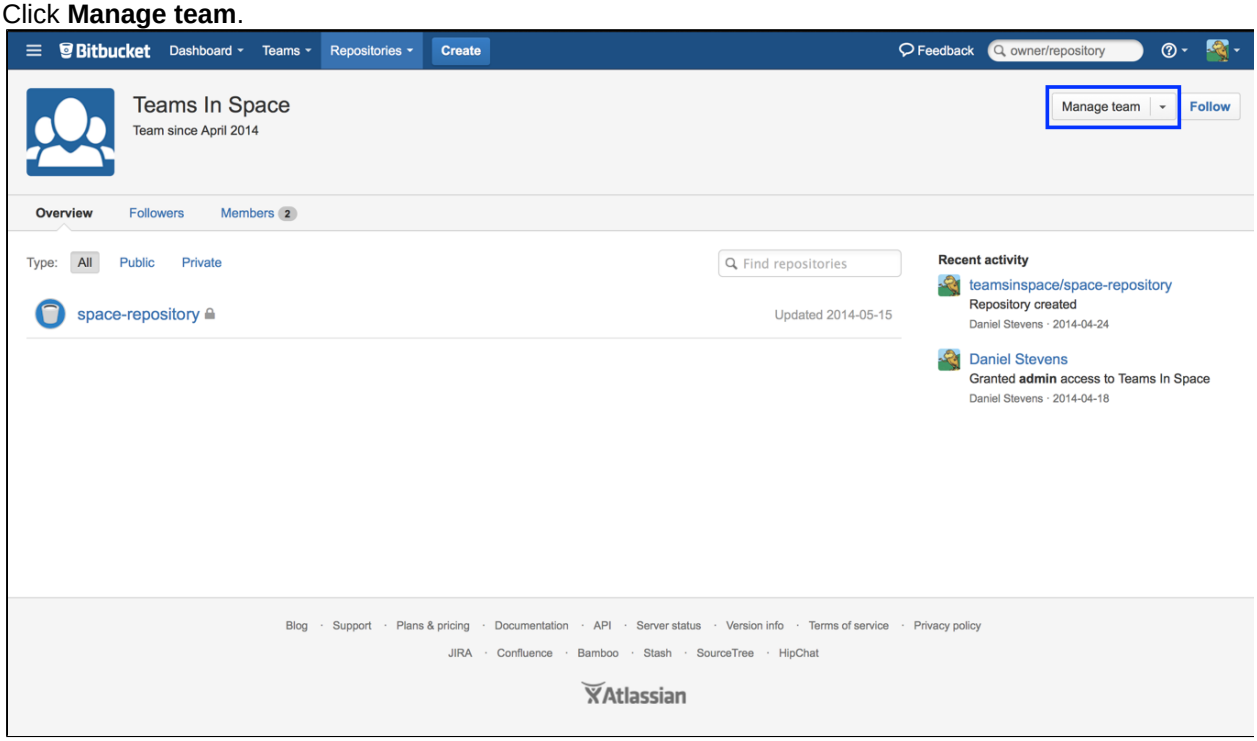
Congratulations you have a team! You are taken to the team overview page where you can create your first team repository or manage the team's settings.

Step 2. Invite team members to your team on Bitbucket

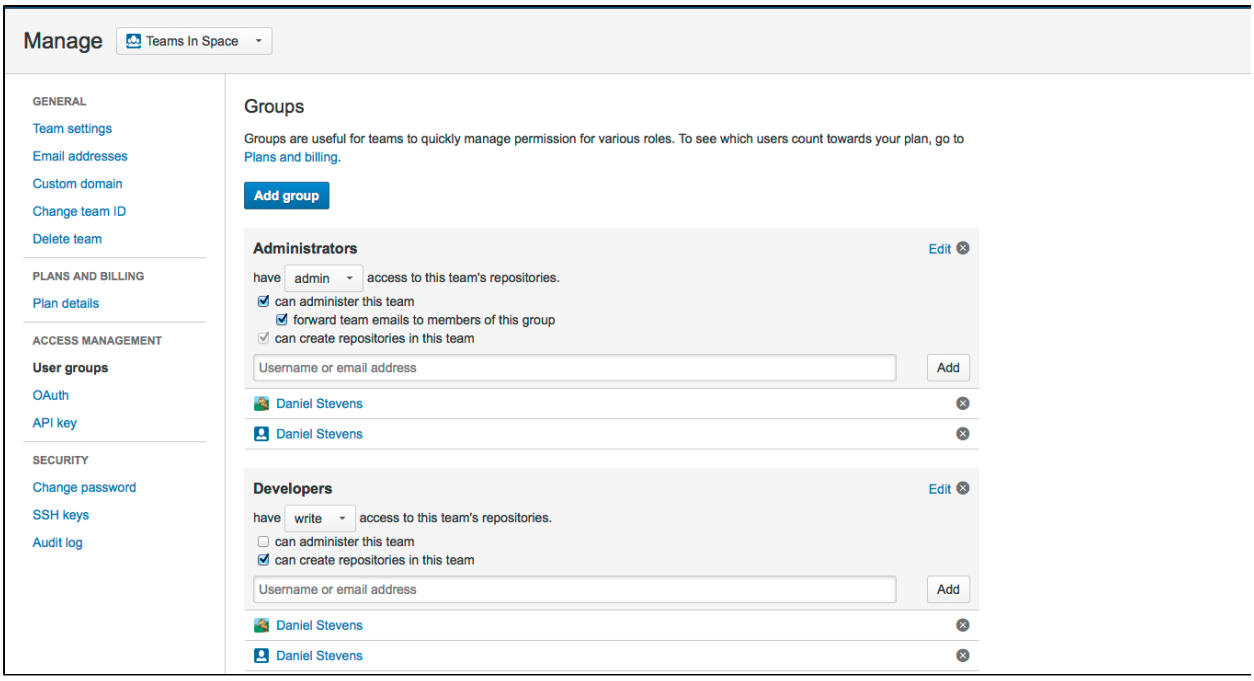
You can add team members to your linked Bitbucket team account. These may be the same users you added to JIRA. It is your choice. Users that you add to Bitbucket need not have accounts on the JIRA instance.

When editing a team, log into an account with administrative access to the team. To add a user to a team, you add the user to one of that team's groups by doing the following:

1. Log in to Bitbucket as a user with administrative rights on the team.
2. Select the team you want to administer from the **Teams** list.
   The system takes you to the team profile page.
3. Click **Manage team**.



   The system takes you to the **Groups** page in the team's account settings.
4. Locate the group you want to add the user to.
5. Enter a **Username or email address** in the field provided.



When you type in the field, the system attempts to auto complete the name for you. You can also enter a email address.

6. Click **Add**.
   If you entered an email address and it has a corresponding Bitbucket account, the system resolves the account for you. If Bitbucket could not resolve the address, it sends the user an invitation to join the team by creating a Bitbucket account.
7. Repeat steps 5-6 for each user you want to add.

Step 3. Create repositories in or move repositories to your Bitbucket account

A repository (sometimes called a repo) contains your project code.
Create a repository and add some code.

You can choose either an individual account or, if you have permission, a team as owner of the repository.

Creating a repository with you as the owner also makes you creator and you will have administrator access which is, essentially, irrevocable.

If you are a member of a team creating a team owned repository means:

- You are listed as the creator and automatically have administrator access to the repository.
- As creator you will have to be removed from the team to be removed from the repository.

This page

Related pages

Bitbucket Cloud Teams
All about how to create, manage, and grow your Bitbucket team.
Use and administer repositories
Once you have a repository learn how to work with all the features Bitbucket provides for your repository.

Create a repo on Bitbucket

To create a repo, do the following:

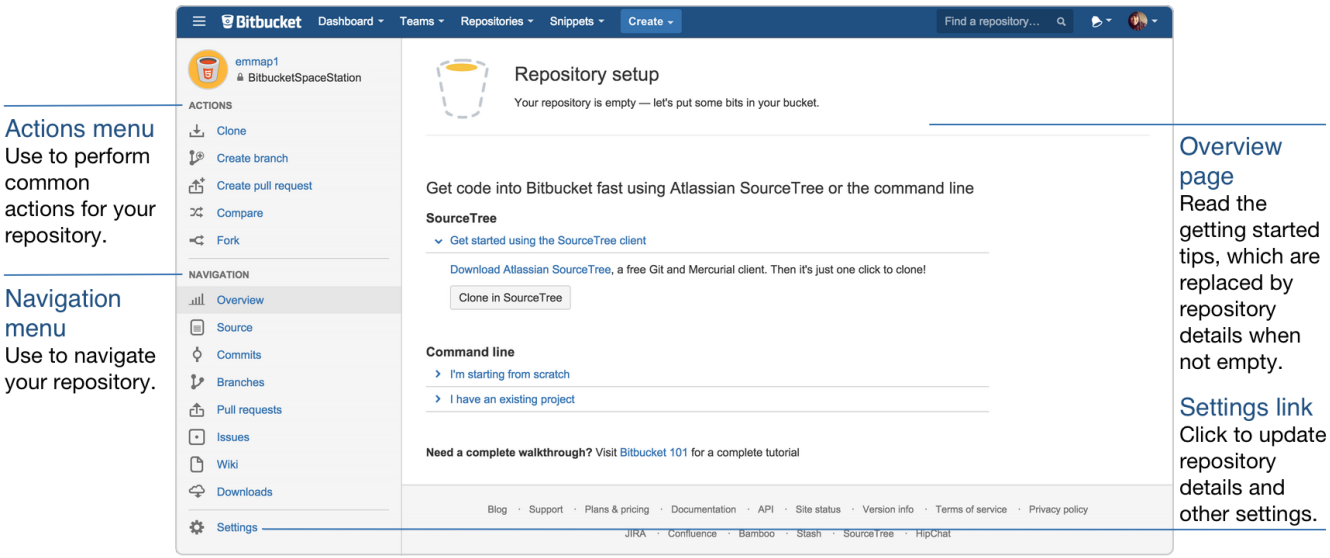
1. Log into Bitbucket Cloud under your individual account.
2. Click **Repositories > Create repository** or the Create new repository link.
3. Choose a repository **Owner**.
   This only appears if you are creating under an account with membership in one or more teams.
4. Enter a **Name** and **Description** for your repository.

5. Tick **Private** if you want to hide your repository from the general public, so that only selected people can see it.
6. Select the **Repository type.**
7. Click **Create repository**.

> If you create a repository with mixed upper and lower case name, Bitbucket converts the name to all lower case in the repository URL. The name appears in mixed case in the UI. You cannot create two repositories with names that result in the same URL.

How to add code to your newly created repo

After you create a repository, Bitbucket provides you with a help panel. This help panel is an easy way to get command tips for moving forward. Simply click to expand the section on the **Overview** page that applies to your situation.



For example, if you have an existing project to push up:

Git	Mercurial

`cd /path/to/my/repo`
`git remote add origin ssh://git@bitbucket.org/username/bbreponame.git`
`git push -u origin --all`
	`cd /path/to/my/repo`
	`hg push ssh://hg@bitbucket.org/userr`

Or if you are starting from scratch:

Git	Mercurial

`mkdir /path/to/your/project`
`cd /path/to/your/project`
`git init`
`git remote add origin ssh://git@bitbucket.org/username/bbreponame.git`
	`cd /path/to/my/repo`
	`hg push ssh://hg@bitbucket.org/username/b`

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 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

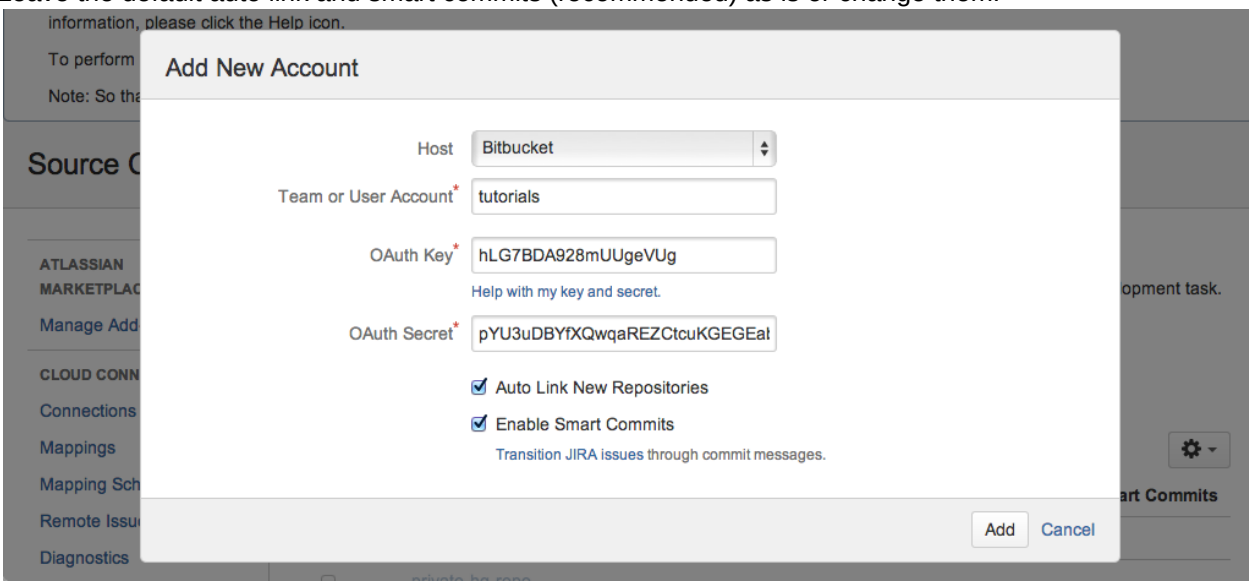
To complete the link between your DVCS and JIRA:

1. Log in to JIRA Cloud as a user with administrative rights.
2. From the JIRA dashboard click the

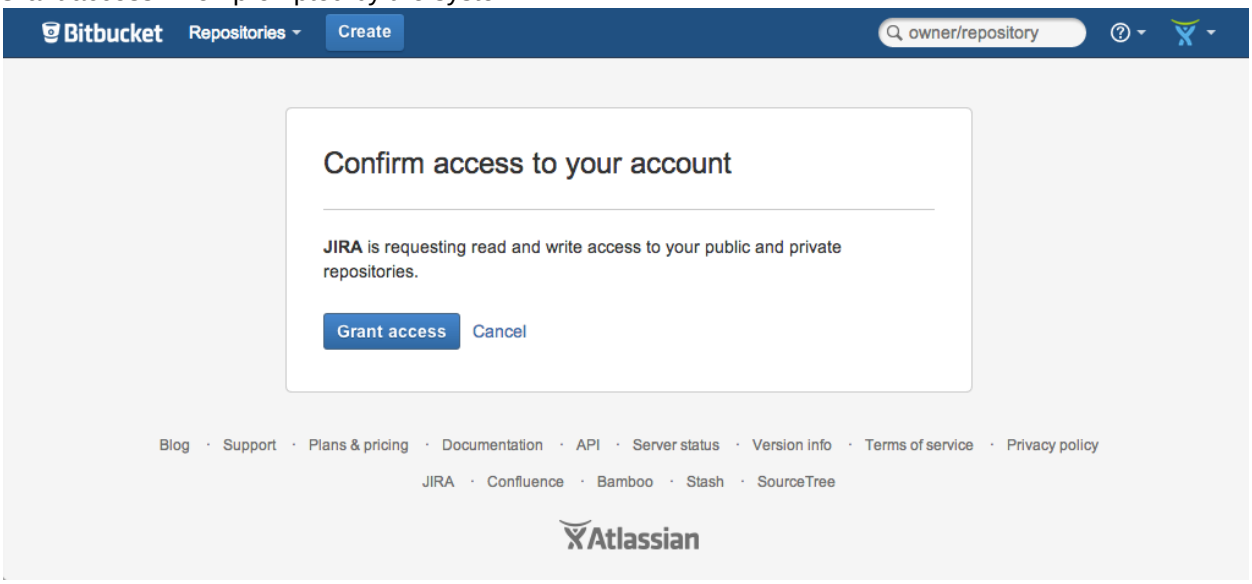

(settings) icon.

3. Choose **Applications** then **DVCS accounts** (under 'Integrations' in the left-hand panel).
4. Click **Link a Bitbucket Cloud or GitHubaccount**.
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 synchronisation starts automatically. After that, the system continues to sync your repository automatically on a regular basis.

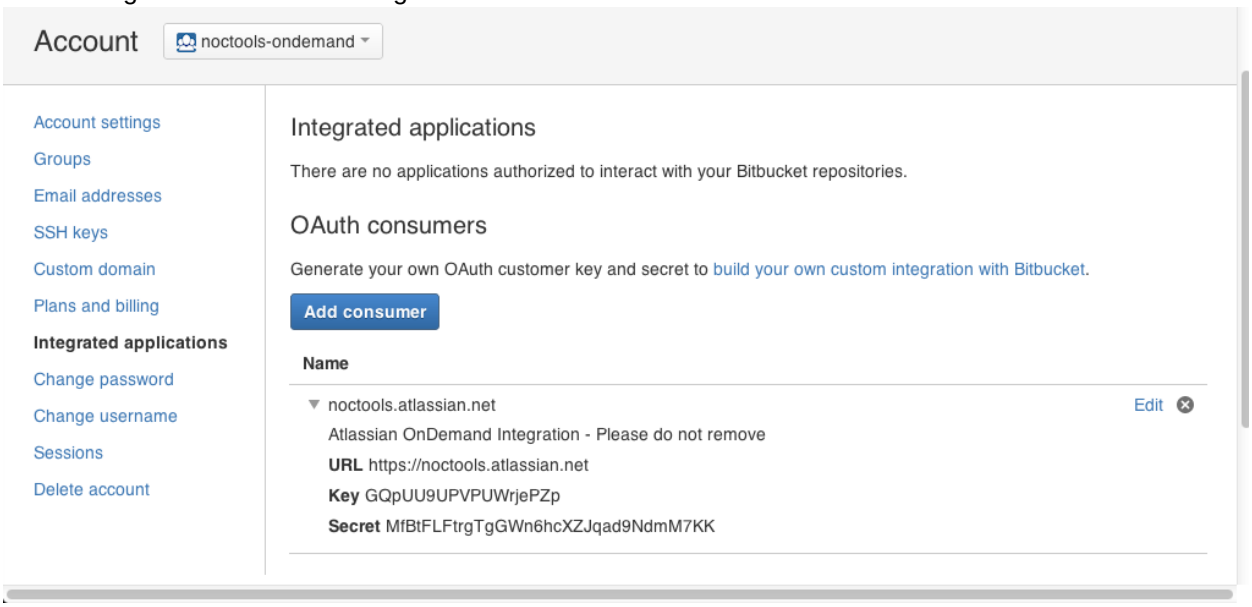
Automatic synchronisation and temporarily disabling a link

After you link an account, JIRA automatically starts looking for commits that reference existing issue keys. The summary shows the synchronisation results and errors, if any. A synchronisation of commit data from the DVCS repository to JIRA can take some time. As the synchronisation progresses, the commits appear in related issues. You can always enable and disable the linking of repositories with JIRA as needed.

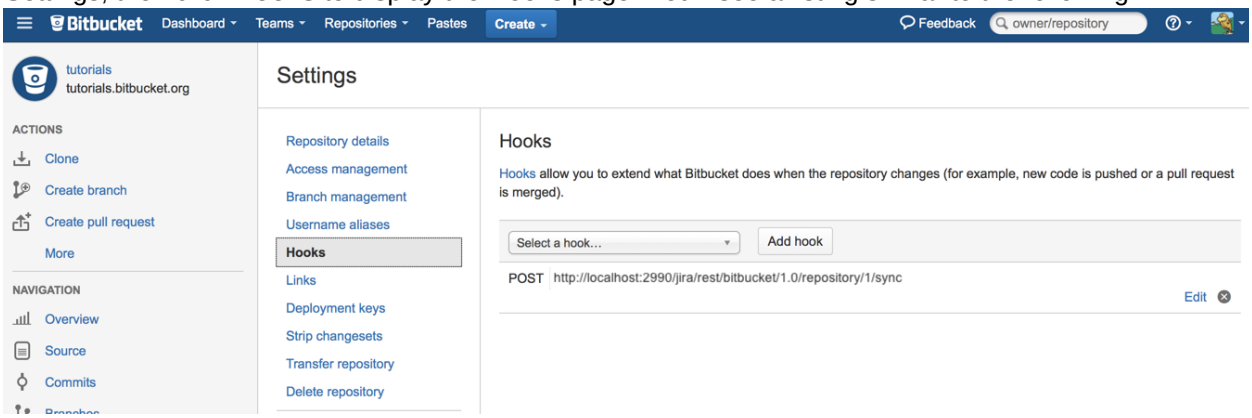
How the link appears in Bitbucket

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.

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.

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.

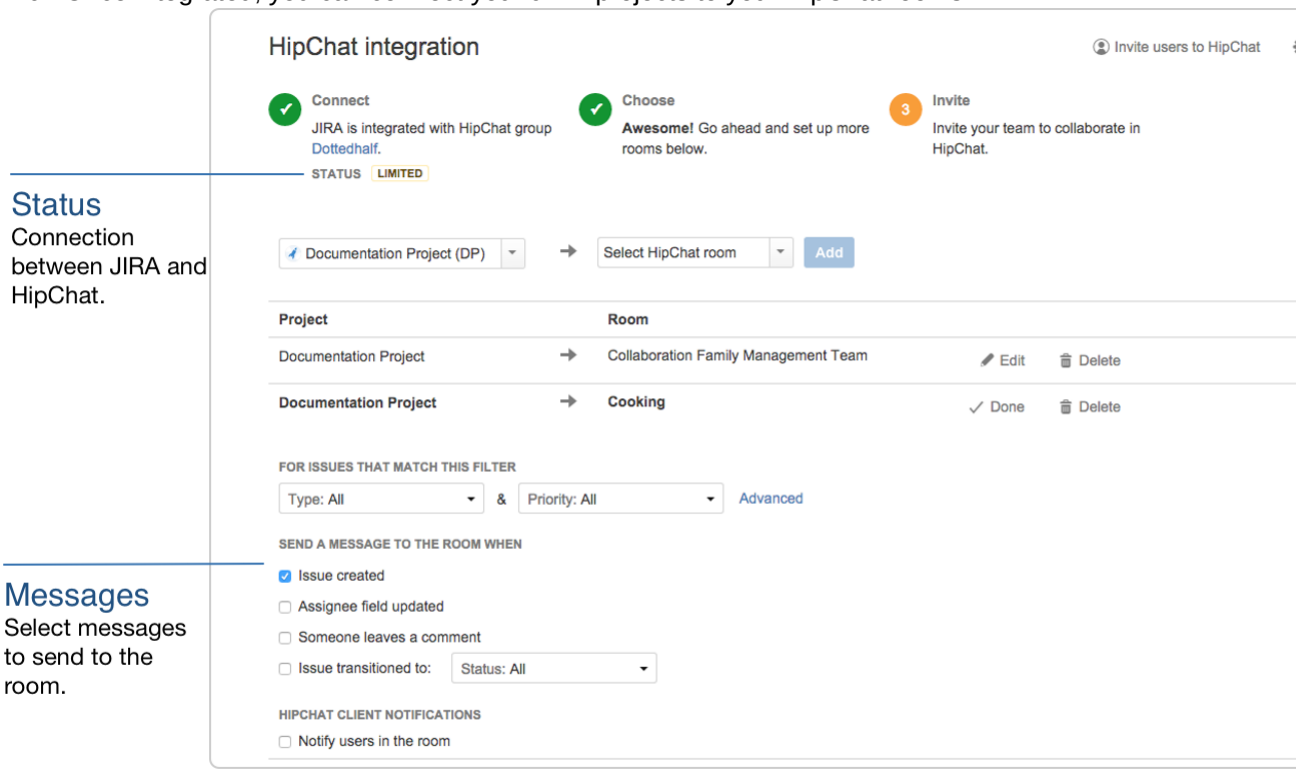
- **Connected:** HipChat and JIRA are connected and working just fine. Carry on.
- **Limited:** HipChat cannot connect to your JIRA server - it may be behind a firewall. You can still receive messages from JIRA in HipChat, but some functionality (such as Issue Preview and @mentions) may not work.

- **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

The JIRA and HipChat integration is packaged with JIRA 6.4 and later versions. For previous versions of JIRA or for the latest integration plugin go to the HipChat for JIRA integration plugin page and follow the installation instructions.

1. Log in as a JIRA administrator or a Project Administrator.
2. Go to the **JIRA** administration console > **Applications**.
3. Scroll down the page to the **Integrations** section and select **HipChat**.
4. Select **Connect HipChat**.
5. Follow the instructions to link JIRA to your HipChat site.
6. Once integrated, you can connect your JIRA projects to your HipChat rooms.



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.

Using AppLinks to link to other applications

Application Links (sometimes called "AppLinks") is a bundled plugin that allows you to link Atlassian applications

to each other. Linking two applications allows you to share information and access one application's functions and resources from within the other.

Atlassian recommends only 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 user impersonation with 2-Legged OAuth authentication will be configured for this link. You can change this later if necessary.
    - 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** link.
- 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

Atlassian's application links provide both OAuth and OAuth with impersonation authentication types:

OAuth authentication

Non-impersonating authentication allows you to link applications when the applications don't share the same user base.

It always uses a pre-configured user, and not the logged-in user, when making a request. The server handling the request determines the level of access to use based on the access permissions of that pre-configured user, and this is used for requests from all users.

See OAuth security for application links for more information.

OAuth with impersonation

Impersonating authentication makes requests on behalf of the user who is currently logged in. People see only the information that they have permission to see. This authentication type should only be used when the two servers share the same user base.

Impersonation provides greater convenience for your users – they don't need to authenticate when they request restricted resources from the remote application. The following restrictions apply:

- Both applications must be Atlassian applications.

- Users should have the same user account and use the same password on both applications.

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.

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 p age. 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:

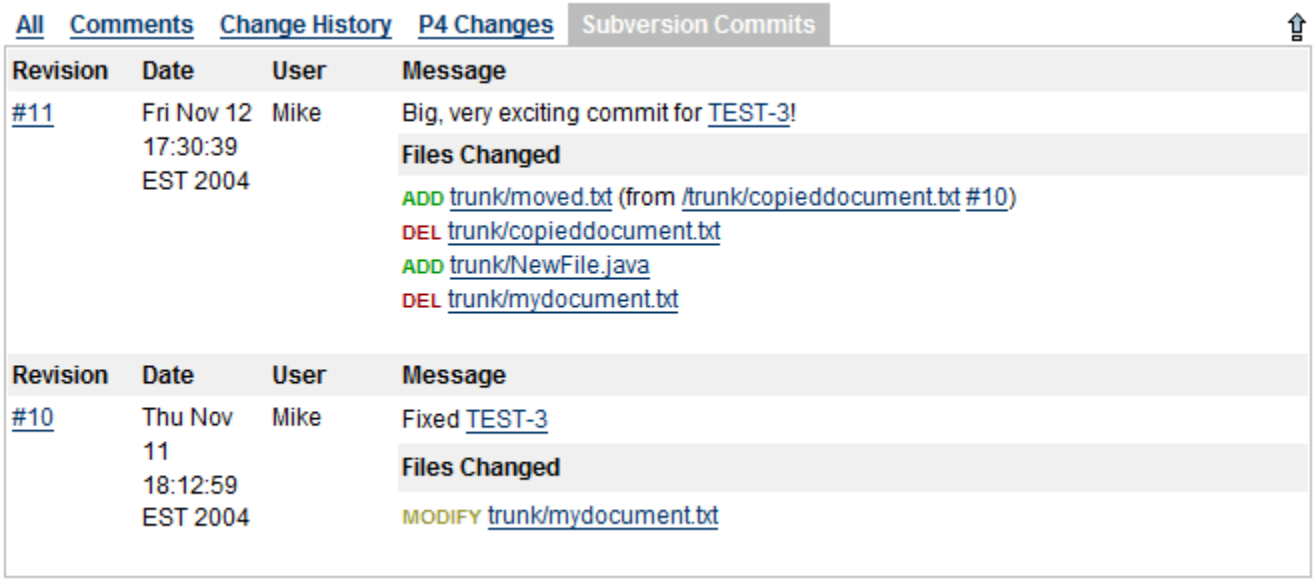
See Zephyr's JIRA Overview & Setup documentation.

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 either by using Atlassian FishEye or the Subversion add-on. 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.



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

JIRA has the following concrete Listeners (which extend the base JiraListener interface):

com.atlassian.jira.event.**JiraListener**	The base interface which all other JIRA listener interfaces extend. Covers core listener properties like uniqueness, description, parameters etc. API doc
com.atlassian.jira.event.issue.**IssueEventListener**	The main listener interface in JIRA, used whenever anything happens to an issue. API doc
com.atlassian.jira.event.user.**UserEventListener**	This listener is called whenever anything happens to a user within JIRA. API doc

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.

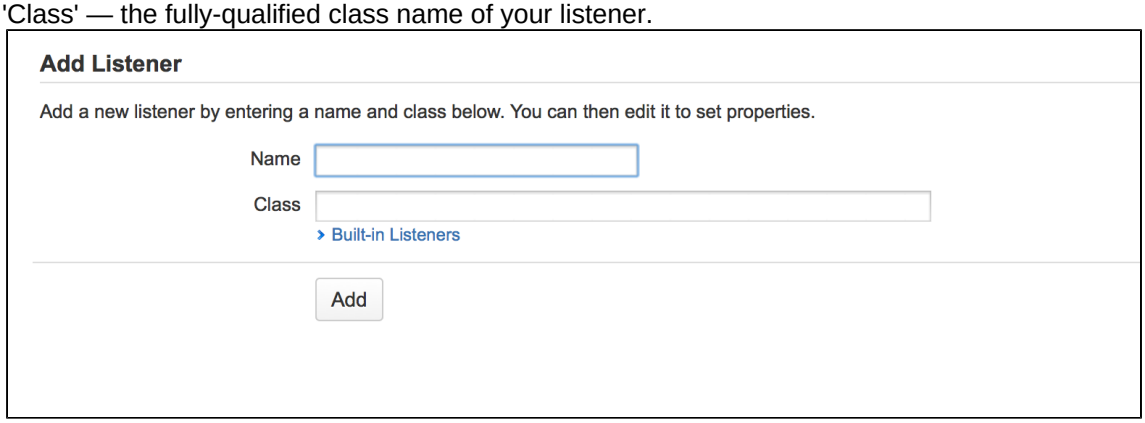
Other examples of useful tasks that can be accomplished with listeners are:

- **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

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-application-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.



   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 add-ons

About add-ons

On this page:

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.

> You may notice that the terms 'add-on' and 'plugin' both appear in the Atlassian documentation and tools. While the terms are often used interchangeably, there is a difference. A plugin is a type of add-on that can be installed into an Atlassian host application. Plugins are what developers create with the Atlassian SDK. But there are other types of add-ons as well. For example, the JIRA client is an add-on that runs as a separate program rather than as a plugin to JIRA. This documentation uses the term 'add-on' most often.

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.

Managing webhooks

On this page:

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.

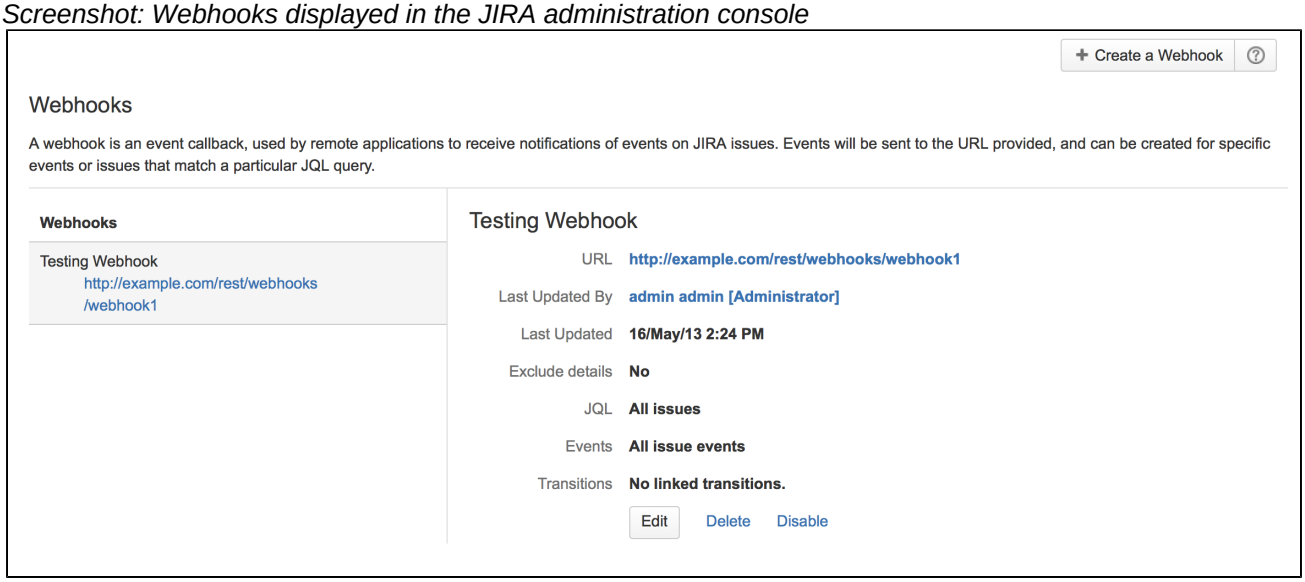
*Please read the **JIRA Webhooks** page (JIRA developer documentation) for detailed information on how to configure JIRA webhooks, including a*

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.

On this page:

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 installation directory (as JAR files).

1. Choose
   

> 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.
      To use one of JIRA's built-in service classes, first click the **Built-in Services** link to expand the list of service classes and then click the name of the specific class in the list. The fully-qualified class name of the built-in service will be added to the **Class** field.
    - **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.
   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.

Upgrading JIRA applications

> If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

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
High / Mission Critical *This method is recommended for enterprise environments where extended or unplanned downtime might negatively impact the business.*	Any	any	any	any	Upgrading JIRA with a fallback method
Low – Medium *If you use JIRA in a non-mission critical environment, depending on your specific installation details, you use either the safe (no downtime) or manual method to upgrade.* *If you are upgrading from JIRA 4.3.0 or later, you also have the option to use the upgrade capabilities built into the installer to perform a rapid upgrade of your existing JIRA installation - this method is the fastest way to upgrade - however due to its in-place upgrade method, having recent backups is crucial for this option.*	Neither operation system, database or home directory will be changed.	MS Windows / Linux	Standalone	4.3.0 or later	Upgrading JIRA using a rapid upgrade method
				4.2.x or earlier	Upgrading JIRA manually
		Solaris	any	any	Upgrading JIRA manually

Notes:

- We strongly recommend that you upgrade to the latest production version available. If you upgrade to an older version of JIRA, it may contain security vulnerabilities that were patched or fixed in subsequent versions. If you have to upgrade to an older version, check if there are any security advisories that apply to the version.
- If you plan to skip multiple major versions of JIRA when you upgrade, please review the Skipping major versions when upgrading JIRA for important information on the recommended way to skip versions.

Upgrading JIRA applications manually

If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

This page describes how to upgrade JIRA installations that don't support the rapid upgrade method or fallback method. You should use this method to upgrade JIRA if you meet any of the following criteria:

- You use JIRA 4.0.0 or later on Solaris.
- You use JIRA 4.0.0 – 4.2.x on Windows or Linux.

See Upgrading JIRA applications for more information on the methods you can use to upgrade JIRA.

On this page:

1. Before you start

- **Read about the new version** - Review the release notes and upgrade notes for the version of JIRA that you are upgrading to. See our Release notes for JIRA Server. If you plan to skip a few JIRA versions during your upgrade, **we strongly recommend** that you read the upgrade guides for all major versions between your current version and the version to which you are upgrading.
- **Check your license** - Verify that your license support period is still valid.
- **Check for known issues** - Use the JIRA Knowledge Base to search for any issues in the new version that will affect you.
- **Check for compatibility**:
  - Confirm that your operating system, database, other applicable platforms and hardware still comply with the requirements for JIRA 7.1. The End of Support Announcements page also has important information regarding platform support for future versions of JIRA.
  - If you have installed JIRA plugins (i.e. not included with JIRA), verify that they will be compatible with the version of JIRA you are upgrading to. You can find a plugin's compatibility information from the plugin's home page on the Atlassian Plugin Exchange.
  - Some anti-virus or other Internet security tools may interfere with the JIRA upgrade 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 before proceeding with the JIRA upgrade.

We strongly recommend performing your upgrade in a test environment first. Do not upgrade your production JIRA server until you are satisfied that your test environment upgrade has been successful.

- If you have any problems with your test environment upgrade which you cannot resolve, create an issue at our support site so that we can assist you.
- If you have any problems during the upgrade of your production JIRA server, do not allow your users to start using this server. Instead:
  - Continue to use your old JIRA server — this will help ensure that you do not lose production data.
  - Also create an issue at our support site so that we can help you resolve the problems with your upgrade.

2. Backing up

Before you begin the JIRA upgrade, we strongly recommend that you back up your existing JIRA installation.

2.1 Stop users from updating JIRA data

During the upgrade process, you'll export JIRA's database from your existing JIRA installation (via an XML backup) and then restore this backup into a new JIRA installation. To ensure that the data in the XML backup is consistent with the latest data in the system, you must temporarily restrict access to JIRA so users can't update the data. Refer to the Preventing users from accessing JIRA applications during backups page for more information.

⚠ **Be aware!** Inconsistent XML backups cannot be restored!

2.2 Back up your database

Perform an XML backup of your existing JIRA installation's external database. For large JIRA installations, this process may require several hours to complete.

> The 'embedded database' is the H2 database supplied with JIRA for evaluation purposes only. If you accidentally use the H2 database in a production system, perform an XML backup of this database and continue on with this procedure.

2.3 Back up your JIRA home directory

1. Shut down JIRA.
2. Locate the JIRA home directory. You can find information about the location of the directory by navigating to the `<jira-application-dir>/WEB-INF/classes/jira-application.properties` file in your JIRA application installation directory. Alternatively, you can open the JIRA configuration tool to see the directory that is set as your JIRA Home.
3. Navigate to the directory specified in the configuration file and create a backup of it in another directory.
4. ✖ Delete the file <jira-home>/dbconfig.xml from the original folder as soon as the backup is complete.

2.4 Back up your attachments and index directories if located outside your JIRA home directory

If the attachments and index directories are located outside of your JIRA home directory, you must back them up separately. These pages describe how to find out where these directories are located in your implementation:

- Your attachments directory — Refer to Configuring file attachments page in the documentation for your version of JIRA.
- Your index directory — Refer to Search indexing page in the documentation for your version of JIRA.

Also refer to Backing up data for more information about backing up attachments in JIRA.

2.5 Back up your JIRA installation directory

The 'JIRA Installation Directory' is the directory into which the JIRA application files and libraries were extracted when JIRA was installed.

3. Setting up your new JIRA installation

> If you are running a 'mission-critical' JIRA server, we highly recommend performing the remaining steps of this guide in a test environment (e.g. using a separate test JIRA database and a copy of your JIRA Home directory) before performing the upgrade in production.

3.1 Install the new version of JIRA

Download and extract the JIRA distribution you require to a new directory. **Do not overwrite your existing JIRA installation. Ensure this has been shut down and install the new JIRA version to a new location.**

Follow the installation instructions Installing JIRA applications.

3.2 Point your new JIRA to (a copy of) your existing JIRA Home directory

If your new JIRA 7.1 installation is on a new server, **copy the backup of your existing JIRA Home Directory** from the old server to the new server before proceeding.

To set up a distribution:

1. Open the JIRA configuration tool.
2. Click the **JIRA Home** tab.
3. Update the **JIRA Home Directory** field:
    - If your JIRA 7.1 installation is on a new server, update the **JIRA Home Directory** field to the path of your copied JIRA Home directory.
    - If your JIRA 7.1 installation is on the same server, update the **JIRA Home Directory** field to the path of your **existing** JIRA Home directory.
      For more information about this directory, see JIRA home directory.

You can also set your JIRA Home Directory's location by defining an operating system environment variable `JIRA_HOME`. This value of this variable takes precedence over the value of the `jira.home` property in the `jira-application.properties` file in your JIRA installation directory. See Setting your JIRA home directory for details.

3.3 Connect the new version of JIRA to a new, empty database

Create a new, empty database that your new JIRA installation will use to store its data.

Follow the appropriate '**Connecting JIRA to...**' instructions for your database from stage 2, although from stage 4 of that procedure, be aware of the yellow note below:

- Connecting JIRA to PostgreSQL
- Connecting JIRA to MySQL
- Connecting JIRA to Oracle
- Connecting JIRA to SQL Server 2008

> If you are using a database (called **jiradb**, for example) with your existing JIRA installation and the database for your new JIRA installation is running on the same machine or database server, create your new database with a different name (e.g. something intuitive like **jiradb_440** for JIRA 4.4.0). However, ensure the new database has identical access permissions to the old JIRA database. Consult your database administrator if you need assistance with this.

You do not need to create a new database if you are using the embedded H2 database.

3.4 Migrate your existing JIRA configurations over to your new JIRA installation

If you have modified properties in configuration files of your existing JIRA installation, make the same modifications in your new JIRA installation. However, because the properties in the configuration files may have changed between versions, you cannot simply copy the configuration files from your existing installation and replace the equivalent files in the new installation.

For each file you have modified in your existing JIRA installation, you need to **manually edit each equivalent file in your new JIRA installation and re-apply your modifications**. If a file is not present in your new JIRA installation (for example, `osuser.xml` in recent JIRA versions), then simply copy that file over to your new JIRA installation.

The table below lists the most commonly modified files and their locations within your JIRA Installation Directory:

File	Location in 'recommended' (formerly 'Standalone') JIRA distributions	Description

jira-application.properties	`atlassian-jira/WEB-INF/classes`	Location of the JIRA Home Directory and Advanced JIRA Configuration in JIRA 4.3.x and earlier. Any custom property values defined in the `jira-application.properties` file of your existing JIRA 4.3.x (or earlier) installation must be migrated across to the `jira-application.properties` file of your new JIRA 7.1 installation *before you start your new JIRA installation*. Upon starting your new JIRA installation, any custom property values in the `jira-application.properties` file will automatically be migrated across to either the JIRA database or jira-config.properties file. `jira.home` is the only property of the `jira-application.properties` file subsequently used by JIRA.
setenv.bat (Windows) or setenv.sh (Linux)	`bin`	Increasing JIRA Memory
osuser.xml (not required if upgrading from JIRA 4.3.0 or later)	`atlassian-jira/WEB-INF/classes`	Modified if you have integrated LDAP with JIRA, integrated Crowd with JIRA, or if you are using a custom form of external user management or user authentication.
seraph-config.xml	`atlassian-jira/WEB-INF/classes`	Modified if you have integrated Crowd with JIRA.
server.xml	`conf`	Modified in the following situations: • If you had previously configured JIRA's TCP ports differently from their defaults. • If you had implemented SSL. • When connecting JIRA to a database in JIRA 4.3.x and earlier.

The version-specific upgrade notes contain details on properties which may have changed in these commonly modified files.

In addition to the files above, you should also consider and/or perform the following configurations as part of the upgrade process:

- **Using JIRA with Atlassian's Crowd?** — If you are using Crowd with JIRA, configure your new JIRA to talk to Crowd as described in Integrating Crowd with JIRA.
  - Remember to configure Crowd to grant JIRA's new hostname/IP access.
- **Allocating additional memory to JIRA** — If you had previously allocated additional memory to JIRA, do the same for your new JIRA instance. For more information refer to Increasing JIRA memory.
- **Plugins** — For any plugins that you had installed in your old JIRA, download the plugin version for your new version of JIRA from the http://plugins.atlassian.com site.
- **Character encoding** — Ensure that character encoding (i.e. locale) is the same on the new and old locations. Your new version of JIRA may not function correctly if attachments are moved between two

system with incompatible encoding.
- **Customisations** — If you had made any customisations (code, templates or configuration files), copy over compatible versions of these changes to the new JIRA. (The developers within your organisation who made the customisations to your old version will need to build and test equivalent changes for the new version, and provide you with the files to copy to your upgraded JIRA installation.)
- **(Optional) Running JIRA on a different port** — If your new JIRA is installed on the same machine as your old JIRA, you may wish to make sure it runs on a different port (in case you ever need to restart your old JIRA). See Changing JIRA's TCP ports for details.

3.5 Start your new version of JIRA

1. Verify that your old JIRA installation is shut down — if this JIRA server is still operating, shut it down.
2. Start up your new version of JIRA by follow the Starting JIRA instructions.

> **Do not restart your old JIRA installation...**
> If your new JIRA 7.1 installation is on the same server as your old one, it may still be configured to use the same JIRA Home directory as your new JIRA installation. Running two separate JIRA installations which share a common JIRA Home directory can lead to serious data corruption.
>
> Nevertheless, we recommend that you do not delete any aspect (or backed up component) of your old JIRA installation, until you are satisfied that your upgraded JIRA installation is functioning as expected.

3.6 Import your old JIRA data into your new JIRA

After you have started your new JIRA installation you should be logged in to your JIRA application, and ready to import the data from your old instance into the new instance. You will need the backup file of data from your old JIRA that you created earlier in these instructions (above).

To import your old JIRA data into your new JIRA:

1. Log in as a user with the 'JIRA System Administrators' global permission.
2. Select **Administration > System > Import & Export > Restore System** (tab) to open the 'Restore JIRA data from Backup' page.
3. In the **File name** field, specify the XML backup file you created previously during the export process (above). That zipped file should contain two xml files: `activeobjects.xml` and `entities.xml`. Both of these files must be included in the zipped file for the import process to work.
4. Restore the attachments directory that you backed up previously, into the attachments directory of your new JIRA. (See Restoring data.)
   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. Access JIRA via your web browser again and log in using a username from your previous JIRA installation.
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.

4. Post upgrade checks and tasks

It is strongly recommended that you perform the following checks and tasks after you have started your new instance of JIRA:

1. Check your server logs for error messages, even if JIRA appears to be running correctly. If there are any errors there that you cannot resolve, create a support case, attach your log file, and we will advise you on the errors.
2. If you were previously using External User Management, enable it in the new JIRA instance.
3. If you changed machines when upgrading, change the paths to the indexes, attachments and backup directories, from within the **Administration** section of JIRA.
4. Enable email, if you disabled it during testing.
5. If you migrated any customizations from your old JIRA to the new JIRA, ensure that they are tested thoroughly.
   a. If you had downloaded plugins for the new version of JIRA, install the downloaded JAR file(s) in your new JIRA version and carry out any other required installation for the plugin.
   b. If the plugin has a properties file, apply the same changes to it as you had in the old properties

file (don't just copy over the old properties file).

6. Once you have confirmed that the new server is working correctly, ensure that the production license is updated for the new server ID, as follows:
    a. Log in to https://my.atlassian.com.
    b. Locate the appropriate license.
    c. Edit the Server ID, as per the new production Server ID, and save it.
    d. Update the production license in the new server.

Congratulations! You have completed your JIRA migration/upgrade.

See also

Disabling auto-export
Restoring data
Upgrading JIRA applications
Switching databases

Upgrading JIRA applications with a fallback method

> If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

This page describes how to upgrade JIRA 4.4.x or later 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. You can also use this method so you have a fallback option if you are performing a complex system change, such as changing the operating system that will run JIRA, the database that will store JIRA's data or the location of JIRA's index and/or attachments paths.

Because this process is designed to provide the safest possible upgrade method, it requires advanced knowledge of database administration tasks. We recommend you have the following resources and/or skill sets available for your upgrade:

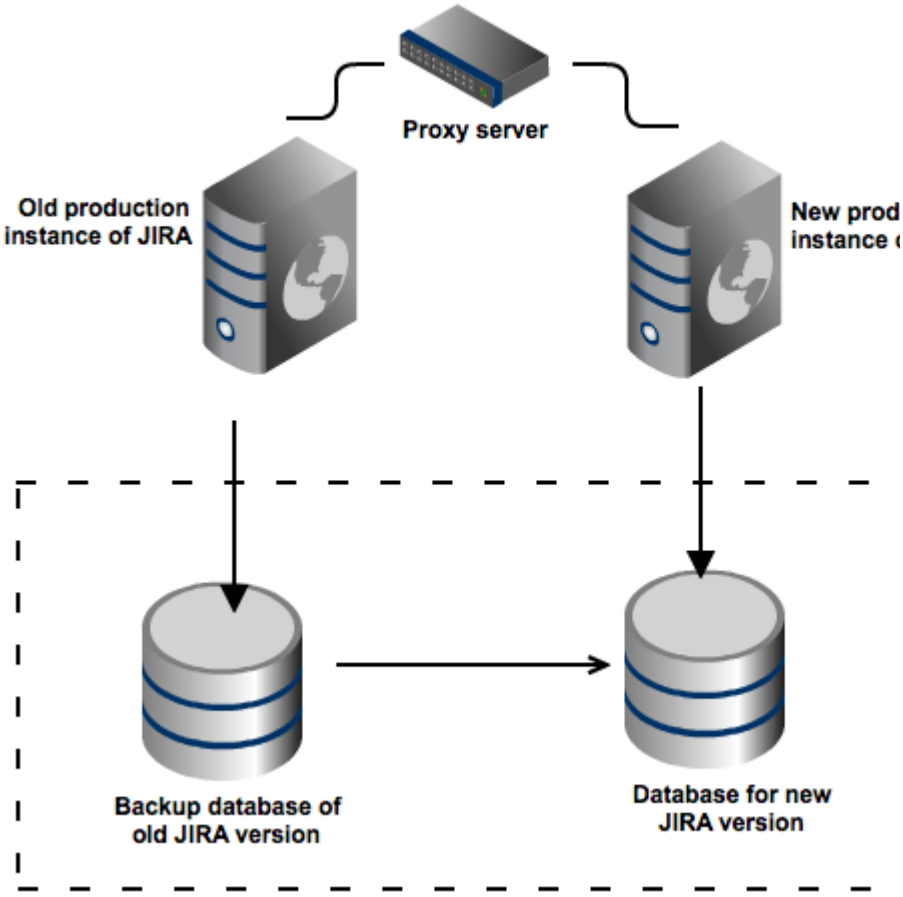
- **Database administrator** - for general production-level database administration *(i.e. run backups, create, remove, restore, etc.)*
- **JIRA application administrator** - for general application administration and upgrade managment *(i.e. JIRA SME, user with System Administrator privileges and deep understanding of application and associated dependencies within your organization.)*
- **Systems/Network administrator** - for managing systems and networks *(i.e. proxy servers, DNS changes, monitoring, VM's, hardware, etc.)*

This upgrade process also requires you to make backups of your database, which can be time-consuming. Customers with large JIRA environments should plan for four hours of downtime. If you know your system takes several hours to re-index, you might want to allocate more than four hours for the upgrade.

See Upgrading JIRA applications for more information on the methods you can use to upgrade JIRA.

> 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.



Before you start

- **Read about the new version** - Review the release notes and upgrade notes for the version of JIRA that you are upgrading to. See our Release notes for JIRA Server. If you plan to skip a few JIRA versions during your upgrade, **we strongly recommend** that you read the upgrade guides for all major versions between your current version and the version to which you are upgrading.
- **Check your license** - Verify that your license support period is still valid.
- **Check for known issues** - Use the JIRA Knowledge Base to search for any issues in the new version that will affect you.
- **Check for compatibility**:
  - Confirm that your operating system, database, other applicable platforms and hardware still comply with the requirements for JIRA 7.1. The End of Support Announcements page also has important information regarding platform support for future versions of JIRA.
  - If you have installed JIRA plugins (i.e. not included with JIRA), verify that they will be compatible with the version of JIRA you are upgrading to. You can find a plugin's compatibility information from the plugin's home page on the Atlassian Plugin Exchange.
  - Some anti-virus or other Internet security tools may interfere with the JIRA upgrade 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 before proceeding with the JIRA upgrade.

If you have any problems during the upgrade process, create an issue at our support site so that we can help you resolve the problems with your upgrade. We strongly recommend that you perform the below procedure first as a test only. This will allow you to note any complications (e.g. with customized settings or add-ons) ahead of time so that you can minimize the downtime of the system.

1. Prepare your production instance for upgrade

When you begin preparing to upgrade, it's best practice to halt any major changes to your production system (such as plugin upgrades, customizations, etc.). Keeping your production system as stable as possible will

make testing the upgrade version simpler.

It's also a good idea to let your users know about planned downtime, either through email or by using JIRA's announcement banner.

2. 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 - this change will be invisible to the end-user. If, at any point during the upgrade process, you encounter issues you can't resolve and you need to rollback to your existing JIRA instance, simply 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

3. Pre-stage and test the new version of JIRA

1. If you want to use a copy of your production data when you test the upgraded JIRA system, make a copy of your production database using your native database backup tools. See Backing up data. You can alternatively skip this step and use a new database for testing.
2. Install the version of JIRA you want to upgrade to onto a system you can use for testing (use either a test server or a separate directory on an exisitng system). This will become your new production system after you complete the upgrade process. Follow the instructions here to install a new version of JIRA: Installing JIRA applications.
3. Migrate any customizations you use in your production system. Follow the instructions in step 3.4 (Migrate your existing JIRA configurations over to your new JIRA installation) in the "Migrating JIRA to another server" page.
4. Connect the new version of JIRA to the copy of the production database (not the existing production database) or a new testing database. See Connecting JIRA to a database.
5. Start the new version of JIRA. See this Knowledge Base article about how to test mail settings without accidentally sending notifications to users from the test system: How to Prepare a Development Server's Mail Configuration.
6. Install any plugins that you use in your existing production version of JIRA. Some plugins have different compatibility for different JIRA vesions, so this step will ensure that your plugins are updated for this new JIRA version.
7. Re-index JIRA so the new plugin information is captured in the index.
8. Check out the features and functionality you use in the new version to understand how they behave and how any changes will impact your team. It can be very helpful to have a group of users look at the new system and carry out their usual tasks to make sure they won't run into any issues when the new version is in production.
   When you are ready to begin the process of migrating your production data to this new version, shut down JIRA (for example, by executing either the `/bin/stop-jira.sh` or `\bin\stop-jira.bat` fil e in your JIRA application installation directory, or by stopping the JIRA service).

4. Disable the old JIRA production instance and start the new instance

Before disabling your old JIRA production instance, identify the location of your attachments and index directories. If they are located outside of your JIRA home directory, you will back them up manually later during the upgrade process. These pages describe how to find out where these directories are located in your environment:

- Your attachments directory — Refer to the Configuring file attachments page for your version of JIRA.
- Your index directory — Refer to the Search indexing page for your version of JIRA.

If your attachments and index directories are located outside of the JIRA home directory, note their location so you can easily find them later.

After you've located the attachments and index directories, disable the old JIRA production instance so you can perform a database backup:

1. Shut down your old production JIRA instance (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. Using your database's native backup tools, perform a backup of the data in your old production JIRA instance. See Backing up data.
3. Set the newest copy of the production database as the new database for production.

   > Make sure that the database set up for the new production version of JIRA is clearly distinguishable from the database backup of your old production JIRA, and that the new production instance is not configured to connect to the old production database.

4. Synchronize the JIRA attachment directories:
   a. Locate the JIRA home directory. You can find information about the location of the directory by navigating to the `<jira-application-dir>/WEB-INF/classes/jira-application.properties` file in your JIRA application installation directory. Alternatively, you can open the JIRA configuration tool to see the directory that is set as your JIRA home.
   b. Navigate to the directory specified in the configuration file and create a backup of it in another directory.
   c. If the attachments and index directories are located outside of your JIRA home directory, you must back them up separately. (See the beginning of this task for information on how to find these files.)

      Also refer to Backing up data for more information about backing up attachments in JIRA.
   d. Replace the JIRA home directory (and the attachment and index directories, if separate from the JIRA home directory) in the new JIRA production environment with the backups you made of the old production directories.
5. Start the new version of JIRA in your new production environment. When you start this version, JIRA will upgrade your data and may perform a re-index. When the re-indexing is complete, verify that your data is present and that there are no issues with the system.

   > The re-indexing may take up to several hours, depending on the size of your instance. If you know that your instance takes a long time to index, make sure to plan your scheduled downtime accordingly.

6. Reconfigure the proxy server you set up in step 2 so that the new version of JIRA becomes your production instance. Make sure to let your users know about the new instance (including the new domain name) and any changes that might affect them.

If you experience any issues in the new production environment, you can simply revert the proxy server settings and re-instate your old production instance until you can resolve the issue.

5. Post upgrade checks and tasks

It is strongly recommended that you perform the following checks and tasks after you have started your new instance of JIRA:

1. Check your server logs for error messages, even if JIRA appears to be running correctly. If there are any errors there that you cannot resolve, create a support case, attach your log file, and we will advise you on the errors.
2. If you were previously using External User Management, enable it in the new JIRA instance.
3. If you changed machines when upgrading, change the paths to the indexes, attachments and backup directories, from within the **Administration** section of JIRA.
4. Enable email, if you disabled it during testing.
5. If you migrated any customizations from your old JIRA to the new JIRA, ensure that they are tested thoroughly.

   a. If you had downloaded plugins for the new version of JIRA, install the downloaded JAR file(s) in your new JIRA version and carry out any other required installation for the plugin.
   b. If the plugin has a properties file, apply the same changes to it as you had in the old properties file (don't just copy over the old properties file).
6. Once you have confirmed that the new server is working correctly, ensure that the production license is updated for the new server ID, as follows:
   a. Log in to https://my.atlassian.com.
   b. Locate the appropriate license.
   c. Edit the Server ID, as per the new production Server ID, and save it.
   d. Update the production license in the new server.

Once you have confirmed the availability of compatible versions, you should upgrade your plugins after successfully upgrading JIRA. This can be done via the '**Plugin Repository**' in your **Administration Console** . It is recommended that you re-index JIRA after upgrading your plugins.

Congratulations! You have completed your JIRA migration/upgrade.

Upgrading JIRA applications using a rapid upgrade method

> If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

This page describes how to upgrade the JIRA platform version 4.3.0 or later in the quickest way possible. This method can save you time since it does not require you to set up a separate test instance before you upgrade (that is, you upgrade JIRA applications "in-place"). However, it does assume that your JIRA applications are not mission critical and that users or the business won't be negatively affected when JIRA applications are unavailable during the upgrade.

You should use this method to upgrade JIRA applications if you are upgrading from the JIRA platform version 4.3.0 or later on Windows or Linux. See Upgrading JIRA applications for more information on the methods you can use to upgrade JIRA applications.

On this page:

Before you start

- **Read about the new version** - Review the release notes and upgrade notes for the version of JIRA that you are upgrading to. See our Release notes for JIRA Server. If you plan to skip a few JIRA versions during your upgrade, **we strongly recommend** that you read the upgrade guides for all major versions between your current version and the version to which you are upgrading.
- **Check your license** - Verify that your license support period is still valid.
- **Check for known issues** - Use the JIRA Knowledge Base to search for any issues in the new version that will affect you.
- **Check for compatibility**:
  - Confirm that your operating system, database, other applicable platforms and hardware still comply with the requirements for JIRA 7.1. The End of Support Announcements page also has important information regarding platform support for future versions of JIRA.
  - If you have installed JIRA plugins (i.e. not included with JIRA), verify that they will be compatible with the version of JIRA you are upgrading to. You can find a plugin's compatibility

information from the plugin's home page on the Atlassian Plugin Exchange.
- Some anti-virus or other Internet security tools may interfere with the JIRA upgrade 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 before proceeding with the JIRA upgrade.

1. Checking for customizations

Using the rapid upgrade method allows the installer to automatically perform many of the upgrade tasks for you. However, if you have made customizations to your JIRA application installation, you must migrate customized files manually to the upgraded installation. The installer checks for and migrates automatically:

- Legacy database configurations defined as a datasource within the application server (used in JIRA 4.3.x and earlier) to the new database configuration used in JIRA platform version 4.4 and later. See JIRA 4.4 Upgrade Notes for details.
- TCP port values in your existing JIRA application installation's `server.xml` file. Other configurations or customizations in this file are not migrated.
- Custom values in your existing JIRA application installation's `jira-application.properties` and `setenv.sh` / `setenv.bat` files.
  In the `setenv.sh` / `setenv.bat` file, only the following values are migrated:
  - `JVM_SUPPORT_RECOMMENDED_ARGS`
  - `JVM_MINIMUM_MEMORY`
  - `JVM_MAXIMUM_MEMORY`
  - `JIRA_MAX_PERM_SIZE`

During the upgrade process, the installer detects and notifies you of any files (other than `jira-application.properties` and `setenv.sh` / `setenv.bat`) in the `atlassian-jira` subdirectory of your existing JIRA application installation directory, which had been deleted, added or modified from a 'default' JIRA application installation. If you have made customizations to your `seraph-config.xml` file or any other file customizations in your JIRA application installation directory which are not handled by the upgrade wizard, you must migrate these to the upgraded JIRA application installation manually.

The upgrade feature also re-uses your existing JIRA application home directory so that any key data stored in this directory from your previous JIRA application installation will be retained after the JIRA application upgrade.

Please Note:

- The upgrade process requests that you conduct a backup of your database using your database's backup utilities. If your database does not support online backups, you can stop the upgrade process, shut down JIRA applications, perform your database backup and then restart the upgrade process to continue on.
- The installer automatically backs up the installation and home directories of the existing JIRA application installation. If your attachments and index files are located outside your JIRA application home directory, you must manually back up these files. These pages describe how to find out where these directories are located in your environment:
  - Your attachments directory — Refer to the Configuring file attachments page for your JIRA application version.
  - Your index directory — Refer to the Search indexing page for your JIRA application version.

2. Backing up your external database

After you launch the upgrade wizard, but before it begins the upgrade, it asks you to back up your external database. You can back up the database using your database's native backup tools, however, note the following:

- If your database's native backup tools support online backups (i.e. backups that would typically create a "snapshot" of your JIRA application database while the database is still in use), you can leave the upgrade wizard running while you perform the database backup and then continue on with the wizard after verifying that the database backup was created correctly.
- If your database's native backup tools do not allow you to perform an online backup of your JIRA application database, you should:
  1. Quit the upgrade wizard when it prompts you to back up the database.

2. Prevent users from updating your existing JIRA application data (to ensure structural consistency of your database backup) by temporarily restricting access to JIRA applications. Refer to the Preventing users from accessing JIRA applications during backups page for more information.
3. Use your database's native backup tools to perform an "offline backup" of your JIRA application database and verify that this backup was created correctly.
4. Re-start the Linux / Windows Installer to start the upgrade wizard again and continue from where you left off.

- The JIRA application 'internal' database is H2, which should be used for evaluating JIRA applications only. If you happen to accidentally use the H2 database for a production system, quit the upgrade wizard when it prompts you about the backup and use the Migrating JIRA applications to another server procedure to upgrade your JIRA applications.

⚠ **Inconsistent database backups may not restore correctly!** If you are unfamiliar with your database's native backup/restore facilities, then before proceeding, test your database backup's integrity by:

- restoring the database backup to a different (test) system, and
- connecting a test instance of your current JIRA application version to this restored database.

3. Performing the upgrade

Refer to the appropriate upgrade instructions below for your operating system:

- Windows
- Linux

Upgrading JIRA applications on Windows

1. Download the 'JIRA Windows Installer' (.exe) file (for the new JIRA application version) from the JIRA applications download page.
2. Shut down your JIRA applications.
3. Run the '.exe' file to start the upgrade wizard.
   ℹ If a Windows 7 (or Vista) 'User Account Control' dialog box asks you to allow the upgrade wizard to make changes to your computer, specify **Yes**. If you do not, the installation wizard will have restricted access to your operating system and any subsequent installation options will be limited.
4. At the 'Upgrading JIRA?' step, choose the **Upgrade an existing JIRA installation** option.
5. In the **Existing JIRA installation directory** field, specify the JIRA installation directory of your JIRA installation to be upgraded.
   ℹ The upgrade wizard will attempt to find an existing JIRA installation and use its location to pre-populate this field. However, always verify this location, particularly if you have multiple JIRA installations running on the same machine.
6. During subsequent steps of the upgrade wizard, you will be prompted to specify or do the following options:
   a. At the 'Back up JIRA directories' step, ensure the **Back up these directories** option is selected. This creates 'zip' archive file backups of your existing JIRA installation and JIRA home directories in their respective parent directory locations.
      ℹ **Please Note:**
      - **Choosing this option is strongly recommended!**
      - At this point, the upgrade wizard notes any customizations in your existing JIRA installation directory which it cannot automatically migrate to your upgraded JIRA installation. If you are informed of any files containing such customizations, please make a note of these files as you will need to manually migrate their customizations (which are not mentioned in the overview above) to your upgraded JIRA installation. One relatively common customization that the upgrade wizard cannot automatically migrate is an SSL configuration defined in the `conf/server.xml` file of the JIRA installation directory.
   b. At the 'Upgrade Check List' step, back up your external database and check that any non-bundled plugins will be compatible with your upgraded JIRA version. You may have already backed up your database (in step 2: Backing up you external database).
   c. After the 'Upgrade Check List' step, the existing JIRA installation will be shut down if it is still running. The upgrade wizard will then:
      i. Back up your existing JIRA installation.
      ii. Delete the contents of the existing JIRA Installation Directory.
      iii. Install the new version of JIRA to the existing JIRA Installation Directory.

iv. Start your new (upgraded) JIRA installation.
⚠ If you noted any files that contain customizations which must be migrated manually to your upgraded JIRA installation (above), then:
1. Stop the upgraded JIRA installation.
2. Migrate the customizations from these files into the upgraded JIRA installation directory.
3. Restart the upgraded JIRA installation.

7. At the last step of the upgrade wizard, select the option to launch the upgraded JIRA installation in a browser so you can check the upgrade.

Congratulations, you have completed upgrading your JIRA installation on Windows!

Upgrading JIRA on Linux

1. Download the appropriate 'JIRA 'Linux 64-bit / 32-bit Installer' (.bin) file that suits your operating system (for the new version of JIRA) from the JIRA Download page.
2. Shut down JIRA.
3. Open a Linux console and change directory (`cd`) to the '.bin' file's directory.
⚠ If the '.bin' file is not executable after downloading it, make it executable, for example:
`chmod a+x atlassian-jira-X.Y.bin`
(where X.Y represents your version of JIRA)
4. Execute the '.bin' file to start the upgrade wizard.
5. When prompted to choose between creating a new JIRA installation or upgrading an existing installation, choose the **Upgrade an existing JIRA installation** option.
6. Specify the JIRA installation directory of your JIRA installation to be upgraded.
ℹ The upgrade wizard will attempt to find an existing JIRA installation and will provide its location as a choice. However, always verify this location, particularly if you have multiple JIRA installations running on the same machine.
7. During subsequent steps of the upgrade wizard, you will be prompted to specify or do the following options:
    a. Choose the option to back up JIRA's directories. This creates 'zip' archive file backups of your existing JIRA installation and JIRA home directories in their respective parent directory locations.
    ℹ **Please Note:**
        - **Choosing this option is strongly recommended!**
        - At this point, the upgrade wizard notes any customizations in your existing JIRA installation directory which it cannot automatically migrate to your upgraded JIRA installation. If you are informed of any files containing such customizations, please make a note of these files as you will need to manually migrate their customizations (which are not mentioned in the overview above) to your upgraded JIRA installation. One relatively common customization that the upgrade wizard cannot automatically migrate is an SSL configuration defined in the `conf/server.xml` file of the JIRA installation directory.
    b. At the 'Upgrade Check List' step, back up your external database. You may have already backed up your database (in step 2 Backing up your external database).
    c. After the 'Upgrade Check List' step, the existing JIRA installation will be shut down if it is still running. The upgrade wizard will then:
        i. Back up your existing JIRA installation.
        ii. Delete the contents of the existing JIRA installation directory.
        iii. Install the new version of JIRA to the existing JIRA installation directory.
        iv. Starts your new (upgraded) JIRA installation.
        ⚠ If you noted any files that contain customizations which must be migrated manually to your upgraded JIRA installation (above), then:
            1. Stop the upgraded JIRA installation.
            2. Migrate the customizations from these files into the upgraded JIRA installation directory.
            3. Restart the upgraded JIRA installation.
8. The last step of the upgrade wizard provides you with a link to launch the upgraded JIRA installation in a browser, so you can check the upgrade.

Congratulations, you have completed upgrading your JIRA installation on Linux!

4. Post upgrade checks and tasks

Once you have confirmed the availability of compatible versions, you should upgrade your plugins after successfully upgrading JIRA. This can be done via the **Plugin Repository** in your Administration Console.

Congratulations! You have completed your JIRA upgrade.

Skipping major versions when upgrading JIRA applications

> If you're upgrading from a version of **JIRA earlier than 7.0**, you should consult the Migration hub. The release of JIRA 7.0 contained functionality that affects your user management, application access and log in permissions, and your JIRA installations setup, and it's very important that you understand the requirements and the implications before you upgrade. The Migration hub has all this information in one handy space.

To upgrade from early versions of JIRA to newer versions, you must upgrade to JIRA 4.4.5 before upgrading to a later version.

Follow these steps to skip major versions as you upgrade JIRA:

1. **Prepare**: Read the upgrade guides for all the major versions between your current version and the version to which you are upgrading. You can read about important changes between versions in the Important Version-Specific Upgrade Notes.
2. **Upgrade to an interim version**: Upgrade to JIRA 4.4.5 following these Upgrading JIRA with a Fallback Method instructions.
3. **Upgrade to the new version:** Upgrade to the new version of JIRA following these appropriate instructions. Use the table on Upgrading JIRA page to determine which method is appropriate for your environment.

Disabling auto-export

When upgrading JIRA, the new JIRA installation points at the old JIRA database. JIRA will automatically make any structural database modifications required to support new JIRA features.

To be safe, JIRA first tries to create an XML backup of your data at the point just before the upgrade. This would allow you to 'roll back' to the old JIRA version, should anything go wrong.

Sometimes the automatic XML backup procedure fails, often resulting from characters in the database which cannot be represented in XML — such as non-displayable control characters that have been 'cut-and-pasted' into a JIRA field.

In these circumstances, you can force the upgrade to proceed by editing your `jira-config.properties` file (in the JIRA home directory) and setting the property `jira.autoexport=false`
See Making changes to the `jira-config.properties` file for more information.

After having successfully upgraded JIRA, it is best to remove this property (or disable it with a '#') as it should no longer be required.

If you have any upgrade problems not covered here or in the upgrade documentation, please contact us — we're happy to help.

Rolling back a JIRA application upgrade

The 'roll back' procedures on this page describe how to restore your previous JIRA application version in the unlikely event that you encounter an issue with your application upgrade. Please follow the procedure below that relates to the upgrade procedure you used. Note that any data changed since the last backup will not be present after rolling back.

If you upgraded JIRA applications using the Migrating JIRA applications to another server procedure, your previous JIRA application installation should still be 'intact' (assuming you haven't deleted it) and there should not be a need to perform any 'roll back'.

Rolling back a JIRA application upgrade conducted using the upgrade wizard

Use this procedure to roll back a JIRA application upgrade conducted using the upgrade wizard.

Prior to rolling back your JIRA application upgrade, ensure that you have the following backups from

your previous application version:

- The JIRA application database (generated by your database's own backup tools).
- The JIRA application Home Directory.
- The JIRA application Installation Directory.

To roll back your JIRA application upgrade conducted using the upgrade wizard:

1. Stop the JIRA application upgrade or the upgraded JIRA server application if it is running.
2. Use your database server's tools to restore the JIRA application database backup you had created.
3. Delete the contents of the JIRA application Installation Directory.
4. Restore the backed-up JIRA application Installation Directory to the same location in the previous step.
5. Delete the contents of the JIRA application Home Directory.
6. Restore the backed-up JIRA application Home Directory to the same location in the previous step.
7. Start your JIRA application (by running the `start-jira.sh` or `start-jira.bat` file in the `bin` subdirectory of your restored JIRA application installation directory).
   On Windows based systems if your JIRA application was installed as a service, restart the `Atlassian JIRA` service from the Control Panel. The JIRA application service entry will be retained even if there is an error during upgrade in order to facilitate the rollback.

Rolling back a JIRA application upgrade conducted manually

Use this procedure to roll back a JIRA application upgrade conducted using the manual JIRA upgrade procedure (involving an 'in-place' database upgrade). The intended result of this procedure is to restore your previous JIRA installation to its original state (consisting of the restored database as well as the JIRA application Installation and Home directories in their original locations).

Prior to rolling back your JIRA application upgrade, ensure that you have the following backups from your previous application version:

- The JIRA application database (generated by your database's own backup tools).
- The JIRA application Home Directory.
- The JIRA application Installation Directory.

To roll back your JIRA application upgrade conducted manually with an 'in-place' database upgrade:

1. Stop the JIRA application upgrade or the upgraded JIRA server application if it is running.
2. Use your database server's tools to restore the JIRA application database backup you had created.
3. If you had deleted the JIRA application Installation Directory of your previous JIRA application version, restore the backed-up JIRA application Installation Directory to its original location.
4. Delete the contents of the JIRA application Home Directory.
5. Restore the backed-up JIRA application Home Directory to the same location in the previous step.
6. Start your JIRA application (by running the `start-jira.sh` or `start-jira.bat` file in the `bin` subdirectory of your restored JIRA application installation directory).

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.

If you are upgrading to a newer version of JIRA applications during the migration, please see Upgrading JIRA applications for information on the pre-requisite tasks you need to complete before upgrading.

If you're changing your operating system from Windows to Linux, or vice versa, remember that you will need to reverse the 'slashes' when required in your file paths ('/' to '\', or '\' to '/').

On this page:

1. Before you start

- **Check your license** - Verify that your license support period is still valid.
- **Check for known issues** - Use the JIRA Knowledge Base to search for any issues in the new version that will affect you.
- **Check for compatibility**:
  - Confirm that your operating system, database, other applicable platforms and hardware still comply with the requirements for JIRA applications.
  - If you have installed JIRA application add-ons (i.e. not included with JIRA applications), verify that they will be compatible. You can find a add-on's compatibility information from the the add-on's home page on the Atlassian Marketplace. You can also follow the procedure outlined here: Checking add-on compatibility with application updates to have the Universal Add-on Manager help you with this.

We strongly recommend performing your migration in a test environment first. Do not migrate your production JIRA server application until you are satisfied that your test environment upgrade has been successful.

- If you have any problems with your test environment which you cannot resolve, create an issue at our support site so that we can assist you.
- If you have any problems during the migration of your production JIRA server application, do not allow your users to start using this server. Instead:
  - Continue to use your old JIRA server application — this will help ensure that you do not lose production data.
  - Also create an issue at our support site so that we can help you resolve the problems

with your migration.

Some anti-virus or other Internet security tools may interfere with the migration 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 application migration.

2. Backing up

2.1 Stop users from updating JIRA data

During the upgrade process, you'll export JIRA's database from your existing JIRA installation (via an XML backup) and then restore this backup into a new JIRA installation. To ensure that the data in the XML backup is consistent with the latest data in the system, you must temporarily restrict access to JIRA so users can't update the data. Refer to the Preventing users from accessing JIRA applications during backups page for more information.

⚠ **Be aware!** Inconsistent XML backups cannot be restored!

2.2 Back up your database

Perform an XML backup of your existing JIRA installation's external database. For large JIRA installations, this process may require several hours to complete.

> The 'embedded database' is the H2 database supplied with JIRA for evaluation purposes only. If you accidentally use the H2 database in a production system, perform an XML backup of this database and continue on with this procedure.

2.3 Back up your JIRA home directory

1. Shut down JIRA.
2. Locate the JIRA home directory. You can find information about the location of the directory by navigating to the `<jira-application-dir>/WEB-INF/classes/jira-application.properties` file in your JIRA application installation directory. Alternatively, you can open the JIRA configuration tool to see the directory that is set as your JIRA Home.
3. Navigate to the directory specified in the configuration file and create a backup of it in another directory.
4. ✖ Delete the file <jira-home>/dbconfig.xml from the original folder as soon as the backup is complete.

2.4 Back up your attachments and index directories if located outside your JIRA home directory

If the attachments and index directories are located outside of your JIRA home directory, you must back them up separately. These pages describe how to find out where these directories are located in your implementation:

- Your attachments directory — Refer to Configuring file attachments page in the documentation for your version of JIRA.
- Your index directory — Refer to Search indexing page in the documentation for your version of JIRA.

Also refer to Backing up data for more information about backing up attachments in JIRA.

2.5 Back up your JIRA installation directory

The 'JIRA Installation Directory' is the directory into which the JIRA application files and libraries were extracted when JIRA was installed.

3. Setting up your new JIRA application installation

> If you are running a 'mission-critical' JIRA server application, we highly recommend performing the remaining steps of this guide in a test environment (e.g. using a separate test JIRA application database and a copy of your JIRA application home directory) before performing the upgrade for production use.

3.1 Install the new version of your JIRA applications

First, you must start with a fresh installation of your JIRA applications, either the current version or a newer one. If you are upgrading JIRA applications during this process, please see Upgrading JIRA applications for information on the pre-requisite tasks you need to complete before upgrading.

Download and extract the JIRA application distribution you require, to a new directory. **Do not overwrite your existing JIRA application installation. Ensure this has been shut down and install the new JIRA application version to a new location.**

3.2 Point your new JIRA application to (a copy of) your existing JIRA application home directory

If your new JIRA 7.1 installation is on a new server, **copy the backup of your existing JIRA Home Directory** from the old server to the new server before proceeding.

To set up a distribution:

1. Open the JIRA configuration tool.
2. Click the **JIRA Home** tab.
3. Update the **JIRA Home Directory** field:
    - If your JIRA 7.1 installation is on a new server, update the **JIRA Home Directory** field to the path of your copied JIRA Home directory.
    - If your JIRA 7.1 installation is on the same server, update the **JIRA Home Directory** field to the path of your **existing** JIRA Home directory.
      For more information about this directory, see JIRA home directory.

You can also set your JIRA Home Directory's location by defining an operating system environment variable `JIRA_HOME`. This value of this variable takes precedence over the value of the `jira.home` property in the `jira-application.properties` file in your JIRA installation directory. See Setting your JIRA home directory for details.

3.3 Connect the new version of your JIRA application to a new, empty database

Create a new, empty database that your new JIRA installation will use to store its data.

Follow the appropriate '**Connecting JIRA to...**' instructions for your database from stage 2, although from stage 4 of that procedure, be aware of the yellow note below:

- Connecting JIRA to PostgreSQL
- Connecting JIRA to MySQL
- Connecting JIRA to Oracle
- Connecting JIRA to SQL Server 2008

> If you are using a database (called **jiradb**, for example) with your existing JIRA installation and the database for your new JIRA installation is running on the same machine or database server, create your new database with a different name (e.g. something intuitive like **jiradb_440** for JIRA 4.4.0). However, ensure the new database has identical access permissions to the old JIRA database. Consult your database administrator if you need assistance with this.

You do not need to create a new database if you are using the embedded H2 database.

3.4 Migrate your existing JIRA application configurations over to your new installation

If you have modified properties in configuration files of your existing JIRA installation, make the same modifications in your new JIRA installation. However, because the properties in the configuration files may have changed between versions, you cannot simply copy the configuration files from your existing installation and replace the equivalent files in the new installation.

For each file you have modified in your existing JIRA installation, you need to **manually edit each equivalent file in your new JIRA installation and re-apply your modifications**. If a file is not present in your new JIRA installation (for example, `osuser.xml` in recent JIRA versions), then simply copy that file over to your new JIRA installation.

The table below lists the most commonly modified files and their locations within your JIRA Installation Directory:

File	Location in 'recommended' (formerly 'Standalone') JIRA distributions	Description
jira-application.properties	`atlassian-jira/WEB-INF/classes`	Location of the JIRA Home Directory and Advanced JIRA Configuration in JIRA 4.3.x and earlier. Any custom property values defined in the `jira-application.properties` file of your existing JIRA 4.3.x (or earlier) installation must be migrated across to the `jira-application.properties` file of your new JIRA 7.1 installation *before you start your new JIRA installation*. Upon starting your new JIRA installation, any custom property values in the `jira-application.properties` file will automatically be migrated across to either the JIRA database or jira-config.properties file. `jira.home` is the only property of the `jira-application.properties` file subsequently used by JIRA.
setenv.bat (Windows) or setenv.sh (Linux)	`bin`	Increasing JIRA Memory
osuser.xml (not required if upgrading from JIRA 4.3.0 or later)	`atlassian-jira/WEB-INF/classes`	Modified if you have integrated LDAP with JIRA, integrated Crowd with JIRA, or if you are using a custom form of external user management or user authentication.
seraph-config.xml	`atlassian-jira/WEB-INF/classes`	Modified if you have integrated Crowd with JIRA.
server.xml	`conf`	Modified in the following situations: • If you had previously configured JIRA's TCP ports differently from their defaults. • If you had implemented SSL. • When connecting JIRA to a database in JIRA 4.3.x and earlier.

The version-specific upgrade notes contain details on properties which may have changed in these commonly modified files.

In addition to the files above, you should also consider and/or perform the following configurations as part of the upgrade process:

- **Using JIRA with Atlassian's Crowd?** — If you are using Crowd with JIRA, configure your new JIRA to talk to Crowd as described in Integrating Crowd with JIRA.
  - Remember to configure Crowd to grant JIRA's new hostname/IP access.
- **Allocating additional memory to JIRA** — If you had previously allocated additional memory to JIRA, do the same for your new JIRA instance. For more information refer to Increasing JIRA memory.
- **Plugins** — For any plugins that you had installed in your old JIRA, download the plugin version for

your new version of JIRA from the http://plugins.atlassian.com site.

- **Character encoding** — Ensure that character encoding (i.e. locale) is the same on the new and old locations. Your new version of JIRA may not function correctly if attachments are moved between two system with incompatible encoding.
- **Customisations** — If you had made any customisations (code, templates or configuration files), copy over compatible versions of these changes to the new JIRA. (The developers within your organisation who made the customisations to your old version will need to build and test equivalent changes for the new version, and provide you with the files to copy to your upgraded JIRA installation.)
- **(Optional) Running JIRA on a different port** — If your new JIRA is installed on the same machine as your old JIRA, you may wish to make sure it runs on a different port (in case you ever need to restart your old JIRA). See Changing JIRA's TCP ports for details.

3.5 Start your new JIRA application version

1. Verify that your old JIRA installation is shut down — if this JIRA server is still operating, shut it down.
2. Start up your new version of JIRA by follow the Starting JIRA instructions.

> **Do not restart your old JIRA installation...**
> If your new JIRA 7.1 installation is on the same server as your old one, it may still be configured to use the same JIRA Home directory as your new JIRA installation. Running two separate JIRA installations which share a common JIRA Home directory can lead to serious data corruption.
>
> Nevertheless, we recommend that you do not delete any aspect (or backed up component) of your old JIRA installation, until you are satisfied that your upgraded JIRA installation is functioning as expected.

3.6 Import your old data into your new JIRA applications

After you have started your new JIRA installation you should be logged in to your JIRA application, and ready to import the data from your old instance into the new instance. You will need the backup file of data from your old JIRA that you created earlier in these instructions (above).

To import your old JIRA data into your new JIRA:

1. Log in as a user with the 'JIRA System Administrators' global permission.
2. Select **Administration > System > Import & Export > Restore System** (tab) to open the 'Restore JIRA data from Backup' page.
3. In the **File name** field, specify the XML backup file you created previously during the export process (above). That zipped file should contain two xml files: `activeobjects.xml` and `entities.xml`. Both of these files must be included in the zipped file for the import process to work.
4. Restore the attachments directory that you backed up previously, into the attachments directory of your new JIRA. (See Restoring data.)
   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. Access JIRA via your web browser again and log in using a username from your previous JIRA installation.
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.

4. Post migration checks and tasks

It is strongly recommended that you perform the following checks and tasks after you have started your new JIRA application instance:

1. Check your server logs for error messages, even if JIRA applications appear to be running correctly. If there are any errors there that you cannot resolve, create a support case, attach your log file, and we will advise you on the errors.
2. If you were previously using External User Management, enable it in the new JIRA application instance.
3. If you changed machines when upgrading, change the paths to the indexes, attachments and backup directories, from within your application's **JIRA Administration** section.
4. Enable email, if you disabled it during testing.
5. If you migrated any customizations from your old to new JIRA applications, ensure that they are tested

thoroughly.

a. If you had downloaded plugins for the new JIRA application versions, install the downloaded JAR file(s) in your new version and carry out any other required installation for the plugin.
b. If the plugin has a properties file, apply the same changes to it as you had in the old properties file (don't just copy over the old properties file).

Congratulations! You have completed your JIRA application migration/upgrade.

See Also

Disabling auto-export
Restoring data
Upgrading JIRA applications
Switching databases

Migrating from JIRA Cloud to Server applications

This page is for people who are currently using a JIRA Cloud application site and wish to move to a JIRA application instance that is hosted on their own servers. If you want to move a project, not your entire site, then see Restoring a project from backup (note, the instructions on that page take into account the version mismatch between JIRA Server and Cloud applications).

On this page:

Summary

You will need to download and install the latest production release of JIRA server applications. Please check the JIRA application download page for the latest version. Once you have installed the latest production version, you then move your data from your hosted JIRA cloud application site into your new JIRA server application instance.

For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Before you begin

JIRA cloud applications are regularly updated with the absolute latest features and improvements — they are essentially running on a later version than the latest downloadable version of JIRA applications. If you want to migrate from JIRA cloud applications to JIRA server applications, please be aware of the following information before you begin:

Known issues

Read the following known issues before you start your migration:

- **Tempo data loss**: If you have made changes to the Tempo scheduler in JIRA cloud applications after March 24 (JIRA 6.3-OD-01 upgrade), these changes *will be lost* when you migrate to JIRA server applications. This is due to changes that we have made to the scheduler data in JIRA cloud applications that are not available in JIRA 6.2.x. To address this issue, choose one of the following options:
  - Do not make changes to the Tempo scheduler in JIRA cloud applications after March 24. Wait until you have migrated to JIRA server applications before making changes.
  - Keep a record of any changes that you make to the Tempo scheduler in JIRA cloud applications after March 24 and make those same changes in Tempo after you migrate to JIRA server applications.
  - (Not for production sites) Migrate to a JIRA 6.3 EAP instance.

Feature loss

If you migrate from JIRA cloud to server applications, you will likely find a few features missing. This is because we have introduced features from the upcoming JIRA server application version into JIRA cloud

applications. For example, the latest JIRA server application release is JIRA 6.2.x. JIRA 6.3 is currently under development. Some of the JIRA 6.3 features have been made available in JIRA cloud applications, but will not become available for JIRA server applications until the final JIRA 6.3 version is released.

JIRA application licenses

Your Atlassian Cloud license cannot be used in an instance installed from JIRA server applications. You will 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. You will need to contact your vendor for the licenses for all other third-party plugins.

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), please 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 instan ce hosted on your own servers, you will also lose a number of 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.

Instructions

1. Generate a backup of your JIRA cloud application data

1. Log in to your JIRA cloud application site as an administrator.
2. Generate an XML export from your JIRA cloud application data by following the instructions in Exporti ng issues. This includes instructions on how to back up your attachments. Note, the export process will strip your Cloud license and plugin licenses out of the XML, they will not be available when importing into the installed instance, but they will remain available in JIRA cloud applications.
3. Download the backup file from your Cloud WebDAV directory (also described in Exporting issues).

2. Install JIRA server applications

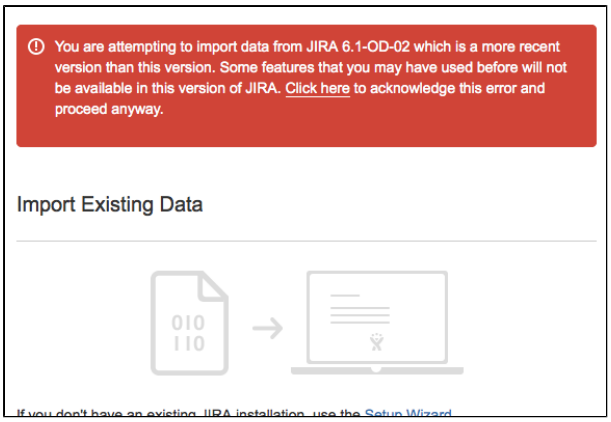
You must use the latest version of JIRA server applications.

1. Download the latest version.
2. Follow the instructions in Installing JIRA applications *until you are instructed to run the setup wizard*, then see step 3 below.

3. Import your JIRA cloud application data into your JIRA server application

Follow the instructions in on the setup wizard until you have configured a database (described in step 1 of Running the setup wizard). If you already have some data in your JIRA application installation, this step will overwrite it.

In step 2 of the setup wizard (Application Properties), you will be asked whether you have existing data. Click **import your existing data** and follow the instructions to import the JIRA Cloud application backup that you generated earlier. Note, you may see the following message. This is a warning only, your import will still work.



See Restoring data for more information.

> For instances with large backups (2Gigabyte and up), we recommend importing the attachments separate from the Issue and user data. To do this:
>
> 1. Unzip the backup file.
> 2. Compress the `activeobjects.xml` and the `entities.xml` files only.
> 3. Import that compressed file.

4. Copy across attachments

If you included your attachments in the export:

1. Extract the backup file that was downloaded.
2. Copy across the contents of the attachments folder to the `$JIRA_HOME/data/attachments` directory for JIRA Server.

5. Change the system administrator 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 are installed on your JIRA cloud application site

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 plugins (add-ons)

For each plugin that you noted in the previous step, install it in your JIRA applications. You must install a version of the plugin that is *equal to or later than* the plugin version that was installed in the cloud. 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.

> The Support Tools Plugin that comes bundled with JIRA will get disabled after completion of the migration. Look it up under "All add-ons" in the Manage add-ons section to re-enable it.

Version matrix for imports

The following table tells you which version of the JIRA server platform to use when migrating from JIRA cloud applications. The version number is dependent on when you exported your data from JIRA cloud applications.

We recommend that you use the latest JIRA version unless otherwise specified below. ***Only use the versions listed below if you cannot use the latest JIRA platform version.***

Date when export was made	Version of JIRA server platform to use
Prior to 3 Dec 2012	Contact support for assistance
3 Dec 2012 — 16 Dec 2012	5.2.1
17 Dec 2012 — 20 Jan 2013	5.2.2
21 Jan 2013 — 6 Feb 2013	5.2.5
7 Feb 2013 — current	Use the latest version available

Establishing staging server environments for JIRA applications

On this page:

This document describes best practices for an enterprise environment setup for JIRA applications:

- Best-practice recommendations for procedural governance around rolling out changes
- Recommendations for development / staging / production architecture
- Technical steps for how to deploy non-production servers

Assumptions:

- For this document we are assuming that as an administrator, you would rather script changes. Therefore we have omitted UI-based changes or separate tools such as the database configuration tool in favor of specifying file system locations.

Please note:

- The procedures described in this document will work with JIRA platform version 4.0 and later.
- Please read the entire document before bringing a staging server live. There are risks associated with connecting to production instances that require attention, which are called out in the document.

1. Architecture strategy

Often, systems administration teams will have an established architecture for enterprise applications, including staging environments and failover setups. We offer these recommendations in this section not to supplant or change those company-wide strategies, but rather to help illustrate what some of the considerations will be with Atlassian applications in staging environments.

Definitions

For the purpose of this document, we'll assume the following definitions:

- **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.

Recommendation

If Atlassian applications are critical systems, we recommend this 3-tier strategy for development, staging, and production.

- The staging environment is primarily for system administrators to test changes and upgrades before going into production.
- The development environment is for different business units to test changes on their own, before requesting a production rollout.

2. Governance strategy

In addition to an architecture, we also recommend establishing a governance strategy for changes. This could include:

- Create a strategy for deploying and testing plugin installation requests. Note that some plugins that are extremely useful in some environments are not appropriate for high-volume critical systems.
- Publish a timeline for refreshing the development environment, so users know when to remove their changes.
- Set up a source control repository to house any file system changes, so you can track when changes were made and by whom, historically. If you don't have one already established, Bitbucket is an option. In addition to file system customizations, record your procedures for upgrades, staging refresh (see below) and any other scripted changesets in your source control.
  Tip: JIRA applications have a tool to manage any changes in your installation. Check the System Information page in the UI for "modified files." This will tell you which files have been customized in your installation directory.
- For changes such as creating new workflows (that require administrative access), you have two options:
  1. Create an administrative user which has temporary access to administrative functions, on a per-request basis. Add this user to the appropriate groups so they can perform the necessary administrative functions. When the user has completed their administrative functions, remove the user from these groups.
  2. Keep your development server devoid of production data and give more administrative privilege on this server. Require end-users to document specific workflow or scheme setups, then repeat these steps in production.

3. How to refresh a staging server

We're assuming that you have an existing staging installation. If not, you can use these instructions to set up your staging environment now.

> Take care to make sure your staging server setup does not interfere with your production environment. Read the tips below before launching your staging (or development) server.

3.1 Create a complete production backup

1. **Back up your home directory.** See Setting your JIRA Home Directory for the location of your production home directory.
   Back up your production attachments and index directories if located outside your JIRA applications home directory. If you're unsure where these are stored, refer to Configuring File Attachments and Search Indexing to determine these locations.
   Refer to Backing Up Data for more information about backing up attachments in JIRA applications.
2. **Back up your installation directory.** The 'JIRA Installation Directory' is the directory into which the JIRA application files and libraries were extracted when the JIRA applications were installed.
3. **Back up your production database.** Use your native backup tools to take a snapshot of your production database.

3.2 Copy your complete production backup to a staging environment

1. **Shut down your staging server.**
2. **Restore your installation and home directories on the staging server.**
3. **Point the newly restored installation directory to the newly restored JIRA application home directory.**
   a. Edit the `jira-application.properties` file located within the `<jira-application-di`

r>/WEB-INF/classes subdirectory of your new Installation Directory.
   b. Update the jira.home property in this file to the path of the new JIRA application home directory to the path of your copied JIRA applications home directory.
   c. Save your updated jira-application.properties file.

   You can also set your JIRA application home Directory's location by defining an operating system environment variable JIRA_HOME. This value of this variable takes precedence over the value of the jira.home property in the jira-application.properties file in your JIRA installation directory. See Setting your JIRA Home Directory for details.
4. **Restore your database to a staging database.**
   ⚠ If you are using a database (called **jiradb** for example) with your existing JIRA application installation and the database for your new JIRA application installation is running on the same machine or database server, create your new database with a different name (e.g. something intuitive like **jiradb_440** for JIRA 4.4.0). Oracle **does not** support schema names with periods or underscores. Ensure the new database has identical access permissions to the old JIRA application database.

3.3 Modify your staging environment for the unique configurations

1. Configure your database connection to point to your staging database. Edit the dbconfig.xml file at the root of your JIRA home directory, or the datasource in <jira-install>/conf/server.xml for older versions.
   ⚠ This is extremely important! Make sure your staging environment is not pointing to your production database.
2. **There are two options to handling email:**
   a. **Disable mail on your staging server.** If you need to perform some initial tests on your new JIRA application installation, you can disable its email access to prevent unintended emails being sent. You can leave emails on, if you're wanting to test email functionality. If you choose to do keep emails enabled, watch particularly for:
      i. Create or comment handlers, which can pull mail from your production mail servers. You can disable these from **Administration > Advanced > Services**, or delete them from 'serviceconfig' table in the database.
      ii. Filter subscriptions, as your users will receive notifications for filters they're subscribed to. Delete filter subscriptions from the 'filtersubscription' table in the database.
      iii. Notifications on tickets that are updated. For these, dissociate any notification schemes to projects you wish to test without email notifications.
   b. **Keep email enabled** and configure your staging instance to test email:
      i. See the guide here: How to Prepare a Development Server's Mail Configuration

3.4 Restart your staging server

You are now ready to restart your server. Once you've restarted, perform the following checks to verify you've done the above steps safely:

1. **Ensure the database is not pointing to production**. To check this, see Viewing your System Information. Check the 'Database URL' to ensure it's pointing to the right place.
2. **Ensure emails are disabled or configured for dev server**. Also when viewing your system information, check the 'JVM Input Arguments' for the line 'atlassian.mail.senddisabled'. If you configured the email for a dev server as described above, this line will not be there.

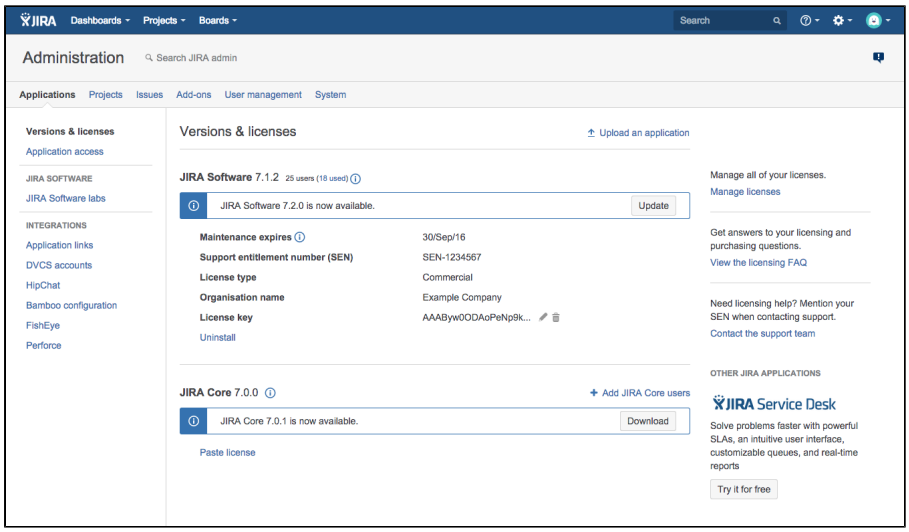
3.5 Post-startup modifications

1. **Modify the instance colors.** See Configuring the look and feel of your JIRA applications. This is a good practice for users to identify that they're on the staging server.
2. **Modify the instance base URL.** See Configuring JIRA options and change the Instance URL to the staging URL.
3. **Consider the URL whitelist.** You may wish to change some of the approved URLs. See Configuring the whitelist.
4. **Apply a development license.** See our licensing FAQ to generate a license for the staging server. Refer to Licensing your JIRA applications to apply it.
5. **Reconfigure applinks.** If you are connecting to other servers via applinks, you'll need to change the server ID for those instances.
   ⚠ If you leave applinks in place, it's possible to have your production instance point back to the staging server, if a link is generated.
   a. Confluence: How to change the server ID of Confluence
   b. JIRA applications: Changing Server ID for Test Installations

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

- 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.

Restricted functions in JIRA Cloud applications

Generally speaking, all functions performed by the JIRA System administrator in JIRA server applications are restricted in JIRA Cloud applications. The following table lists all of the functions that are restricted in JIRA Cloud applications, even to users with administrator permissions. The table below also lists whether the function can be configured by Atlassian Technical Support on request. If not, then it is unavailable for your configuration in JIRA Cloud applications.

Restricted function	Configurable on request?	Notes
General Configuration • External user management • JIRA as a user server	No	A limited number of specific General Configuration and Advanced Settings options are available in JIRA Cloud applications. External user management is not available.
System Administrator Permissions	No	The JIRA 'System Administrator' global permission is not available in JIRA Cloud applications. The 'JIRA administrators' permission is still available. You can see a list of differences between the two administration types her e.
Configuring an outgoing (SMTP) Mail server	No	Atlassian Cloud comes with an internal SMTP server configured to send notifications. The prefix is not configurable.
Incoming Mail Servers • Configuring File system messages	No	POP/IMAP Mail Servers are configurable in JIRA Cloud applications but File system messages are not. Note that you can create issues via email using those POP/IMAP servers.
Changing project key pattern	No	
Configuring listeners	No	Please see **JRA-31598** - Allow the use of Listeners in a future version of Cloud Applications **CLOSED** for the status of this functionality request.
Configuring services	No	
Customizing source files	NO	

Customizing email content	NO	The procedure for customizing email content sent in notification messages requires editing Velocity files within the JIRA web app. This makes it a special case of "customizing source files". See CLOUD-1791 - Customize notification email template CLOSED and JRA-7266 - Create an Email Template Editor in the UI to Replace Editing Velocity Templates OPEN for suggestions to allow customization of email contents through the web UI.
Changing the index path	No	
Running the integrity checker	No	
Configuring logging and profiling information	No	In Labs we have an Audit feature available. The feature can be enabled at the following location: https://<instance>/auditing/settings
Accessing the scheduler	No	
Importing data	Yes	Please read Import and export data for information on supported imports for JIRA Cloud. Imports from certain non-Atlassian products (e.g. Bugzilla FogBugz for Your Server) or single project imports can be performed using a local evaluation copy of JIRA to create a compatible backup that can be imported into JIRA Cloud.
Importing XML workflows into JIRA	No	It is possible to Import Workflow from Marketplace only.
Importing JIRA projects	No	Restoring a project from backup does not apply to JIRA Cloud. Please see JRA-31806 - Provide support for "Single Project" imports into JIRA Cloud OPEN for the related feature request status, and Project Import From JIRA Server to JIRA Cloud for workaround.
Add-ons	Yes	Please see Add-ons and plugins for the list of supported add-ons.
Disabling attachments, or setting the attachment path	No	Attachments enabled by default
Running Jelly scripts	No	Please see CLOUD-1439 - Enable Jelly scripting CLOSED for the related feature request status.
Configuring LDAP integration	No	
Configuring trusted applications	No	

Configuring application links	Yes	
Connecting local Bamboo installations	Yes	
Accessing license details	No	
Modifying SysAdmin users & attributes	No	
Deactivating users	No	It is not possible to 'deactivate' a user in JIRA Cloud applications. See Ma nage application access for information on how restrict a user's access to one or more applications in Atlassian Cloud.
Renaming users	Yes	It is possible to rename a JIRA application and/or Confluence user in Cloud User Management. It is not possible to rename Bamboo users.
Remote API (REST)	Yes	For more information, see JIRA REST APIs.
Installing intermediate SSL certificates	No	It is not possible to install intermediate certificates in JIRA Cloud applications. The potential impact might be that you will run into problems when adding secure POP/IMAP mail servers that have certificates signed by non-root CAs. The workaround is to obtain a certificate signed by a trusted root Certification Authority (CA). The full list is here.
HTML and JavaScript in issue fields	No	This feature creates an XSS vulnerability that could allow malicious users to gain system administrator permissions and use the Cloud infrastructure for nefarious purposes. Wiki markup is still available.
Text Gadget	No	This gadget creates an XSS vulnerability as it can contain arbitrary HTML. Please see JRA-29848 - JIRA Cloud text gadget CLOSED .
Uploading custom icons for issue type and priority	No	This is tracked at JRA-21246 - Allow upload of custom issue type icons to JIRA Cloud RESOLVED .
Default session timeout	No	This is tracked at JRA-34258 - Ability to configure the session timeout for OnDemand OPEN .
External gadgets	No	
Change number of workflow transition buttons	Yes	Create a support request with the number of transition buttons you want, and a 15-minute maintenance window (with 24 hours of notice) and Atlassian Support will configure it for you. (We'll be working from Advanced workflow configuration).

Configuring OAuth authentication	No	You need the **'JIRA System Administrators'** global permission to do that. 3 Legged OAuth is automatically configured for Application Links. 2 Legged OAuth with impersonation is not available. 2 Legged OAuth without impersonation is available and required for the Development Panel. Use this type to connect to Bitbucket Server and Bamboo Server.
Configuring Secure Administrator Sessions	No	

Layout and design

The layout and design of your JIRA applications can be customised 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':

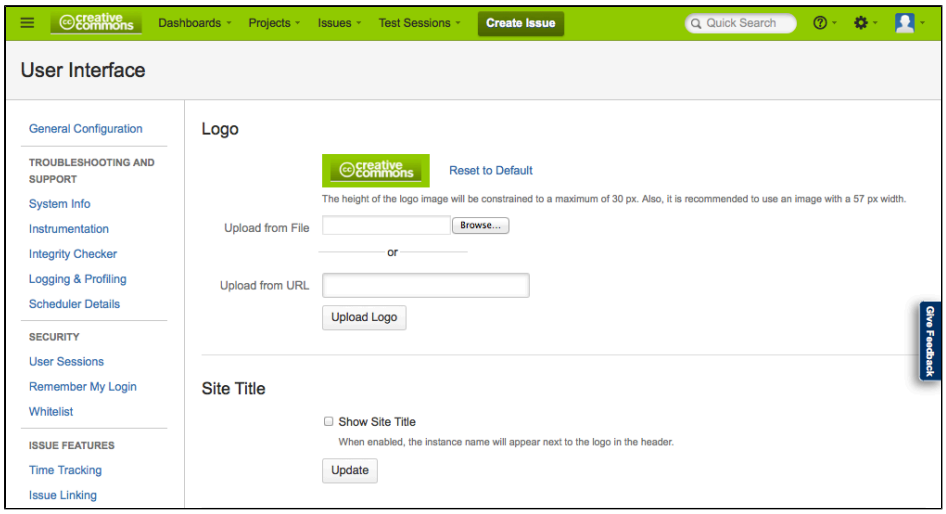
The following pages provide you with more information on setting up your JIRA applications so you can get the most out of them!

- 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

This page tells you how to customize your JIRA installation to match your company's environment. One of the easiest things you can do to get started is to update your JIRA color scheme to match your company's logo (shown below).



- **Upload from File –** click **Browse** to search for and upload a new image for the logo.
- **Upload fromURL** –use one of the following conventions:

- A URL beginning with '`http://`' or '`https://`' is treated by JIRA as an absolute URL/path.
- A URL beginning with a forward slash '`/`' is treated as a path relative to the `<jira-application-dir>` subdirectory of your JIRA application installation directory.

⚠ **Note:** For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

If the JIRA logo does not appear after changing it to a custom one, ensure that the URL specified uses the correct case as this may be case-sensitive.

If you don't like the change, simply click **Undo**.

On this page:

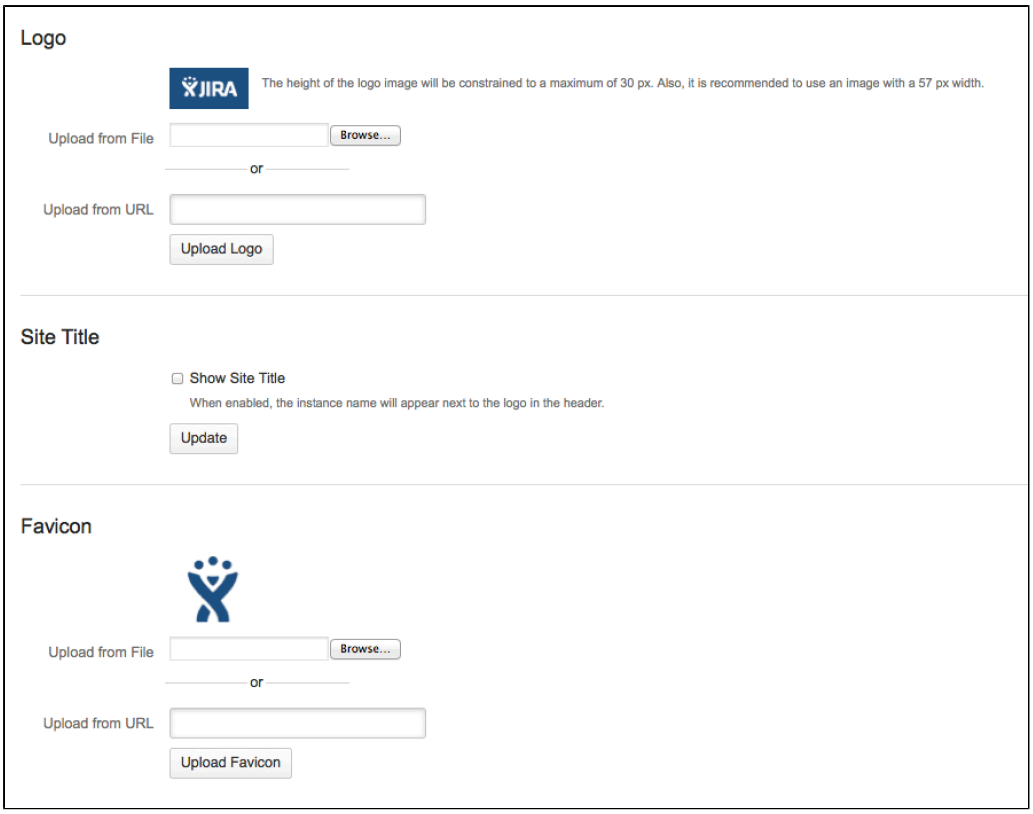
Related pages:

- Configuring the default issue navigator
- Configuring the default dashboard

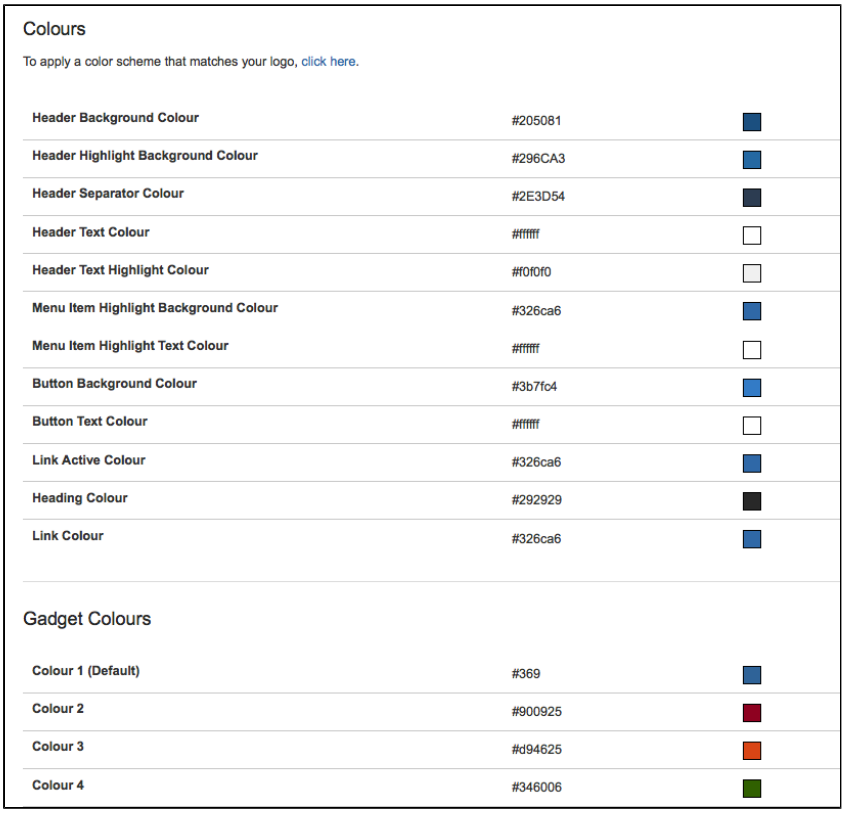
Look and feel configuration

You can easily customize **JIRA's look and feel** to suit your needs**:**

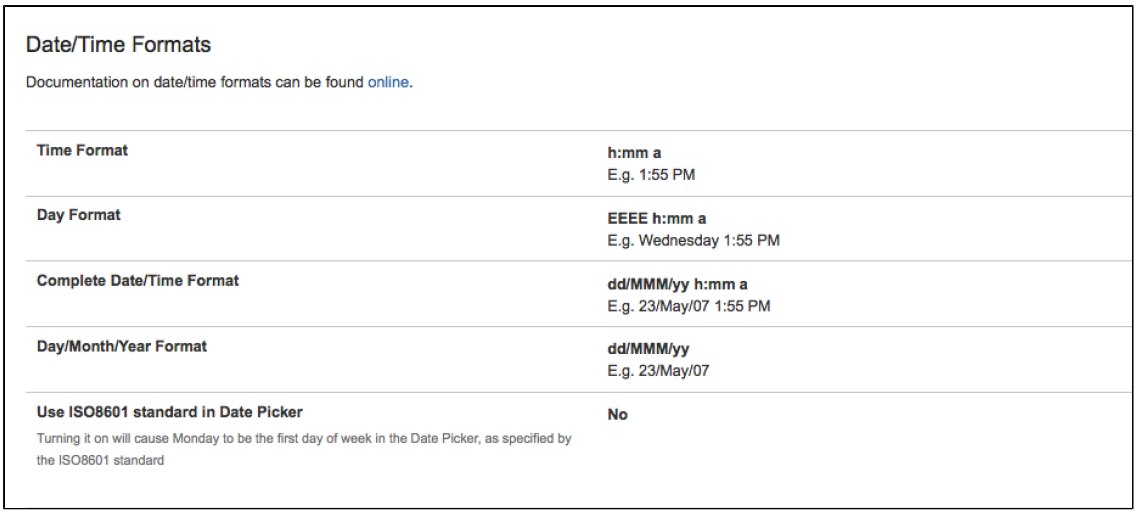
1. Choose  **> System**.
2. Select **User Interface > Look and Feel**.
3. The **Look and Feel** configuration page will be displayed as follows:
   Screenshot: Look and Feel Configuration



Logo, Site Title and Favicon



Colours and Gadget Colours



Day/Time Formats

4. To edit the logo, see the next section on Logo and Favicon.
5. To edit the colors, click on the individual colors and edit them directly. For more information, see the section below on Editing colors.

Logo and Favicon

The logo appears in the top left corner of every JIRA page while the favicon appears typically to the left of your browser's URL field and on browser tabs displaying a page on your JIRA site. You can easily replace the default JIRA logo and/or favicon with an image of your choice.

Element	Sizing information
Logo	The logo's height must be constrained to a maximum of 30px. It is also recommended to use a 57px width image.
Favicon	The favicon must have PNG file format, it should be 32x32 pixels, 71x71 DPI and have 8 bit color depth for best results.

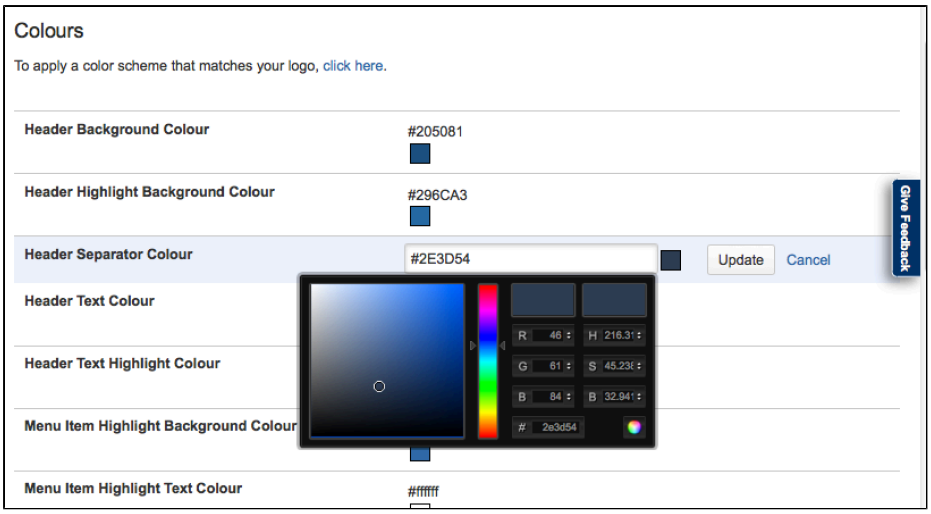
Colors

The following options control the appearance of the entire JIRA user interface.

Editing colors

To edit the colors, click on the individual colors and follow this procedure.

1. Click on the color box for an element.
2. This opens up the color display where you can create customized colors or enter specific color values:



3. To save your changes, click **Update**.
4. If you are unhappy with a color change, click the **Revert** button that displays in the row where you've made the change:

Usage Notes

- The colors you specify for each of the following options can be anything that is valid for both a font tag, and a stylesheet's 'color:' attribute.
- When specifying a color, you can use the pop-up color chooser, or specify your own (eg. *'#FFFFFF', 'red'*) .
- To return to the original color scheme, just clear any values that you have set.

Gadget colors

These seven colors are the seven options from which users can select when changing the color of a gadget's frame on their JIRA dashboard. color 1 is the default frame color for newly-added gadgets.

Please note:

- The colors you specify for each of the eight options can be anything that is valid for both a font tag, and a stylesheet's 'color:' attribute.
- When specifying a color, you can use the pop-up color chooser, or specify your own (eg. '#FFFFFF', 'red').
- To return to the original color scheme, just clear any values that you have set.

Date/Time Formats

The **Look and Feel** page allows you to 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.

When you are not in edit mode on the 'Look and Feel' page, the examples in the rightmost column of the **Date/Time Formats** section show you how the various formats will appear in JIRA.

Relative time is used in date/time formats

Issue date/time fields show a relative instead of absolute date/time format (for example: "Yesterday" 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.If you want to switch off this format, set the `jira.lf.date.relativize` application property to 'false'. See Advanced JIRA application configuration and Disable relative dates in JIRA applications for more information.

Configuring date picker formats

JIRA system administratorscan configure the format of date pickers used throughout the JIRA user interface via options on the Advanced Settings page.

Be aware that these options are different from the **Date/Time Formats** configuration options on the **Look and Feel** page, which only customize JIRA's presentation of times and dates to users.

The date or date/time formats for date pickers are defined by a pair of properties (one for Java and the other for JavaScript). The two properties in this Java/JavaScript pair must match in order for the date (or date/time) picker they define to function correctly.

- For Java formats, specify date/time formats based on the Java SimpleDateFormat.
- For JavaScript formats, specify date/time formats based on the Unix date format.

Here are some example US-based date configurations:

Preferred Date	Value of the `jira.date.picker.java.format` property	Value of the `jira.date.picker.javascript.format` property	Comme
2010-10-01	yyyy-MM-dd	%Y-%m-%d	ISO 860 format
Oct/1/10	`MMM/d/yy`	%b/%e/%y	
10/01/10	`MM/dd/yy`	%m/%d/%y	
Oct 1, 2010	`MMM d, yyyy`	%b %e, %Y	
10/01/2010	`MM/dd/yyyy`	%m/%d/%Y	

Here are some examples of date/time configurations:

Preferred Date/Time	Value of the `jira.date.time.picker.java.format` property	Value of the `jira.date.time.picker.javascript.format` property
2010-10-15 08:50	yyyy-MM-dd HH:mm	%Y-%m-%d %H:%M
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

On this page:
- Configuring an announcement banner
- Banner visibility mode

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.

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.

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

JIRA provides the ability to display summary information about project/issue data on the dashboard, through the use of 'gadgets'. Each gadget can be configured to display project and issue details relevant to particular users. Gadgets can be added to the dashboard — providing a central location for quick access to this information.

> **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, please 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	The **Activity Stream** gadget displays a summary of your recent activity.
Administration Gadget	The **Administration** (**Guide for JIRA Administrators**) gadget displays checklist of common administration tasks and links to administrative functions and documentation.
Assigned To Me Gadget	The **Assigned To Me** gadget displays all open issues in all projects assigned to the current user viewing the dashboard.
Average Age Gadget	The **Average Age** gadget displays a bar chart showing the average number of days that issues have been unresolved.
Bamboo Charts Gadget *	The **Bamboo Charts** gadget displays various charts and plan statistics from a particular Bamboo server.
Bamboo Plan Summary Chart Gadget *	The **Bamboo Plan Summary** gadget displays a graphical summary of a build plan.
Bamboo Plans Gadget *	The **Bamboo Plans** gadget displays a list of all plans on a Bamboo server, and each plan's current status.
Bugzilla ID Search Gadget	The **Bugzilla ID Search** gadget allows the user to search all JIRA issues for references to Bugzilla IDs.
Calendar Gadget *	The **Issue 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 *	The **Clover Coverage** gadget displays the Clover coverage of plans from a particular Bamboo server.
Created vs Resolved Gadget	The **Created vs Resolved** gadget displays a difference chart showing the issues created vs resolved over a given period.

Crucible Charts Gadget *	The **Crucible Charts** gadget displays various charts showing statistical summaries of code reviews.
Favorite Filters Gadget	The **Favorite Filters** gadget displays a list of all the issue filters that have currently been added by you as a favorite filter.
Filter Results Gadget	The **Filter Results** gadget displays the results of a specified issue filter.
FishEye Charts Gadget *	The **FishEye Charts** gadget displays two charts showing showing statistics about a given sourcecode repository.
FishEye Recent Changesets Gadget *	The **FishEye Recent Changesets** gadget displays a number of recent changesets from a FishEye repository.
In Progress Gadget	The **In Progress** gadget displays all issues that are currently in progress and assigned to the current user viewing the dashboard.
Introduction Gadget	The **Introduction** gadget displays a configurable introduction message on the dashboard.
Issue Statistics Gadget	The **Issue Statistics** gadget displays the collection of issues returned from a specified filter, broken down by a specified field.
Pie Chart Gadget	The **Pie Chart** gadget displays issues from a project or issue filter, grouped by a statistic type, in pie-chart format. The issues can be grouped by any statistic type (e.g. Status, Priority, Assignee, etc).
Projects Gadget	The **Projects** gadget provides information and various filters related to a specified project(s).
Quick Links Gadget	The **Quick Links** gadget displays a number of useful links to issues associated with the current user.
Recently Created Issues Gadget	The **Recently Created Issues** gadget displays a bar chart showing the rate at which issues are being created, as well as how many of those created issues are resolved.
Resolution Time Gadget	The **Resolution Time** gadget displays a bar chart showing the average resolution time (in days) of resolved issues.
Road Map Gadget	The **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 *	The **Text** gadget displays a configurable HTML text on the dashboard.
Time Since Issues Gadget	The **Time Since Issues** gadget displays a bar chart showing 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	The **Two Dimensional Filter Statistics** gadget displays statistical data based on a specified filter in a configurable table format.

Voted Gadget	The **Voted Issues** gadget shows issues for which you have voted.
Watched Gadget	The **Watched Issues** gadget shows issues which you are watching.

See the big list of all Atlassian gadgets for more ideas.

*This gadget will only be available if you have installed/configured the relevant plugin.

Extension gadgets

Other gadgets are available as plugins on the Atlassian Marketplace. If you wish to you use these plugins, you need to first install them (using the instructions provided with each plugin) then enable them.

Creating new gadgets

New gadgets can be created 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:

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.streams.streams-jira-plugin:activitystream-gadget/gadgets/activitystream-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.

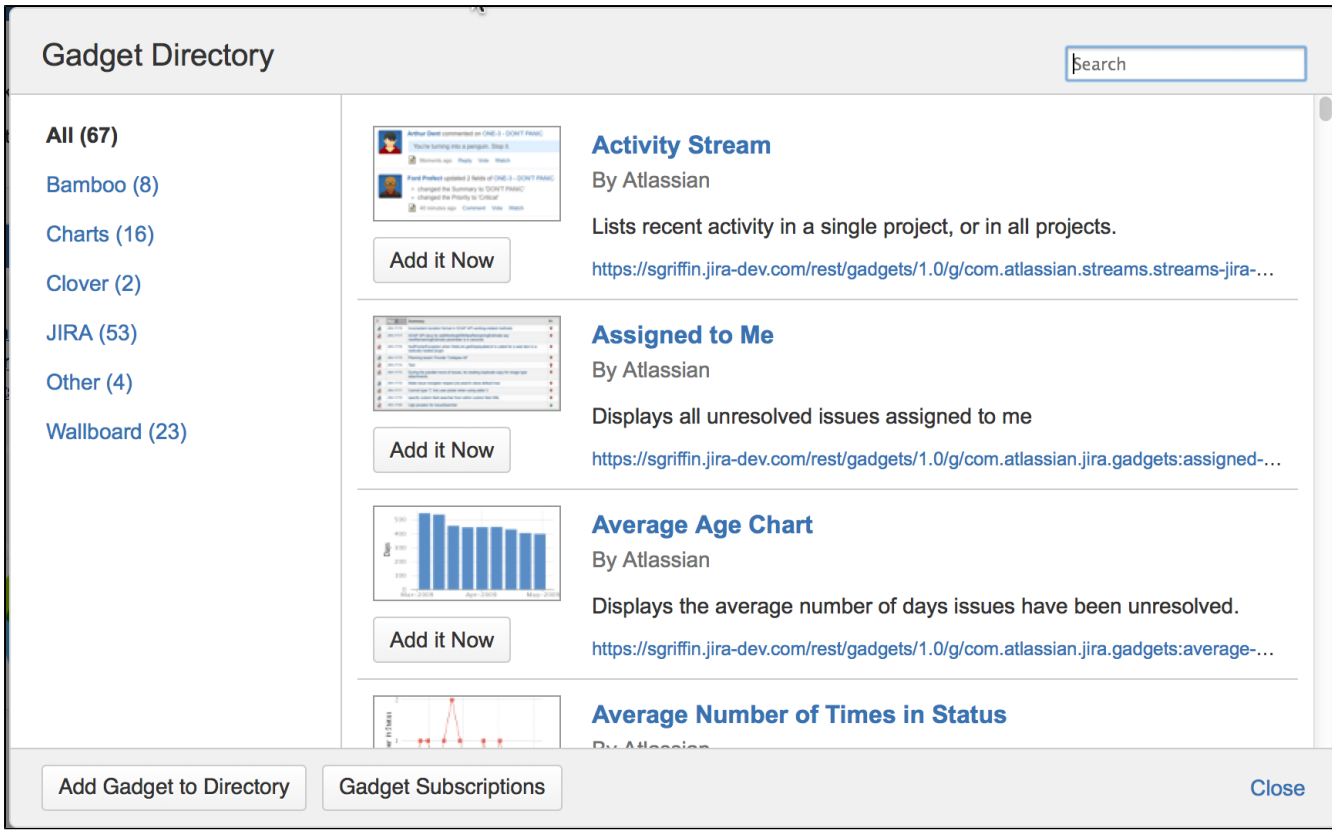
To make use of this feature, you will need two or more applications that support the feature.

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.

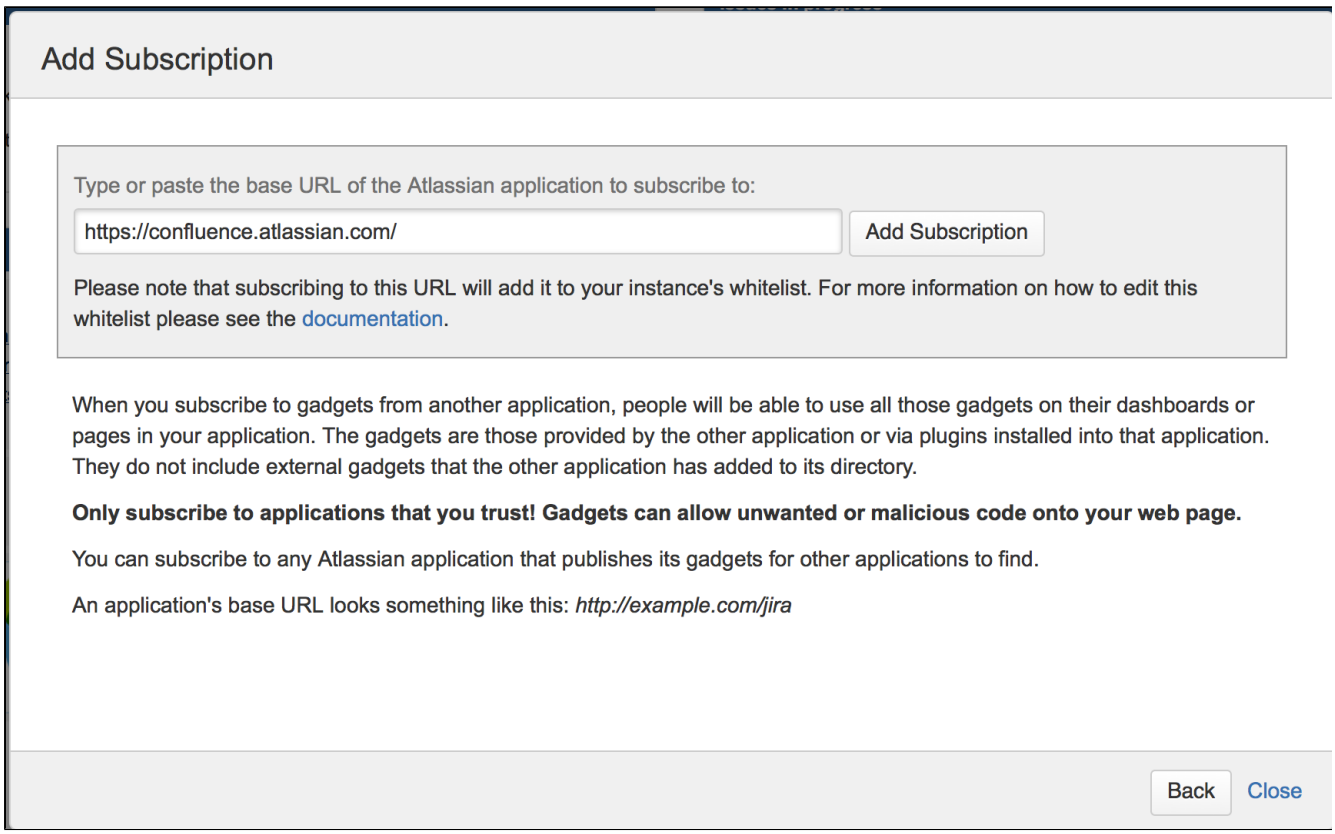
To subscribe to gadgets from another application,

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



Screenshot: Adding a gadget subscription



Related topics

The big list of Atlassian gadgets

Choosing a default language

Overview

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

⚠ **Note:** For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

When JIRA is first installed, the default language may be chosen by selecting it from a list.

On this page:

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.

Related topics

- Translating JIRA

Translating JIRA

This page contains information about translating JIRA into languages other than English.

On this page:

Atlassian Translations – a collaborative environment for creating translations of JIRA

The Atlassian Translations site provides a collaborative environment for customers to translate JIRA. (Refer to the instructions for more information). At present there are thousands of accepted translations across a number of languages. We need your help to make this even better! If you are looking at updating or creating a language pack please use Atlassian Translations and tell us about your experience. You can log in with your My Atlassian account. To provide feedback or submit an existing language pack for import please contact The Internationalization Team.

There is also a plugin currently in Beta release that allows you to translate most JIRA items on the fly: InProduct Translations.

What translations of JIRA are currently available?

Currently, JIRA ships with a number of translations in the most commonly-requested languages. You can easily update these via the Universal Plugin Manager — please 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 for the latest list.

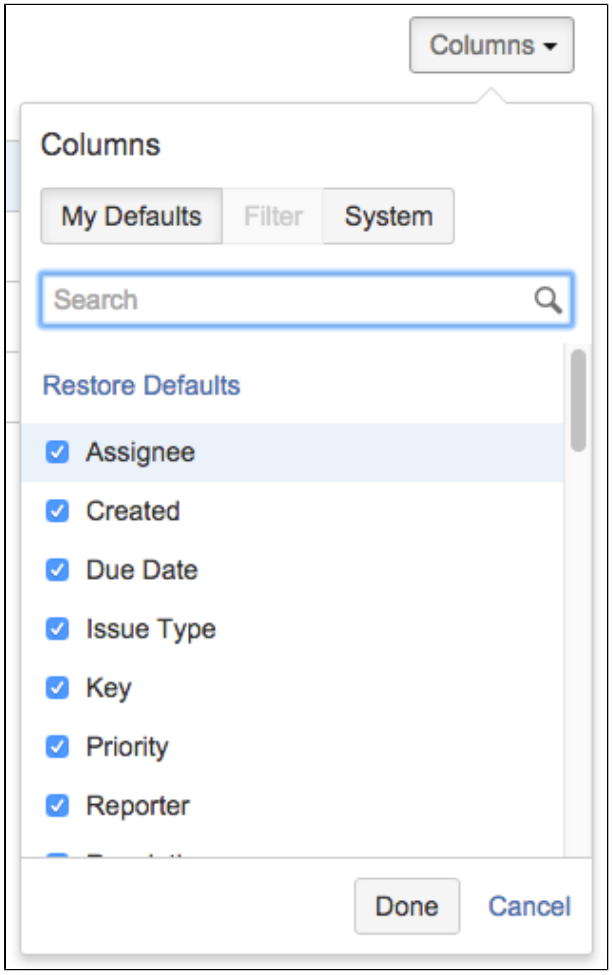
Individual users can also choose their preferred language from the same list in their user profile.

What about translations of the documentation?

We do not currently offer translations of the JIRA documentation into other languages.

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:

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 custom fields, even if selected, do not appear in the Issue Navigator for all issues. For example, project-specific custom fields will be shown only if the filter has been restricted to that project only. Issue type custom fields will only appear if the filter has been restricted to that issue type.

Creating links in the application navigator

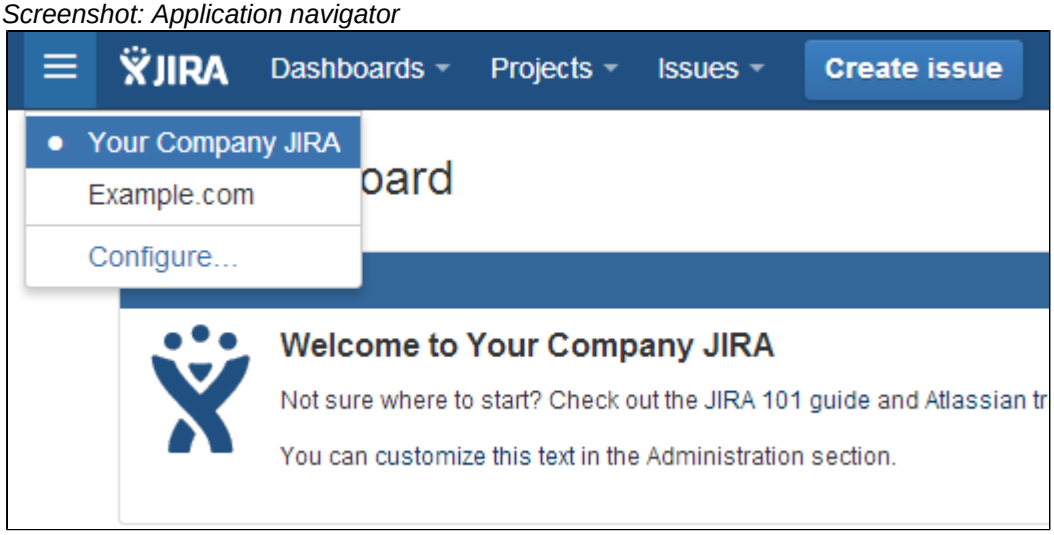
You can add custom links in the application navigator, to make it easier for users to navigate to frequently used information.

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).

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 it's 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 an 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.

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 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 componentlead, 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.

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.

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 eternal 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.

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	Contains people who are JIRA system administrators. By default, this group: • is a member of the **Administrators project** role. • has the **JIRA Administrators** and the **JIRA System Administrators** global permissions.	• 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.	• 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-agents	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.

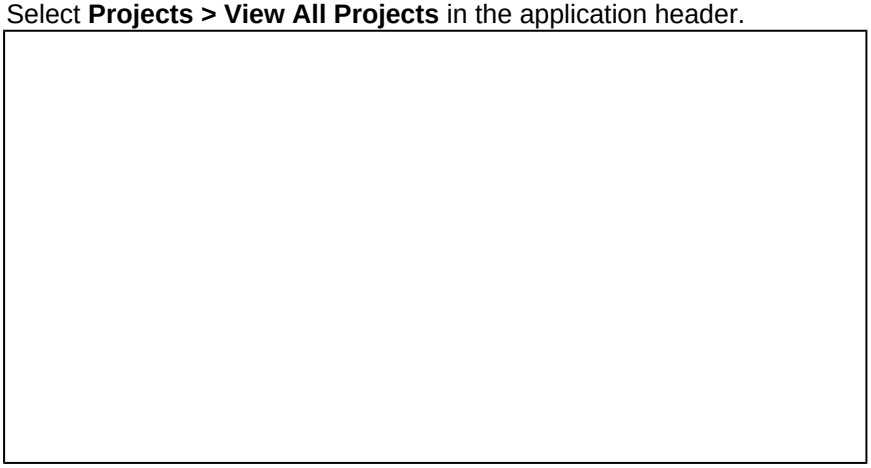
Assign group access to a project role

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.

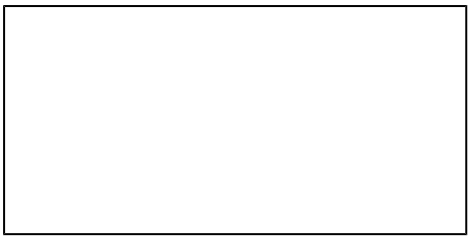
The best way to give users access to a project role is to grant access to a group. This way you can assign a group access and then simply add a user to the group and save yourself some time managing individual user permissions.

To assign access to a project role on the group level:

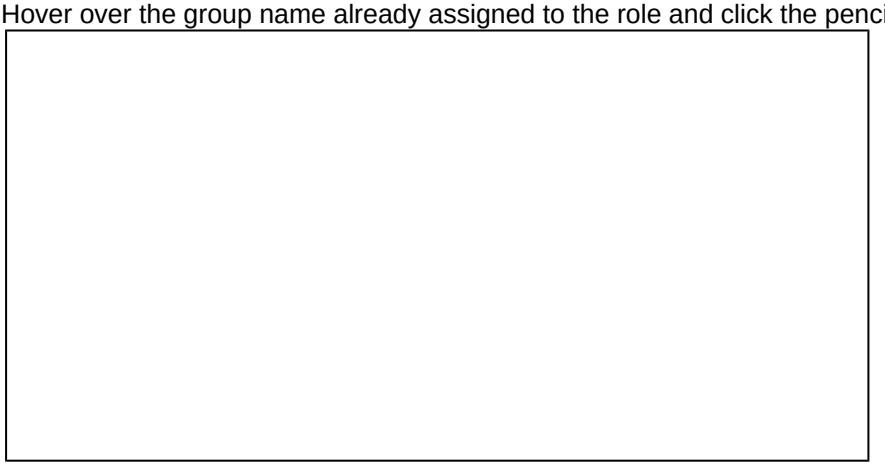
1. Select **Projects > View All Projects** in the application header.



2. Click the title of the project that you want to assign permissions to.
3. On the bottom of the left panel, click
   to open the **Project administration** window.



4. Click **Roles** in the left panel.
5. Hover over the group name already assigned to the role and click the pencil to add more groups.



6. Type the group names you want to assign.
7. Click **Update** to finish.

You can follow this same process to assign an individual user to a project role. Instead of hovering over the group name, hover over the 'Users' column to add a username instead. 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 you 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 an 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

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.

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 centralised 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

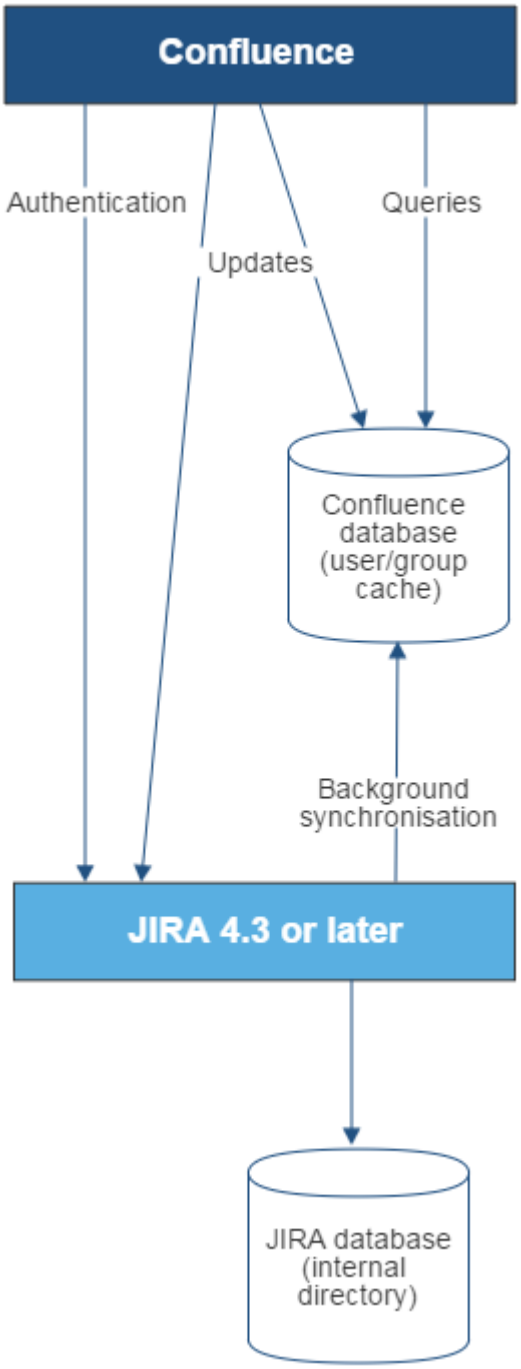


Diagram above: Confluence connecting to JIRA for user management.

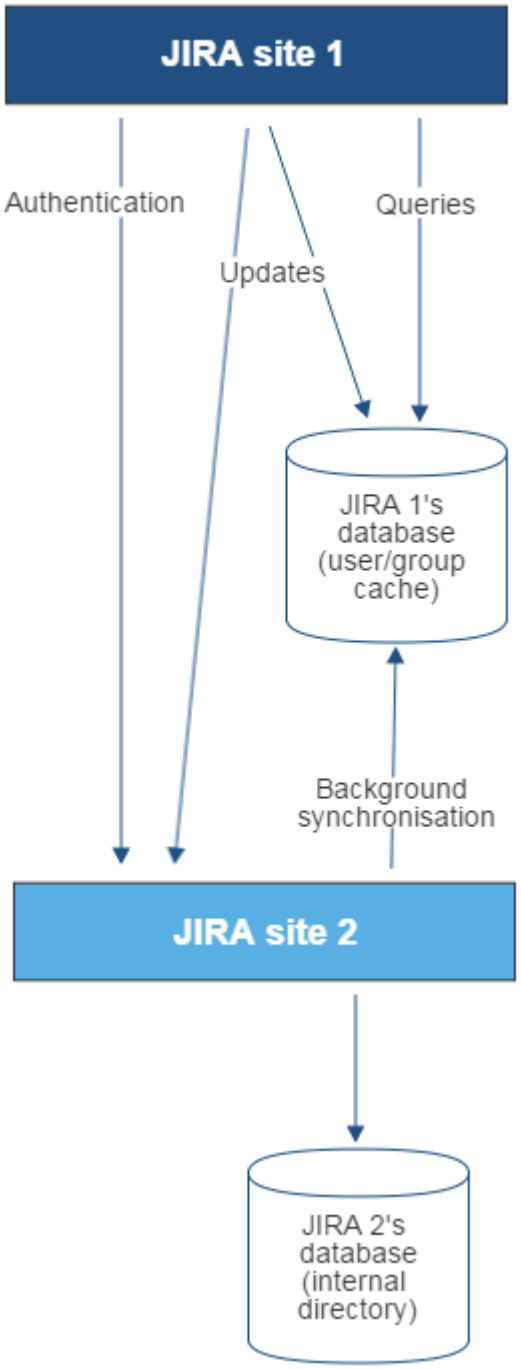


Diagram above: One JIRA site connecting to another for user management. JIRA site 2 does the user management, storing the user data in its internal directory.

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

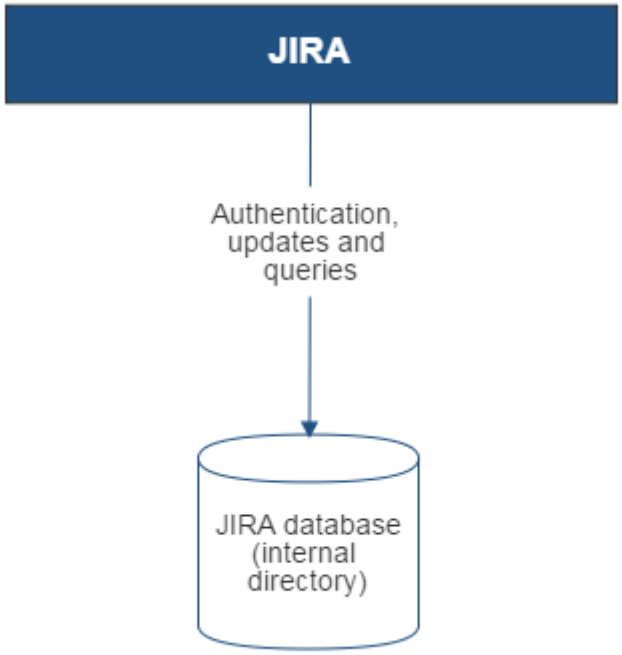


Diagram above: JIRA using its internal directory for user management.

JIRA with read/write connection to LDAP

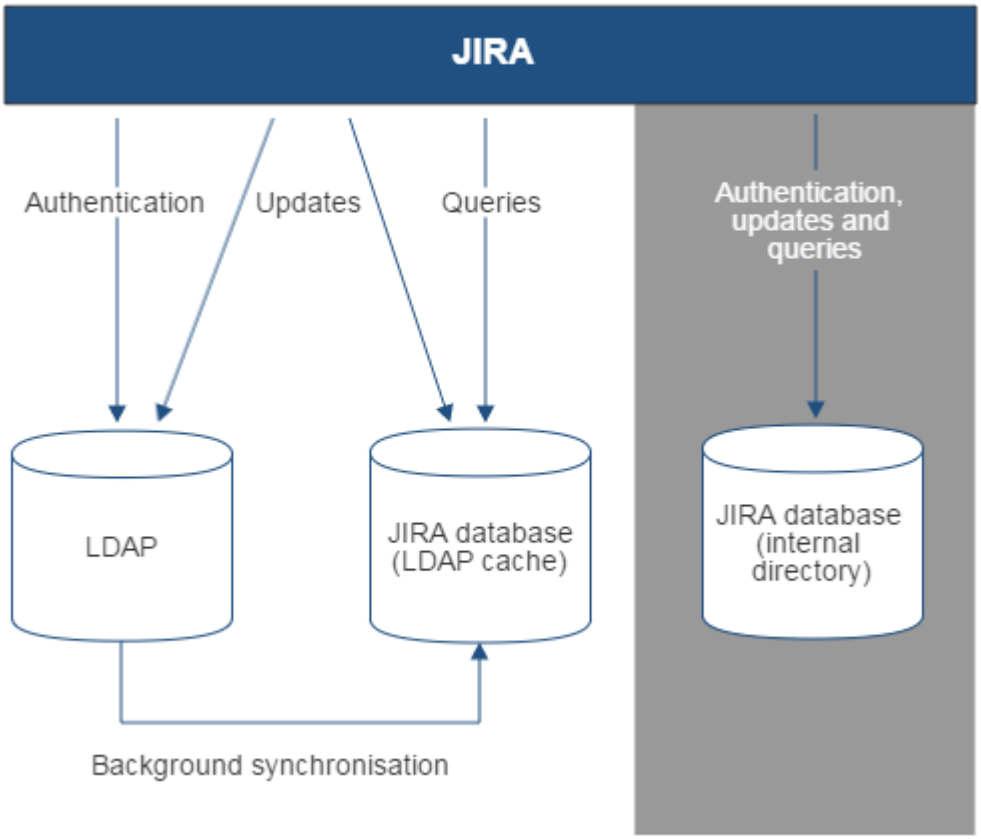


Diagram above: JIRA connecting to an LDAP directory.

JIRA with read-only connection to LDAP, with local groups

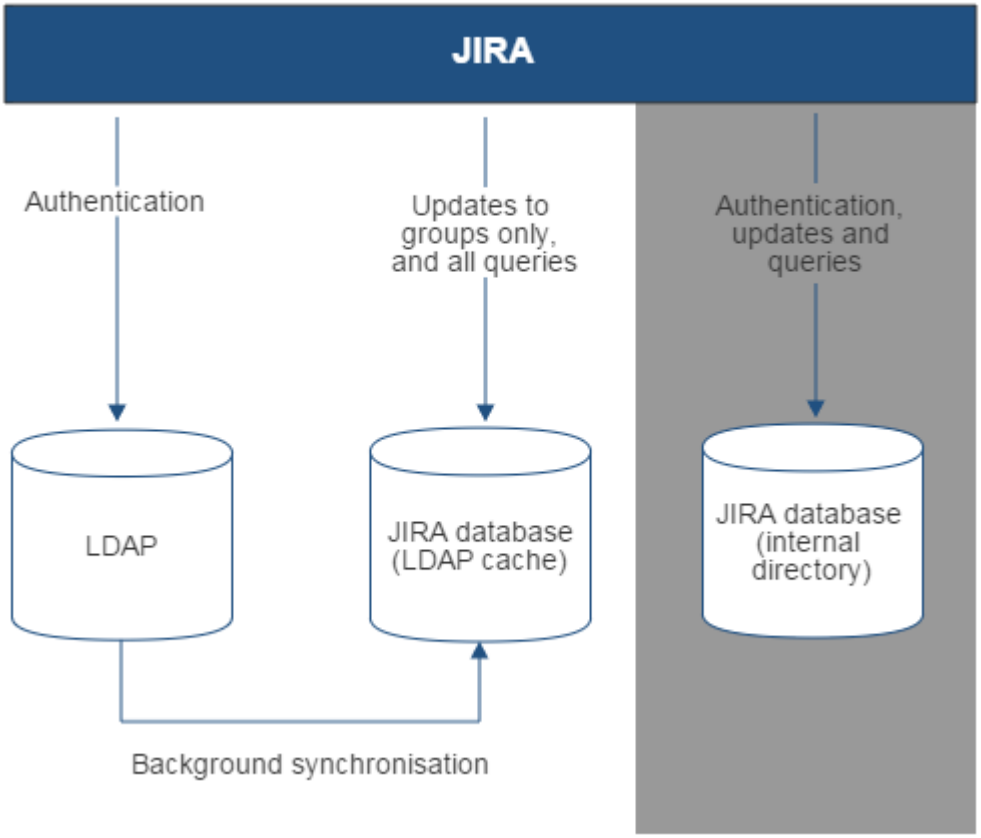


Diagram above: JIRA connecting to an LDAP directory with permissions set to read only and local groups.

JIRA internal directory with LDAP authentication

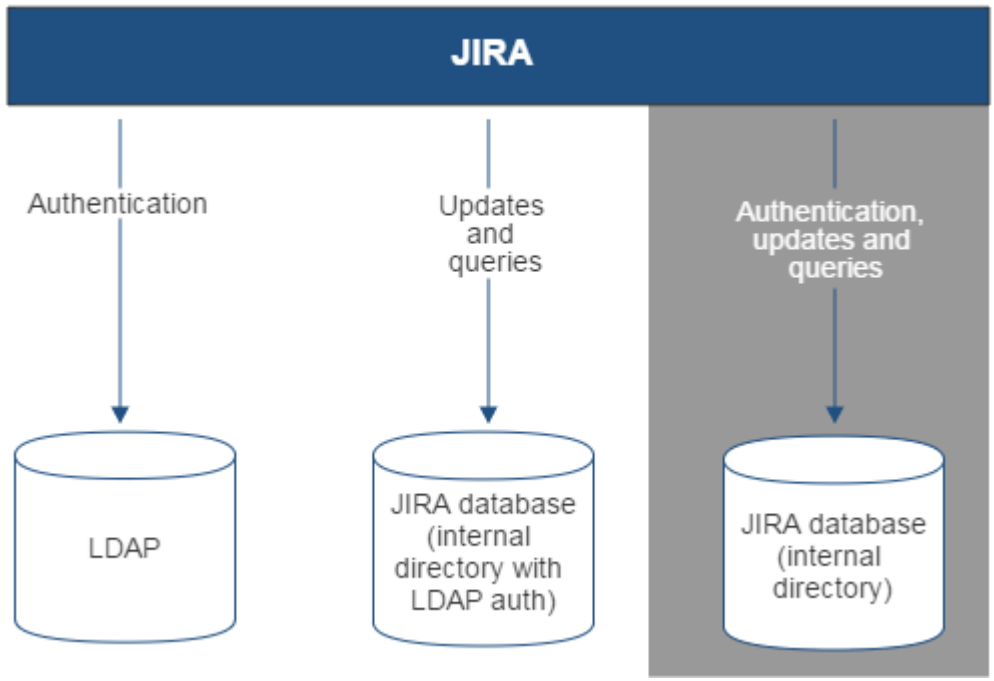


Diagram above: JIRA connecting to an LDAP directory for authentication only.

JIRA with LDAP authentication, copy users on first login

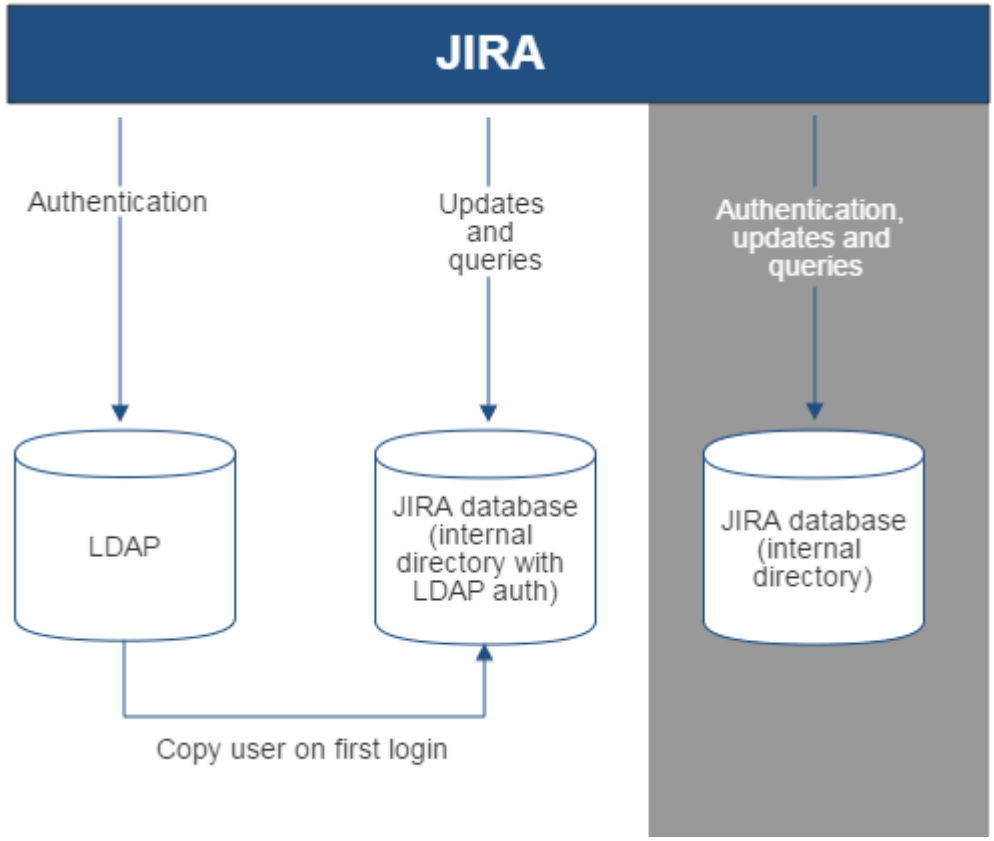


Diagram above: JIRA connecting to an LDAP directory for authentication only, with each user copied to the internal directory when they first log in to JIRA.

One JIRA instance connecting to another

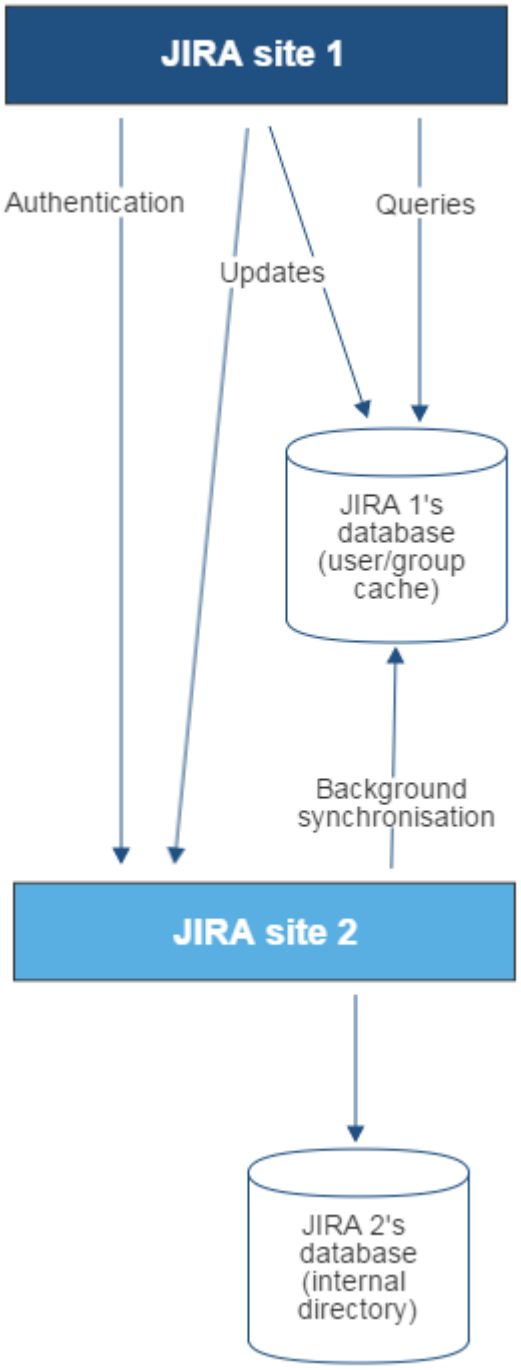


Diagram above: One JIRA site connecting to another for user management. JIRA site 2 does the user management, storing the user data in its internal directory.

Confluence and JIRA connecting to Crowd

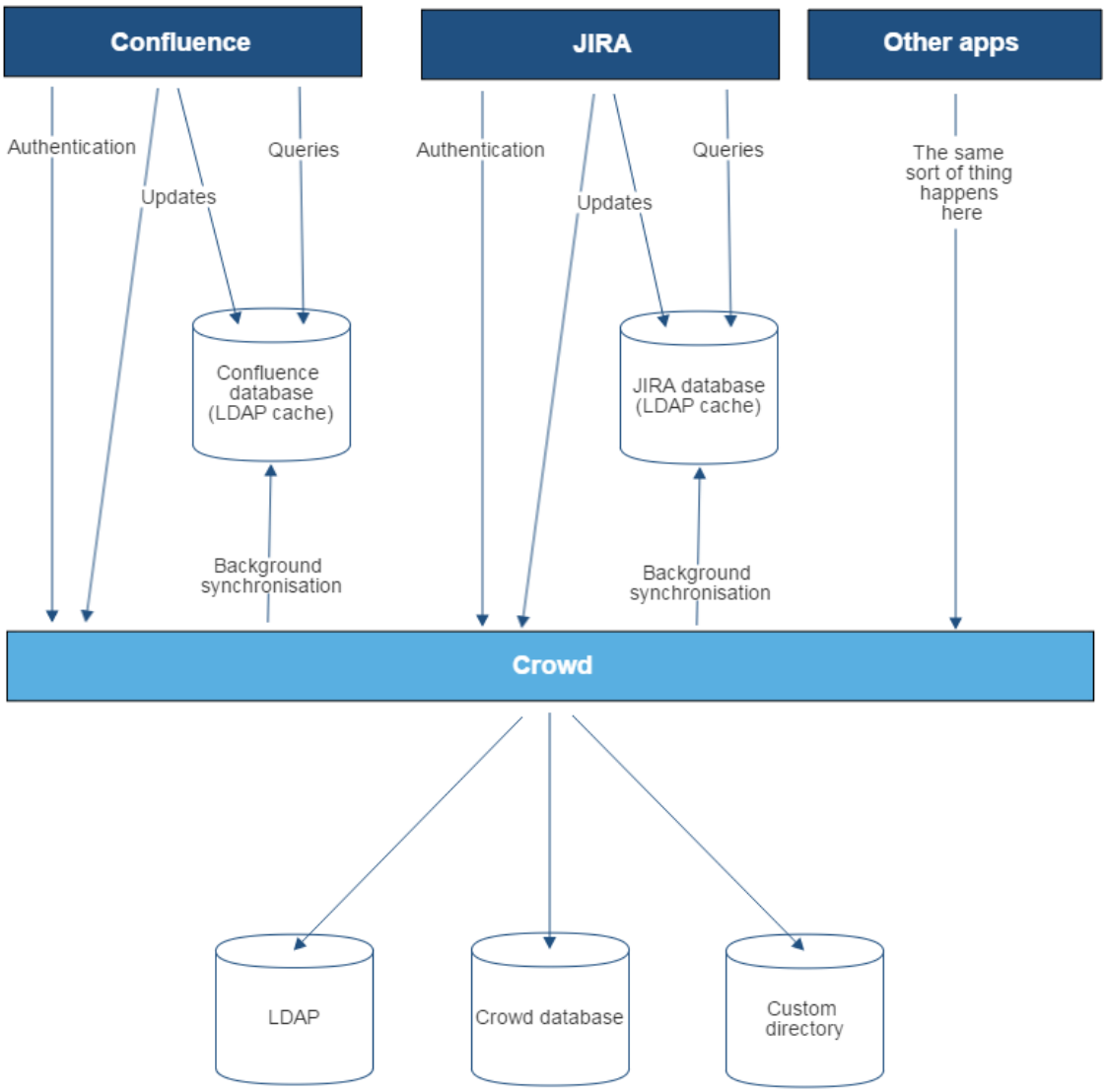


Diagram above: Confluence, JIRA and other applications connecting to Crowd for user management.

A number of applications connecting to JIRA

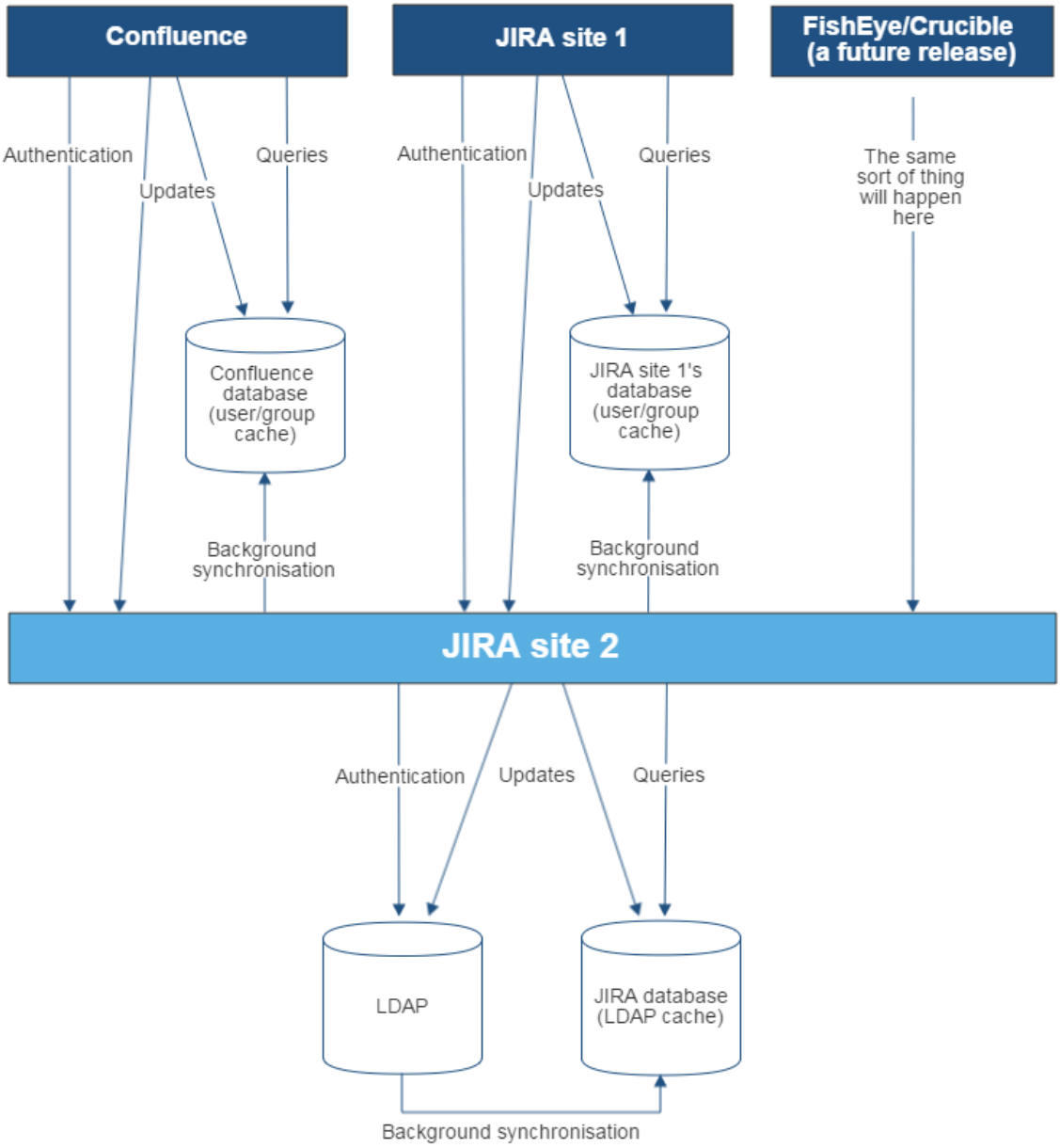


Diagram above: A number of applications connecting to JIRA (site 2) for user management, with JIRA in turn connecting to an LDAP server.

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

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:

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'** li nk 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.

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:

a. Go to **Project administration** > **Request security**.
b. 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 do not 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

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:

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'. 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.

This page describes how JIRA handles nested groups that exist in one or more of your directory servers.

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:

- Before enabling nested groups for a specific directory type in JIRA, please make sure that your directory server supports nested groups.
- 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.

On this page:

Effect of Nested Groups

This section summarises the effect nested groups will have on login and permissions, and on the viewing and updating of users and groups.

Login

When a user logs in, they will be allowed access to the application if they belong to an authorised group or any of its sub-groups.

Permissions

The user will be allowed access to 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 cannot view or edit the nested groups themselves. You will not be able to see that one group is a member of another group.

Adding and Updating Group Memberships

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 (tree) of groups contained in the flattened list, the user will be removed from the group.
- Otherwise, you will 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

Let's assume that 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 groups '**engineering-group**' and '**techwriters-group**'. We want to grant both groups developer-level access to our JIRA site. We will assume you have a group in your JIRA site called 'jira-developers' that have 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**'.

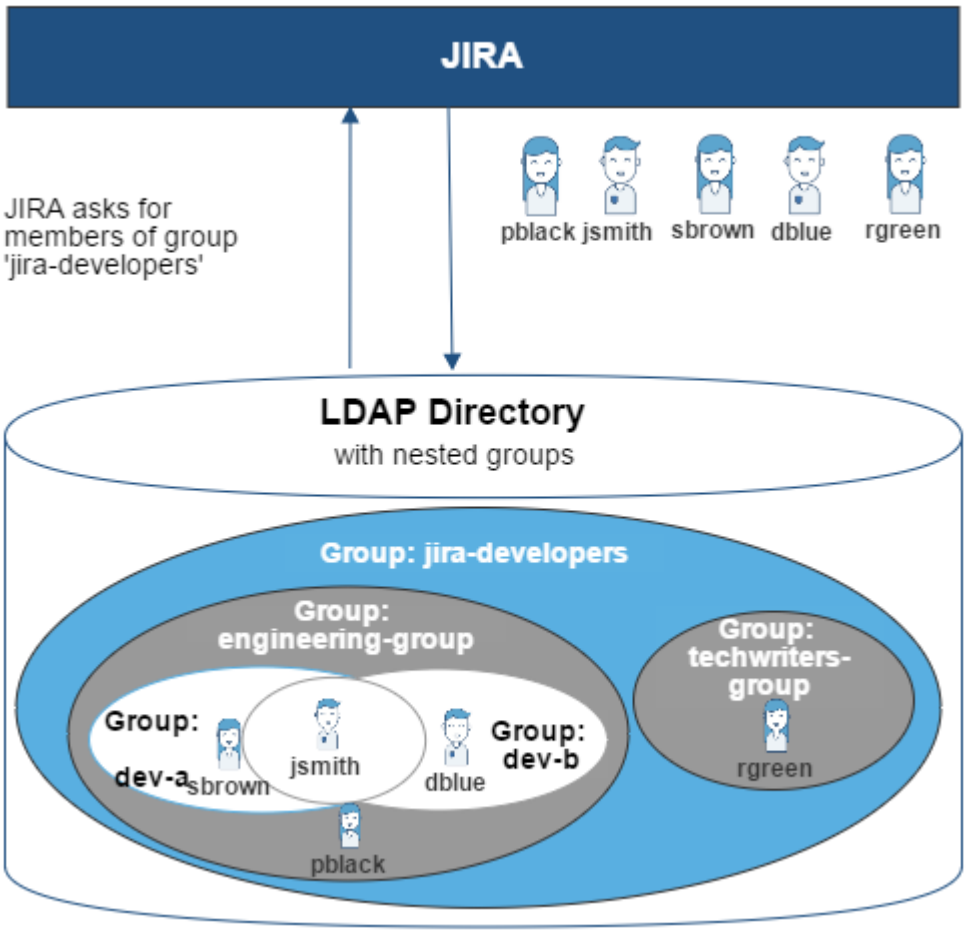
Group memberships are now:

- **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 will receive the following list:

- **pblack**
- **jsmith**
- **sbrown**
- **dblue**
- **rgreen**

Diagram: Sub-groups as members of the 'jira-developers' group



Example 3: Sub-Groups as Members of the 'confluence-users' group

In an LDAP directory server, we have groups '**engineering-group**' and '**payroll-group**'. We want to grant both groups access to our Confluence site.

- Add a group called '**confluence-users**'.
- Add the '**engineering-group**' as a sub-group of '**confluence-users**'.
- Add the '**payroll-group**' as a sub-group of '**confluence-users**'.

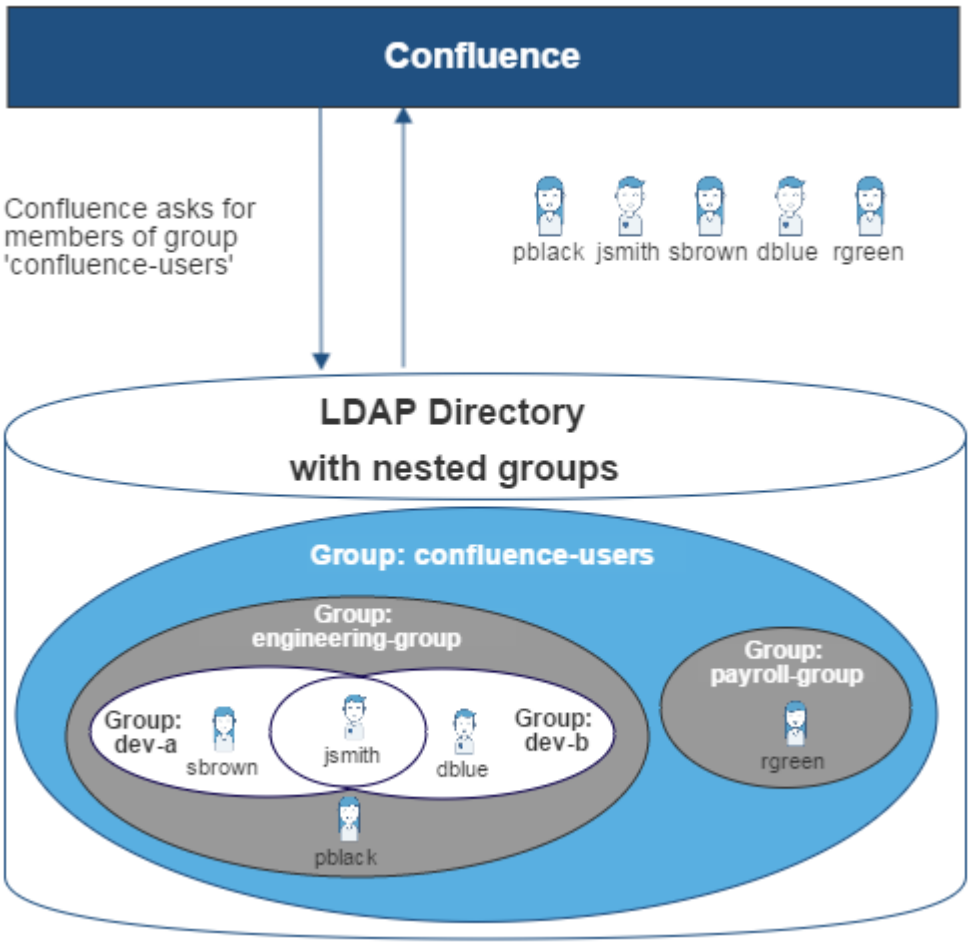
Group memberships are now:

- **confluence-users** — sub-groups: **engineering-group**, **payroll-group**
- **engineering-group** — sub-groups: **dev-a**, **dev-b**; users: **pblack**
- **dev-a** — users: **jsmith**, **sbrown**
- **dev-b** — users: **jsmith**, **dblue**
- **payroll-group** — users: **rgreen**

When Confluence requests a list of users in the '**confluence-users**' group, it will receive the following list:

- **pblack**
- **jsmith**
- **sbrown**
- **dblue**
- **rgreen**

Diagram: Sub-groups as members of the 'confluence-users' 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 defined as 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.

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 synchronisation 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 synchronisation 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 synchronisation task.

Our Test Results

We performed internal testing of synchronisation with an AD server on our local network consisting of 10 000 users, 1000 groups and 200 000 memberships.

We found that the initial synchronisation took about 5 minutes. Subsequent synchronisations 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 synchronisation 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 synchronisations 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 synchronisation process caches data in the database, the performance of your database will affect the performance of the synchronisation.
- **JVM heap size.** If your heap size is too small for your userbase, you may experience heavy garbage collection during the synchronisation process which could in turn slow down the synchronisation.

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 synchronises with Active Directory (AD), the synchronisation task requests only the changes from the LDAP server rather than the entire user base. This optimises the synchronisation process and gives much faster performance on the second and subsequent requests.

On the other hand, this synchronisation 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 synchronise the directory cache after you have made the changes to ensure cache consistency.
2. **Synchronising 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 synchronisation. (You can of course define multiple different directories, each pointing to its own respective AD server.)
3. **Synchronising with AD servers behind a load balancer is not supported.** As with synchronising 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 synchronise 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 synchronisation 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 synchronisation 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
If **all** the following are true: • 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.	Your environment meets the optimal requirements for using a JIRA application for user management.

If **one or more** of the following are true: • 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.	We recommend that you install Atlassian Crowd for user management and SSO.
If you are considering creating a custom directory connector to define your own storage for users and groups...	Please see if one of the following solutions will work for you: • 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

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 behaviour 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.

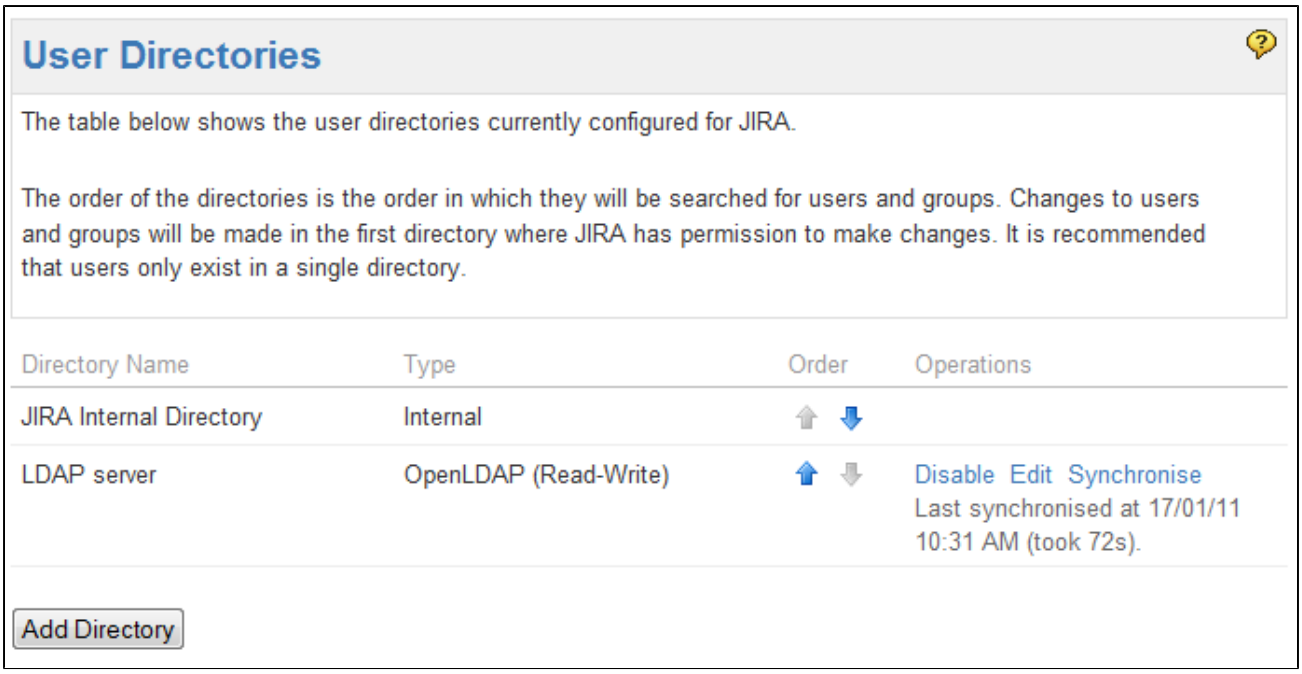
⚠️ We recommend that you keep either an administrator or system administrator user active in your internal directory for troubleshooting problems with your user directories.

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 recognise 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



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

On this page:

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.

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

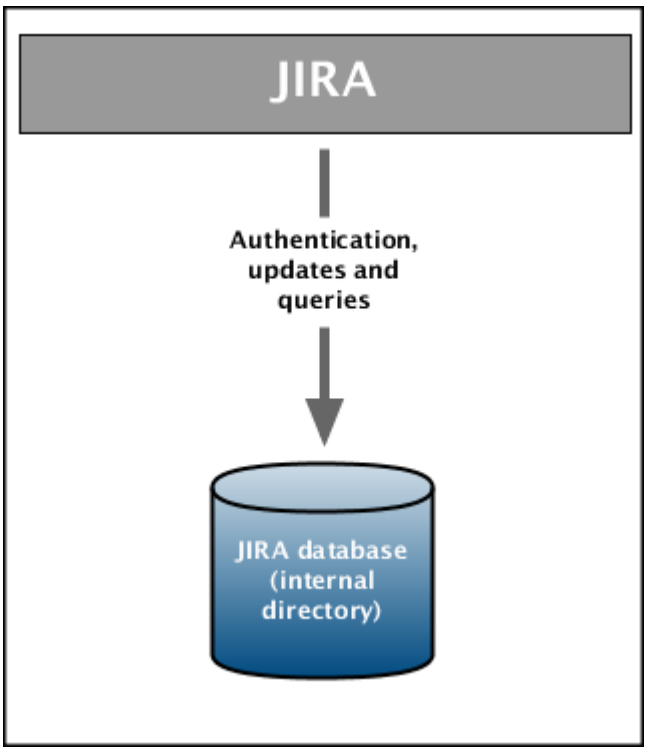


Diagram above: JIRA using its internal directory for user management.

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 synchronising data from external directories.

On this page:

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 synchronising data from external directories.

Server settings

Setting	Description
Name	Enter a meaningful name to help you identify the LDAP directory server. Examples: • `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: • `Microsoft Active Directory` • `OpenDS` • And more.
Hostname	The host name of your directory server. Examples: • `ad.example.com` • `ldap.example.com` • `opends.example.com`
Port	The port on which your directory server is listening. Examples: • `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: • `cn=administrator,cn=users,dc=ad,dc=example,dc=com` • `cn=user,dc=domain,dc=name` • `user@domain.name` Ensure that this is an administrator user for the LDAP engine. For example, in Active Directory the user will need to be a member of the built-in Administrators group. The specific privileges for the LDAP user that is used to connect to LDAP are bind and read (user info, group info, group membership, update sequence number, deleted objects). The need for admin privileges is because a normal user can't access uSNChanged attribute and deleted objects container, causing incremental sync to fail silently, this was reported in a bug here CWD-3093.
Password	The password of the user specified above. **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.

Schema settings

Setting	Description
Base DN	The root distinguished name (DN) to use when running queries against the directory server. Examples: • `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 `ldp.exe` which is useful for finding out and configuring the the LDAP structure of your server.
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: • `ou=Users`
Additional Group DN	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: • `ou=Groups`

Permission settings

Note: You can only assign LDAP users to local groups when 'External Management 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	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. 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 **CONF-28621** - User Loses all Local Group Memberships If LDAP Sync is Unable to find the User, but the User appears again in subsequent syncs OPEN
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	*Option available in Confluence 3.5 and later, and JIRA 4.3.3 and later.* 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. *In Confluence 3.5 to Confluence 3.5.1:* 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. *In Confluence 3.5.2 and later, and JIRA 4.3.3 and later:* 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 *not* be added automatically to any groups. This change in behaviour 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. Please note that there is no validation of the group names. If you mis-type the group name, authorisation failures will result – users will not be able to access the applications or functionality based on the intended group name. Examples: • `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 'nested groups'. 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.
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 synchronisation after the account's expiration date.
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 `java.naming.referral`) 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	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. This setting determines how your application will compare DNs to determine if they are equal. • 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 Synchronisation	Enable incremental synchronisation if you only want changes since the last synchronisation to be queried when synchronising a directory. ⚠ Please be aware that when using this option, the user account configured for synchronisation must have read access to: • The `uSNChanged` attribute of all users and groups in the directory that need to be synchronised. • The objects and attributes in the Active Directory deleted objects container (see Microsoft's Knowledge Base Article No. 892806 for details). 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. This setting is only available if the directory type is set to "Microsoft Active Directory".
Synchronisation Interval (minutes)	Synchronisation 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)	This setting affects two actions. The default value is 0. • 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	This is the name of the class used for the LDAP user object. Example: • `user`
User Object Filter	The filter to use when searching user objects. Example: • `(&(objectCategory=Person)(sAMAccountName=*))` More examples can be found here and here.

User Name Attribute	The attribute field to use when loading the username. Examples: • `cn` • `sAMAccountName` NB: In Active Directory, the 'sAMAccountName' is the 'User Logon Name (pre-Windows 2000)' field. The User Logon Name field is referenced by 'cn'.
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: • `cn`
User First Name Attribute	The attribute field to use when loading the user's first name. Example: • `givenName`
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: • `sn`
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: • `displayName`
User Email Attribute	The attribute field to use when loading the user's email address. Example: • `mail`
User Password Attribute	The attribute field to use when loading a user's password. Example: • `unicodePwd`
User Unique ID Attribute	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. 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.

Group schema settings

Setting	Description
Group Object Class	This is the name of the class used for the LDAP group object. Examples: • `groupOfUniqueNames` • `group`
Group Object Filter	The filter to use when searching group objects. Example: • `(&(objectClass=group)(cn=*))`

Group Name Attribute	The attribute field to use when loading the group's name. Example: • `cn`
Group Description Attribute	The attribute field to use when loading the group's description. Example: • `description`

Membership schema settings

Setting	Description
Group Members Attribute	The attribute field to use when loading the group's members. Example: • `member`
User Membership Attribute	The attribute field to use when loading the user's groups. Example: • `memberOf`
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.) • 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 seleced, 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.) • 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

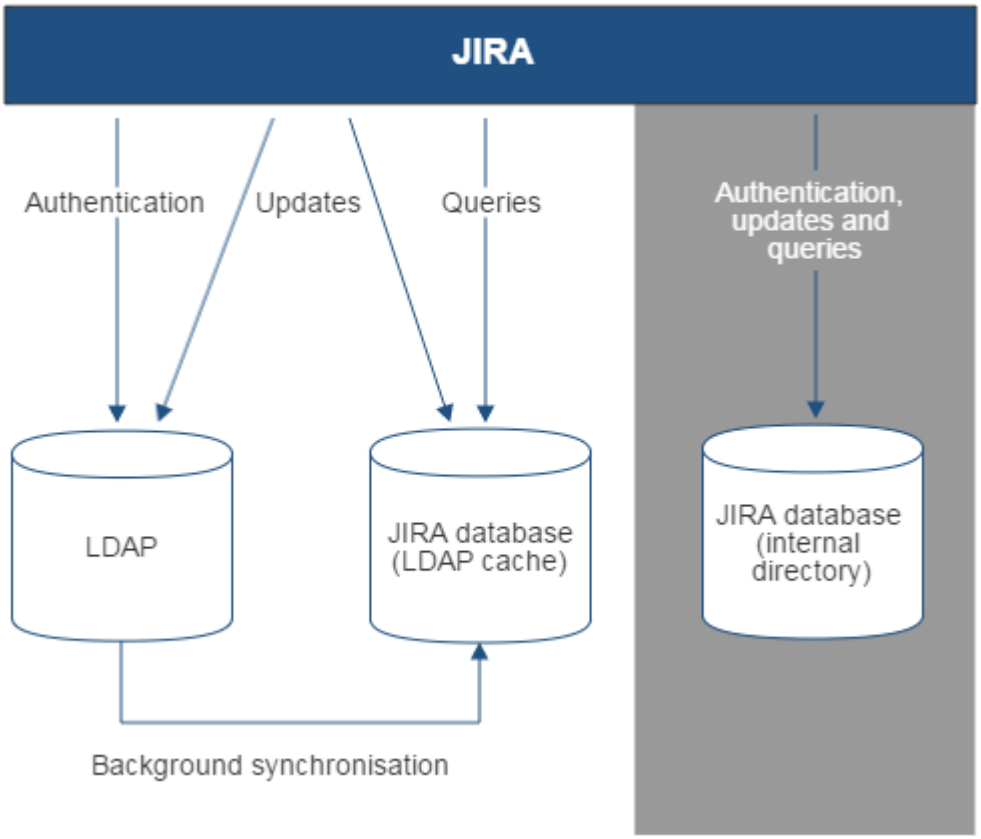


Diagram above: JIRA connecting to an LDAP directory.

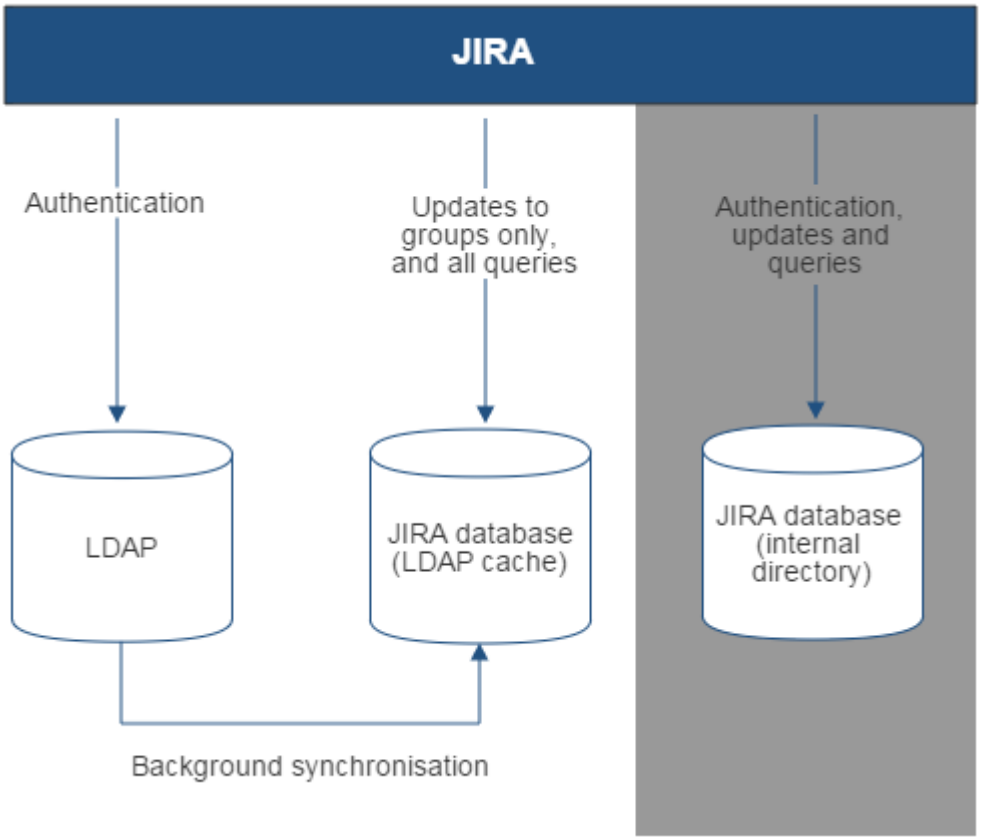


Diagram above: JIRA connecting to an LDAP directory with permissions set to read only and local groups.

Related topics

Configuring user directories

Configuring 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:

> 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.

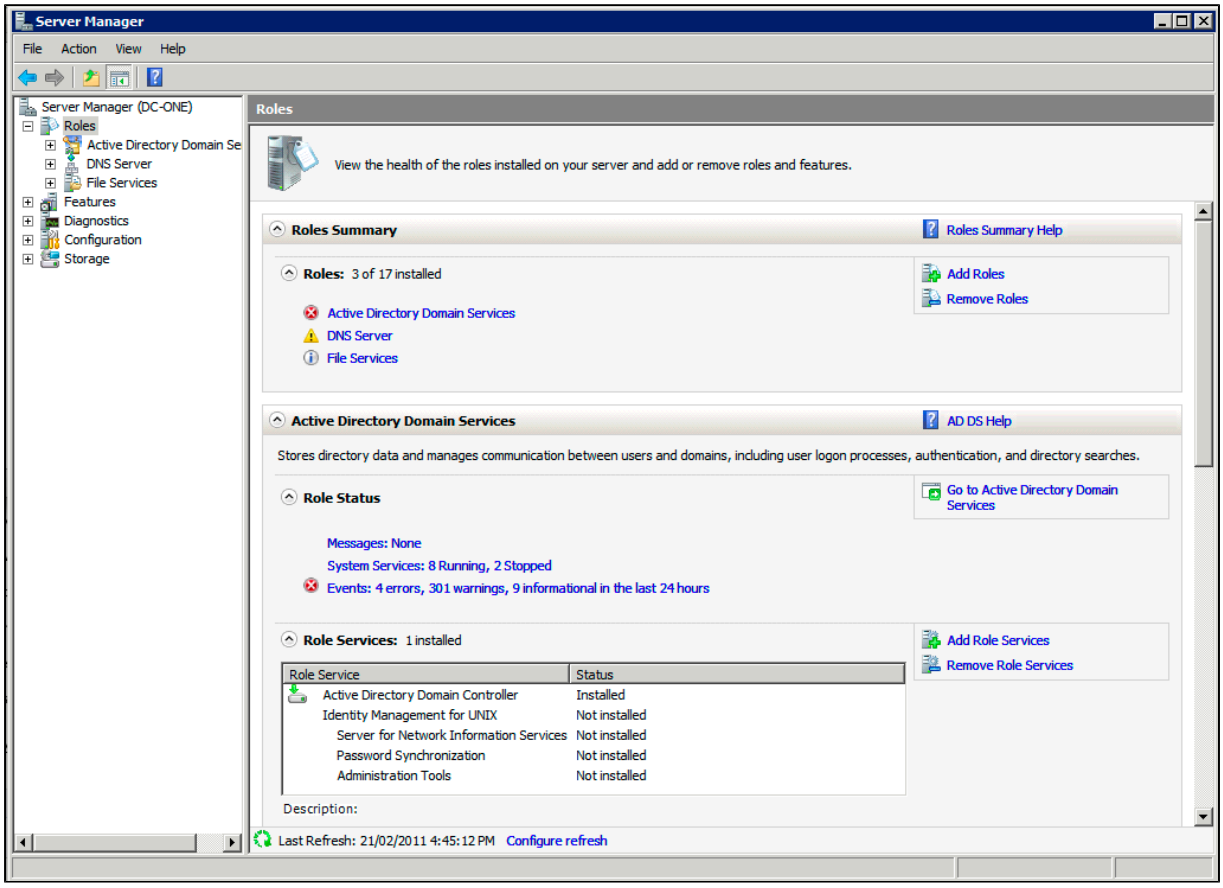
Required Component	Description
Internet Information Services (IIS)	This is required before you can install Windows Certificate Services.
Windows Certificate Services	This installs a certification authority (CA) which is used to issue certificates. Step 1, below, explains this process.
Windows 2000 Service 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).

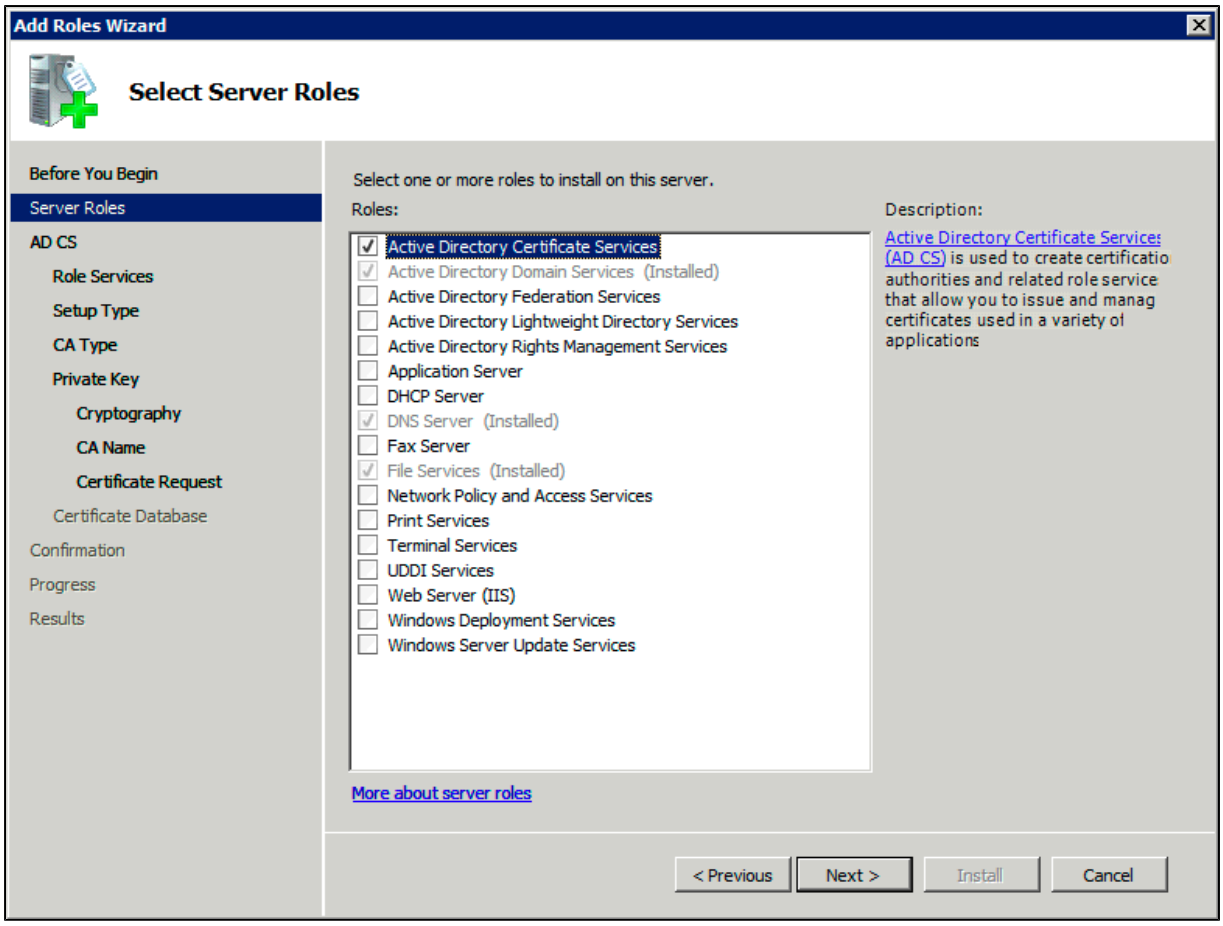
Step 1. Install the Active Directory Certificate Services

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.

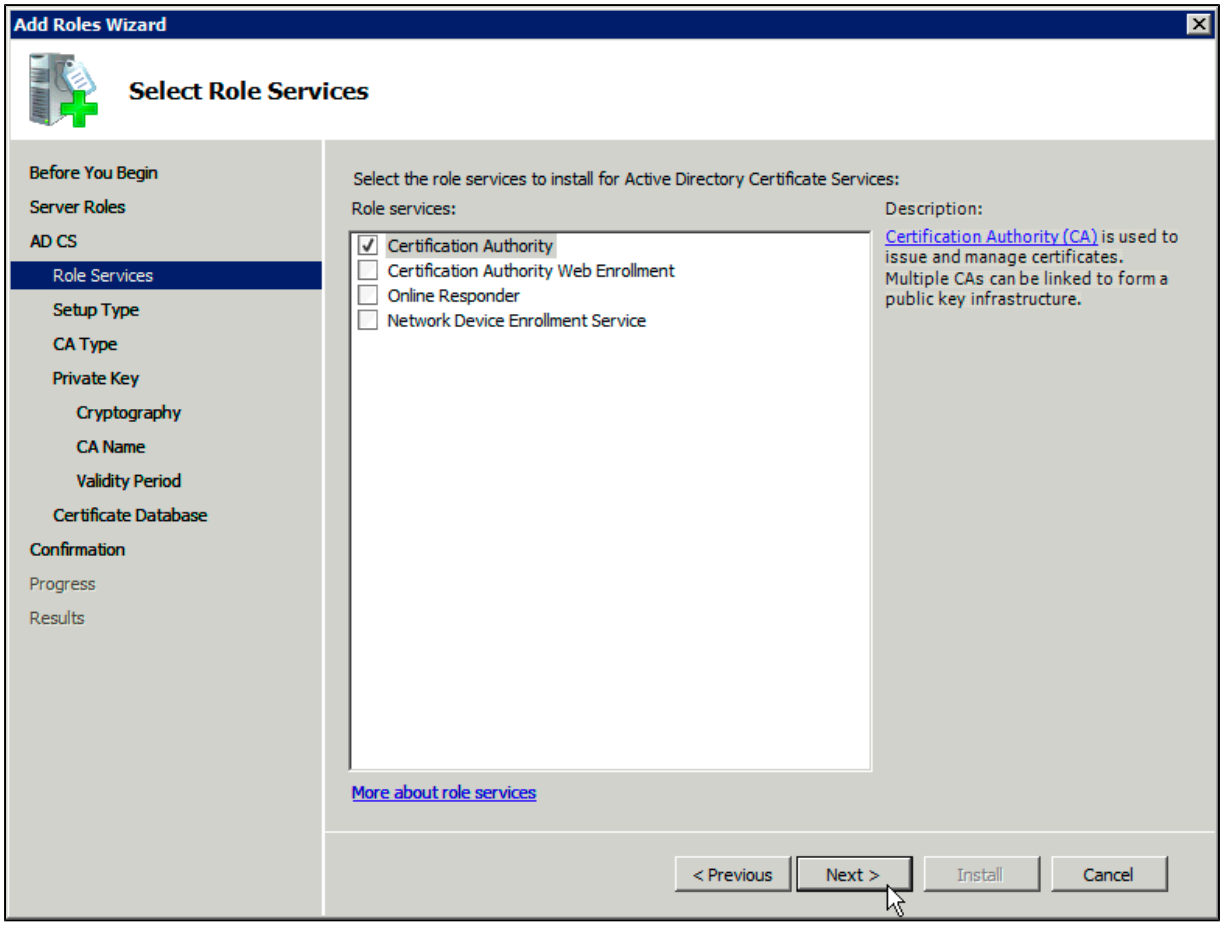
1. Log in to your Active Directory server as an administrator.
2. Click **Start**, point to **Administrative Tools**, and then click **Server Manager**.
3. In the **Roles Summary** section, click **Add Roles**.



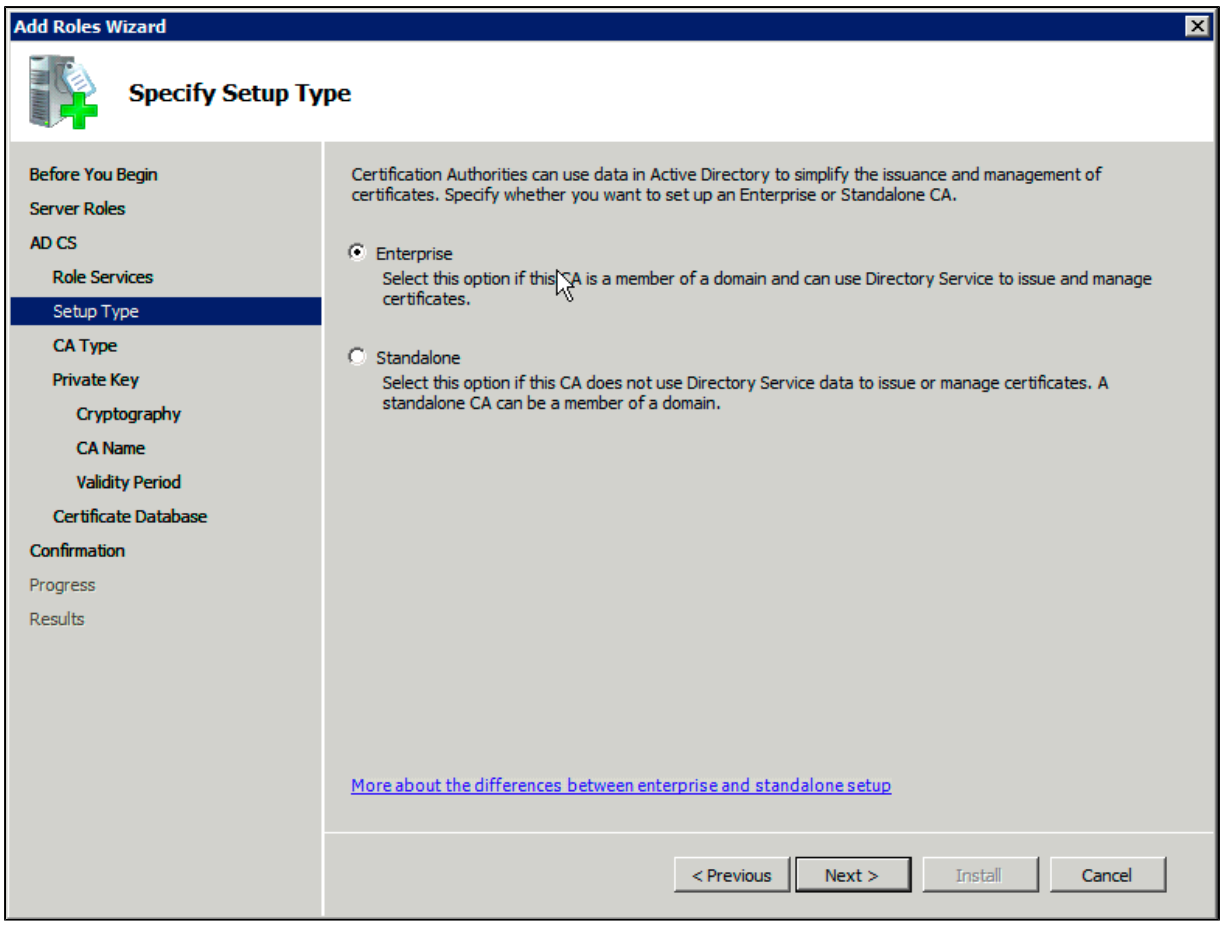
4. On the **Select Server Roles** page, select the **Active Directory Certificate Services** check box. Click **Next** twice.



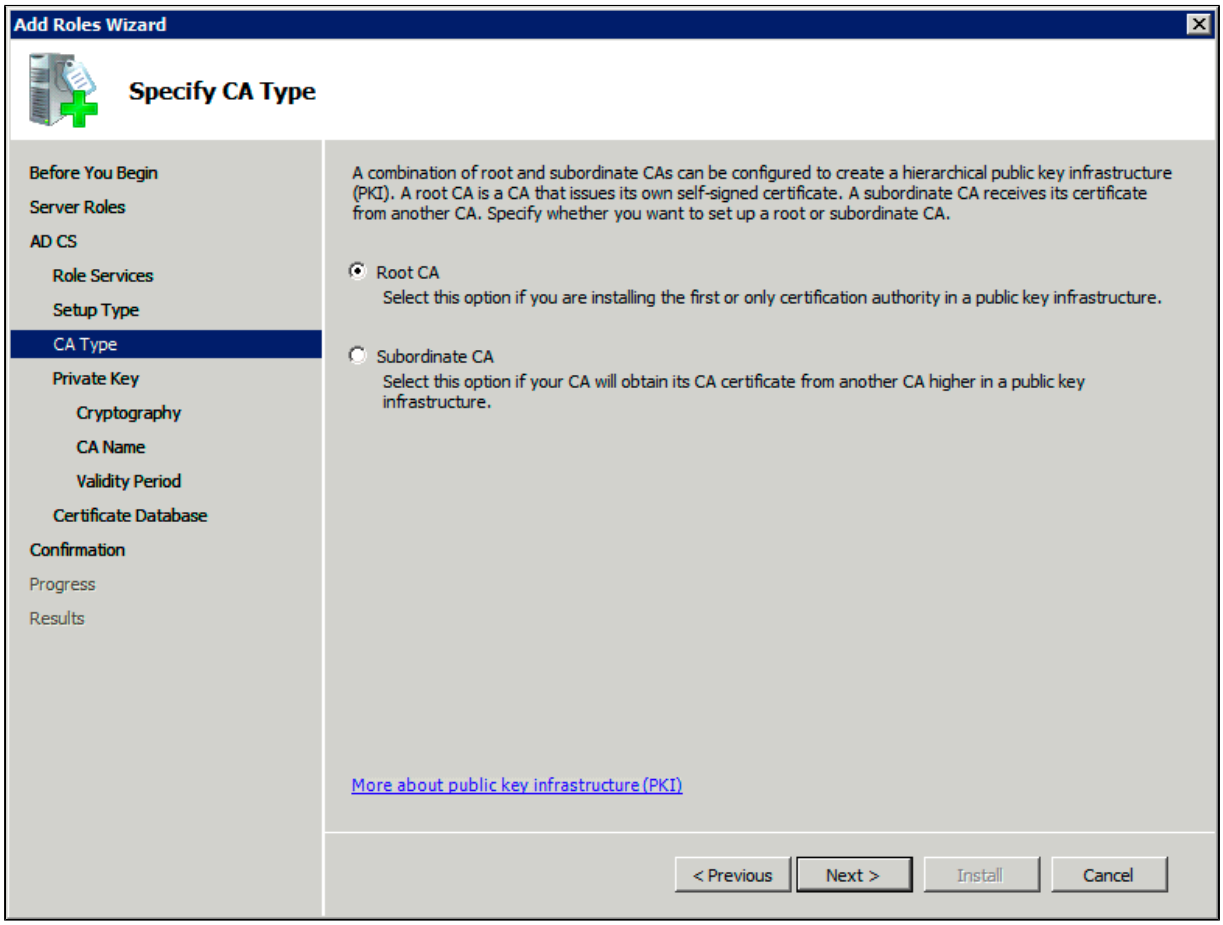
5. On the **Select Role Services** page, select the **Certification Authority** check box, and then click **Next**.



6. On the **Specify Setup Type** page, click **Enterprise**, and then click **Next**.

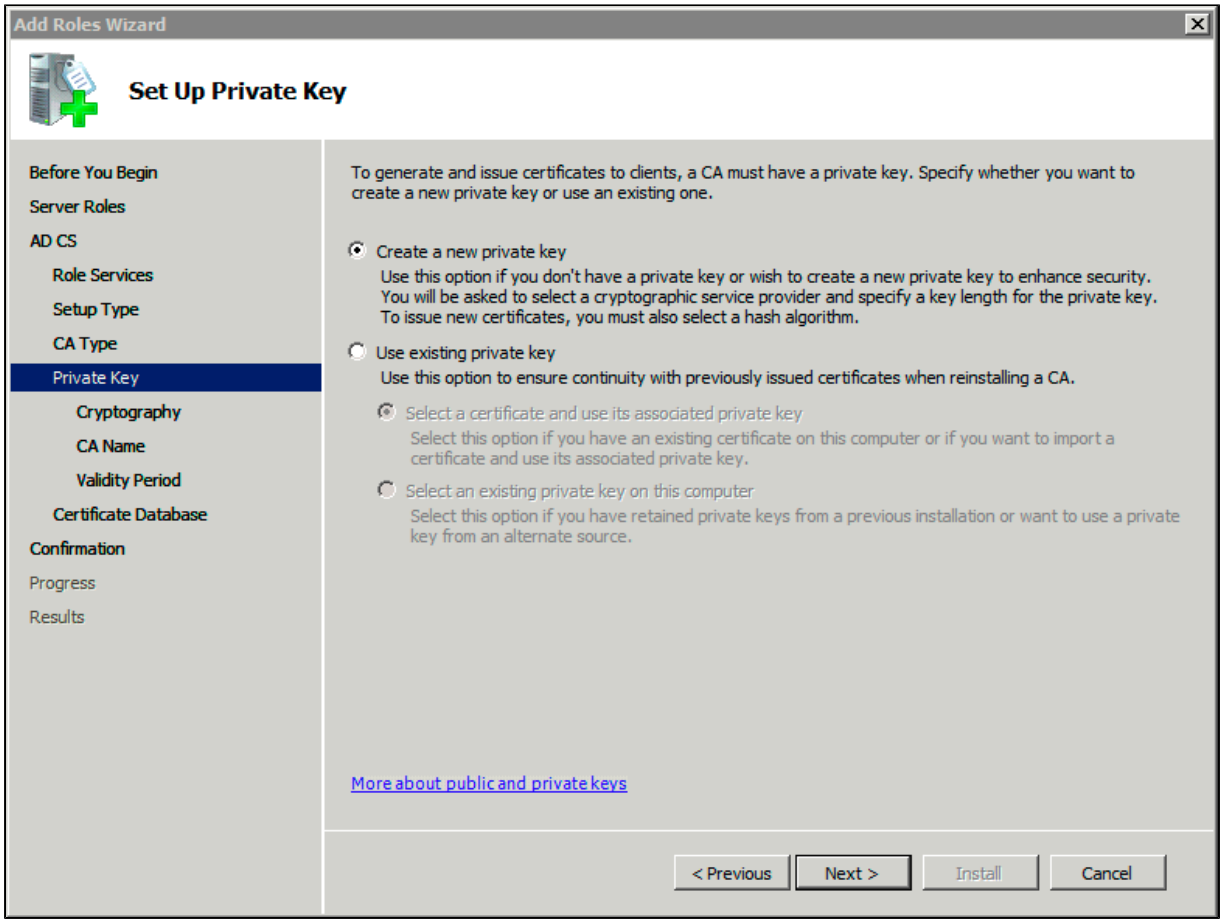


7. On the **Specify CA Type** page, click **Root CA**, and then click **Next**.

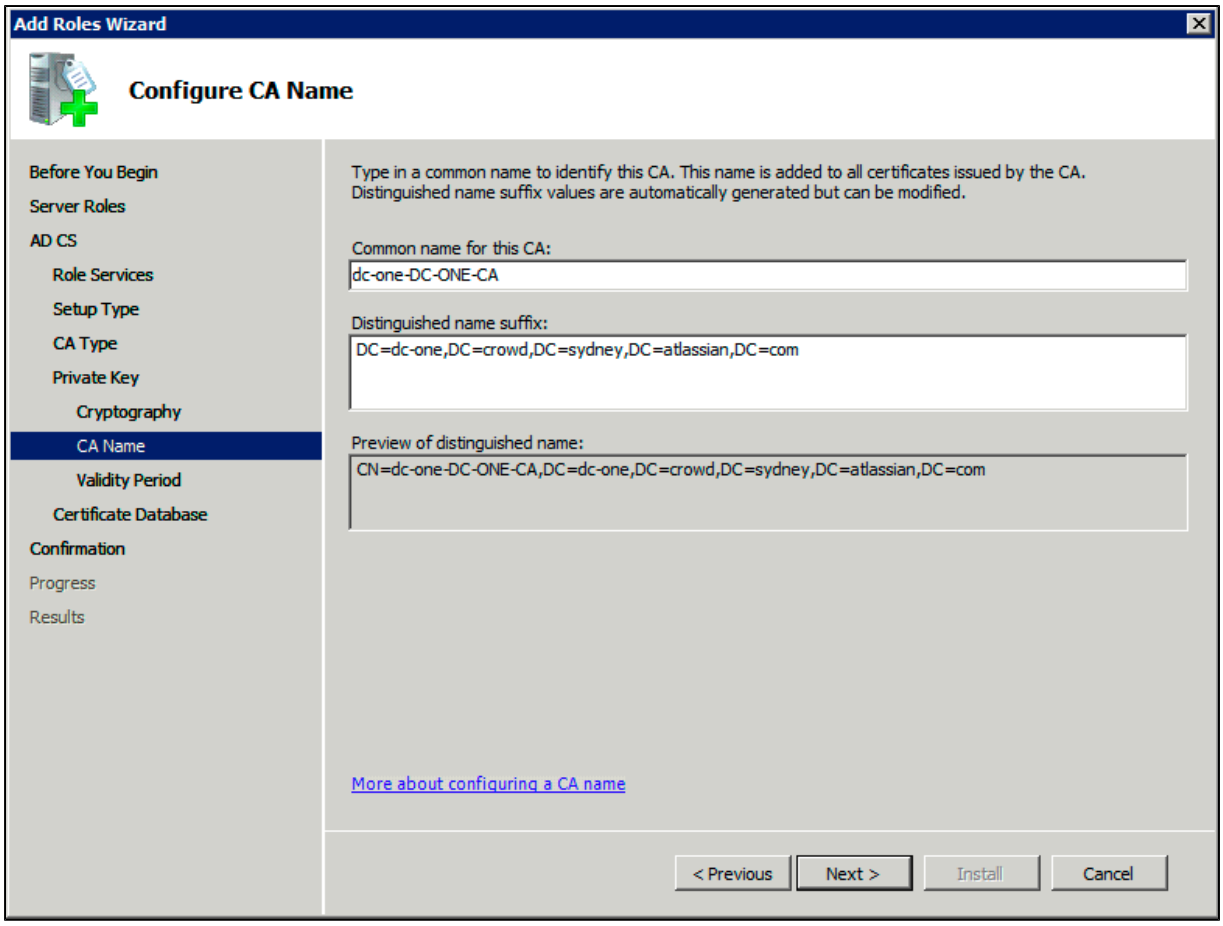


8. On the **Set Up Private Key** and **Configure Cryptography for CA** pages, you can configure optional

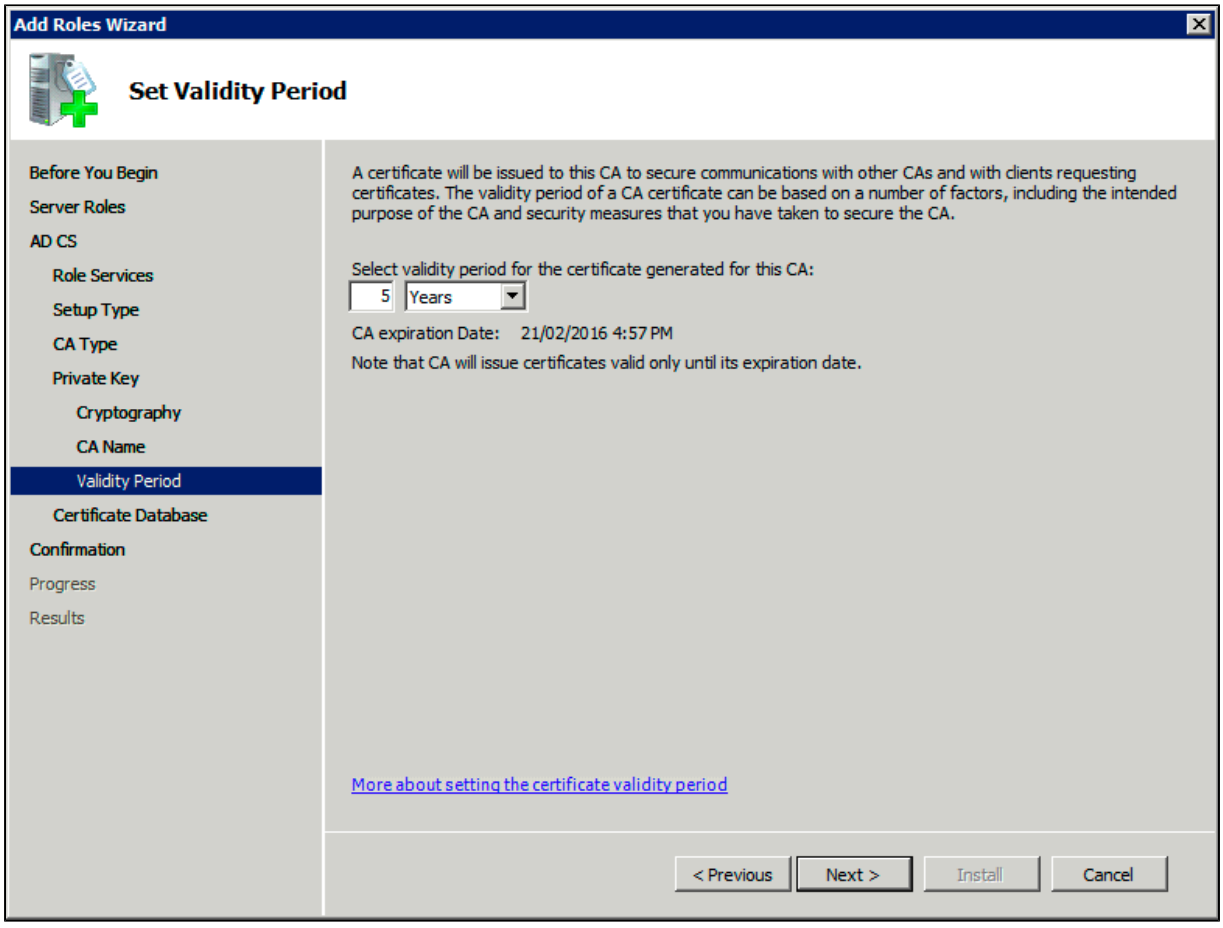
configuration settings, including cryptographic service providers. However, the default values should be fine. Click **Next** twice.

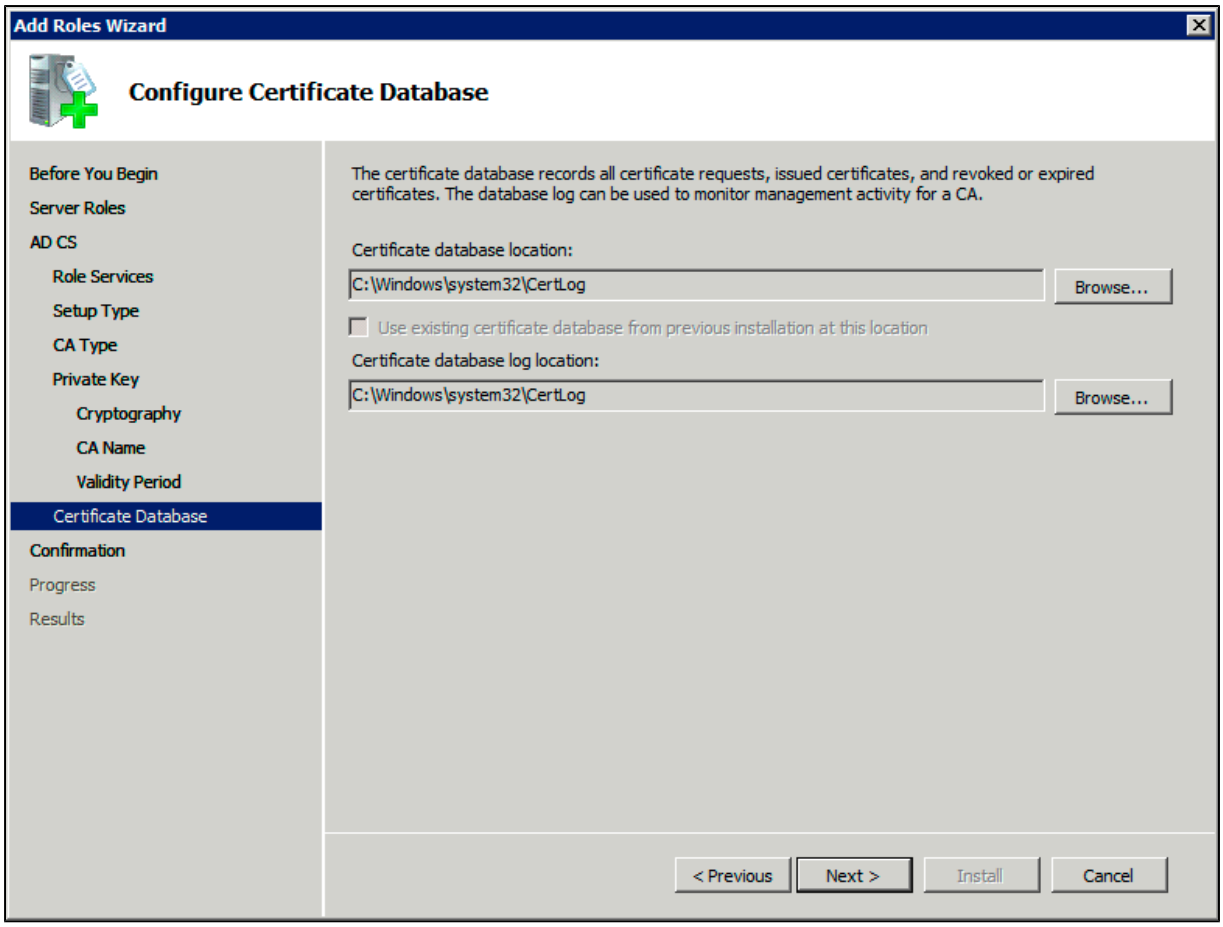


9. In the **Common name for this CA** box, type the common name of the CA, and then click **Next**.

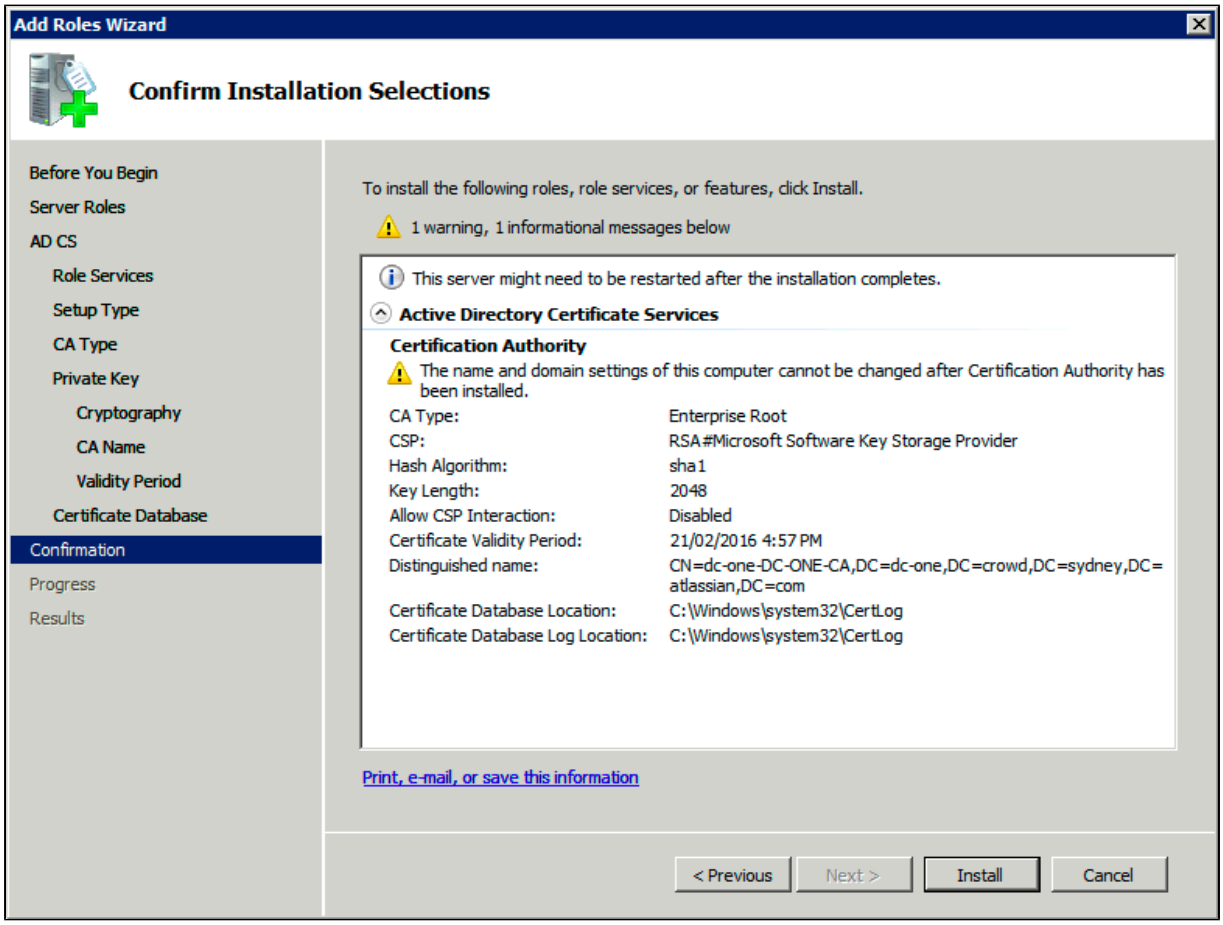


10. On the **Set Validity Period** page, accept the default values or specify other storage locations for the certificate database and the certificate database log, and then click **Next**.

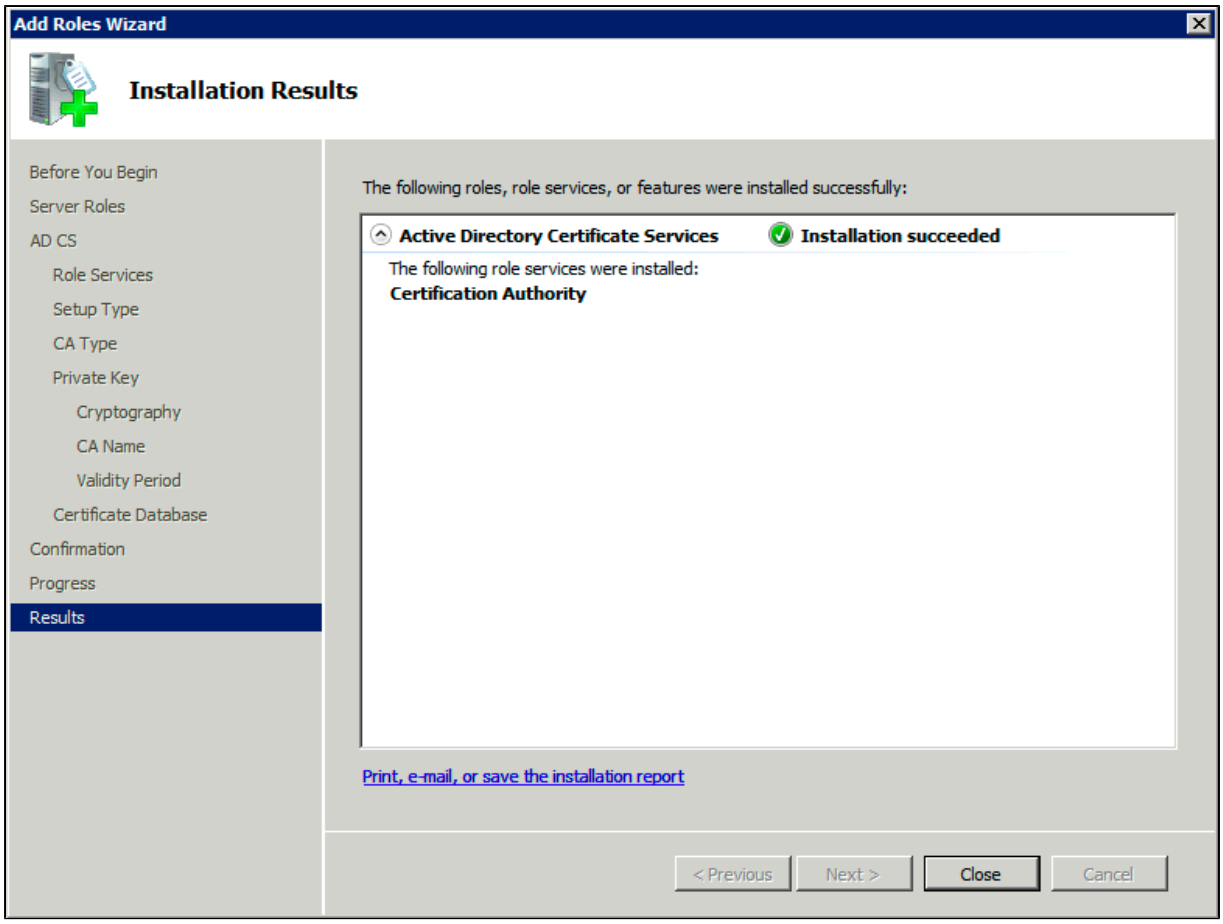




11. After verifying the information on the **Confirm Installation Selections** page, click **Install**.



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 in being authenticated with the certificate file above. In this case, Microsoft's LDAP over SSL (LDAPS) Certificate page might help. Be noted that you need:

1. to 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. to 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 *(not recommended)* — 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.

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.

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

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

Note: The option to **select a directory type** is available only in JIRA 4.3.3 and later.

Setting	Description
Name	A descriptive name that will help you to identify the directory. Examples: • `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: • `Microsoft Active Directory` • `OpenDS` • And more.
Hostname	The host name of your directory server. Examples: • `ad.example.com` • `ldap.example.com` • `opends.example.com`
Port	The port on which your directory server is listening. Examples: • `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: • `cn=administrator,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

Note: The option to **copy users on first login** is available only in JIRA 4.3.3 and later. It currently copies the data across whenever a user logs in, as per the bug

JRA-27541 - Delegated LDAP copy user on first login problem
RESOLVED
.

Setting	Description
Copy User on Login	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 synchronised 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. If you check this box the following additional fields will appear on the screen, which are described in more detail below: • Default Group Memberships • Synchronise Group Memberships • User Schema Settings (described in a separate section below)
Default Group Memberships	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. Please note that there is no validation of the group names. If you mis-type the group name, authorisation failures will result – users will not be able to access the applications or functionality based on the intended group name. Examples: • `confluence-users` • `bamboo-users,jira-administrators,jira-core-users`
Synchronise Group Memberships	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 synchronised with the internal directory each time the user logs in. If you check this box the following additional fields will appear on the screen, both described in more detail below: • 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: • `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 `ldp.exe` 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: • `cn` • `sAMAccountName`

User schema settings (used when copying users on first login)

Note: The **user schema settings** are available only in JIRA 4.3.3 and later.

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: • `ou=Users`
User Object Class	This is the name of the class used for the LDAP user object. Example: • `user`
User Object Filter	The filter to use when searching user objects. Example: • `(&(objectCategory=Person)(sAMAccountName=*))`
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: • `cn`
User First Name Attribute	The attribute field to use when loading the user's first name. Example: • `givenName`
User Last Name Attribute	The attribute field to use when loading the user's last name. Example: • `sn`
User Display Name Attribute	The attribute field to use when loading the user's full name. Example: • `displayName`
User Email Attribute	The attribute field to use when loading the user's email address. Example: • `mail`

Group schema settings (used when enabling 'synchronise group memberships')

Setting	Description
Group Object Class	This is the name of the class used for the LDAP group object. Examples: • `groupOfUniqueNames` • `group`
Group Object Filter	The filter to use when searching group objects. Example: • `(&(objectClass=group)(cn=*))`
Group Name Attribute	The attribute field to use when loading the group's name. Example: • `cn`
Group Description Attribute	The attribute field to use when loading the group's description. Example: • `description`

Diagrams of possible configurations

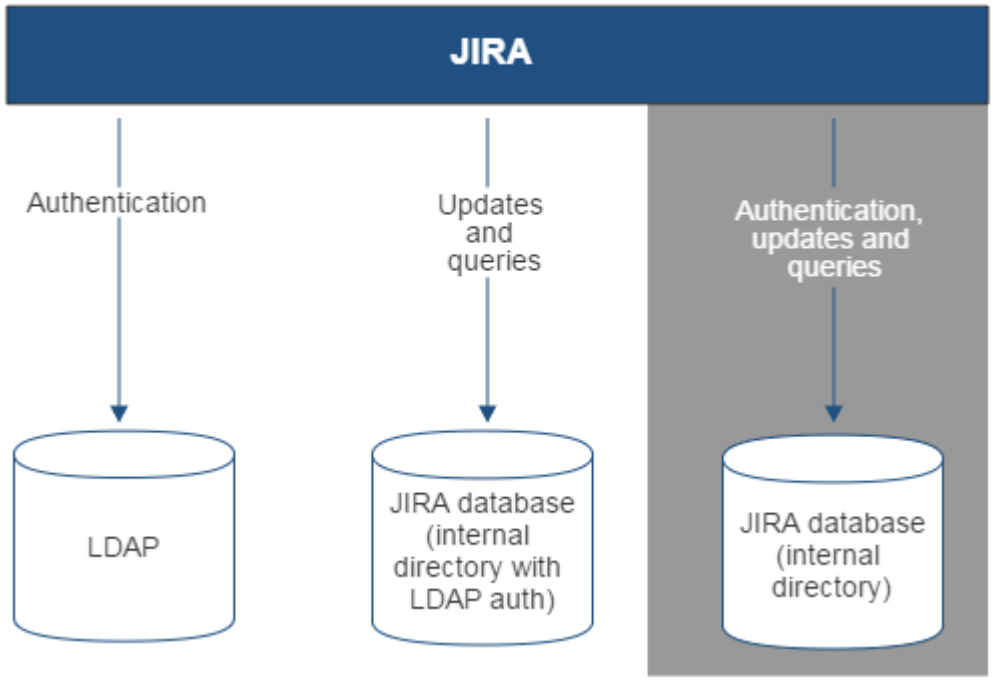


Diagram above: JIRA connecting to an LDAP directory for authentication only.

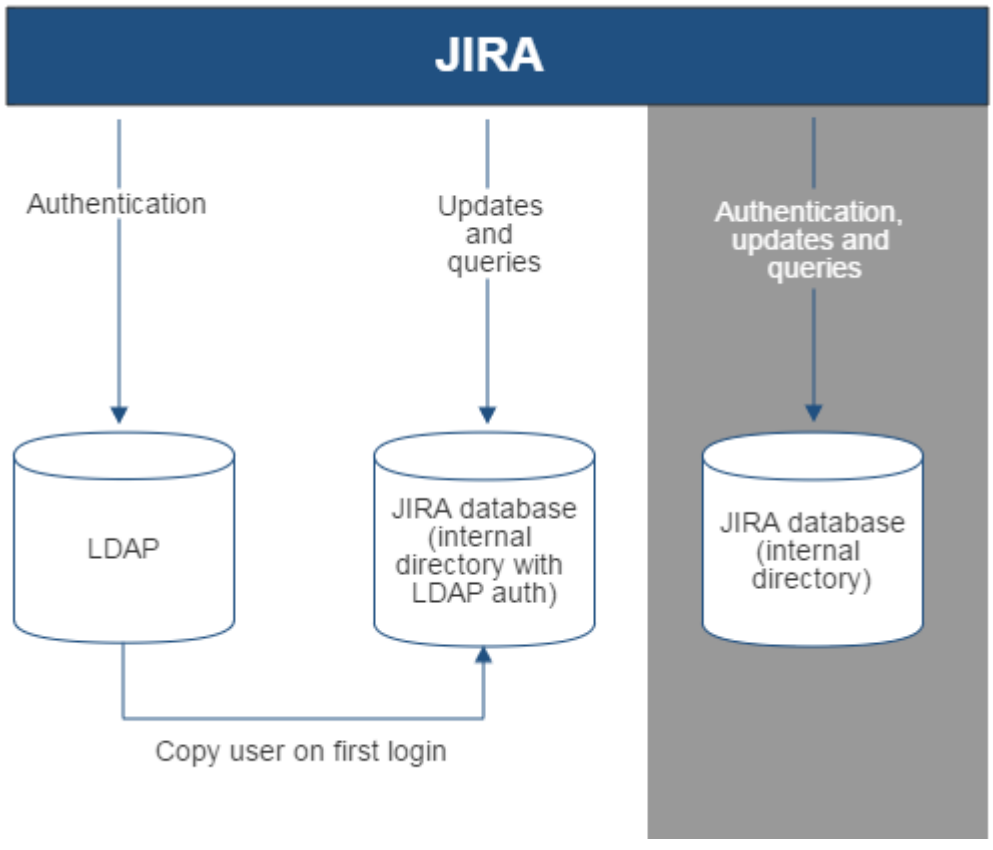


Diagram above: JIRA connecting to an LDAP directory for authentication only, with each user copied to the internal directory when they first log in to JIRA.

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

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 Crowd 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: • `Crowd Server` • `Example Company Crowd`
Server URL	The web address of your Crowd console server. Examples: • `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.
Synchronisation Interval (minutes)	Synchronisation 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 centralised 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 ins**

tance 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:

1. 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.
2. 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: • `JIRA Service Desk Server` • `My Company JIRA`
Server URL	The web address of your JIRA server. Examples: • `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.

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.
Synchronisation Interval (minutes)	Synchronisation 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

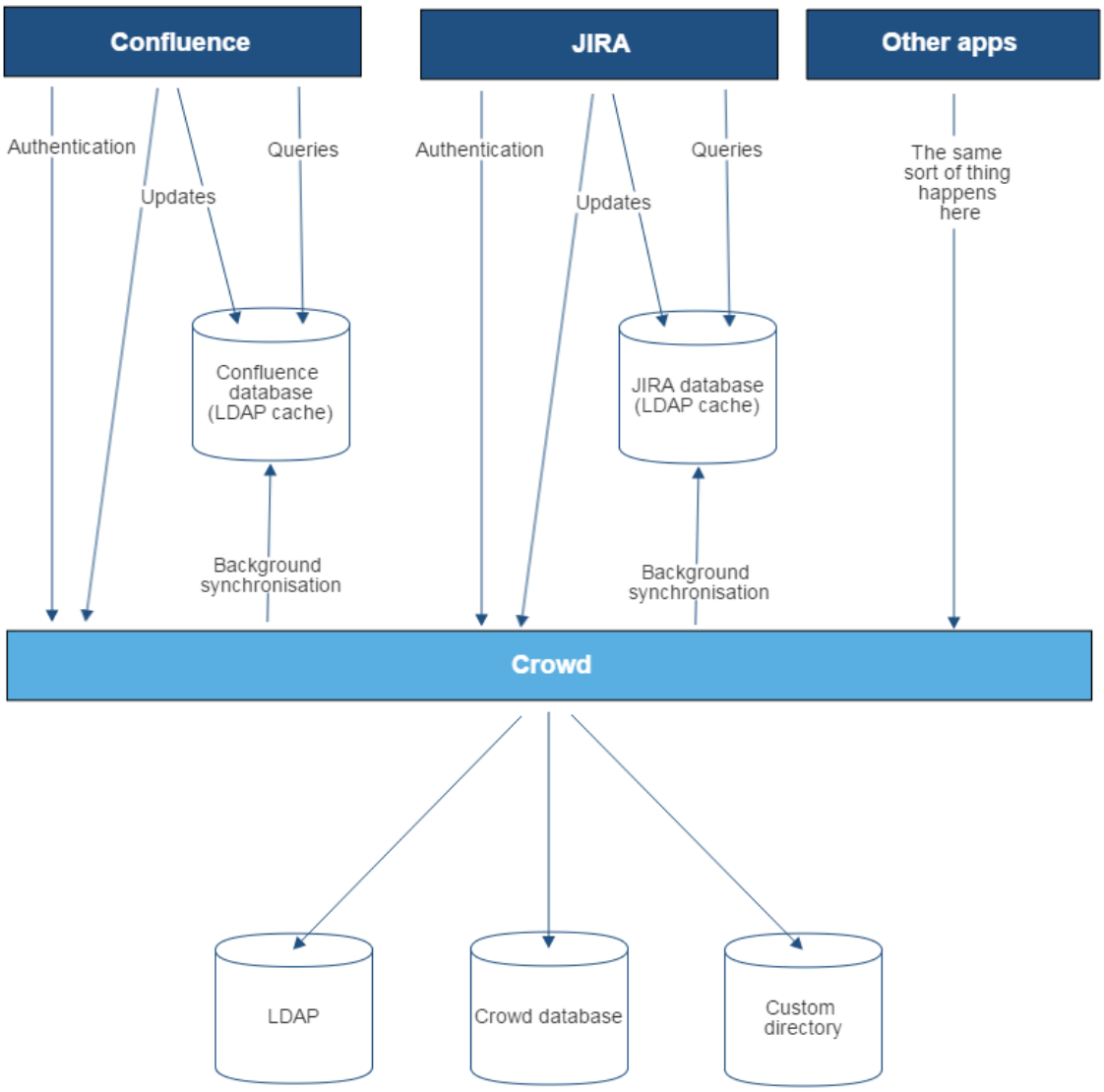


Diagram above: Confluence, JIRA and other applications connecting to Crowd for user management.

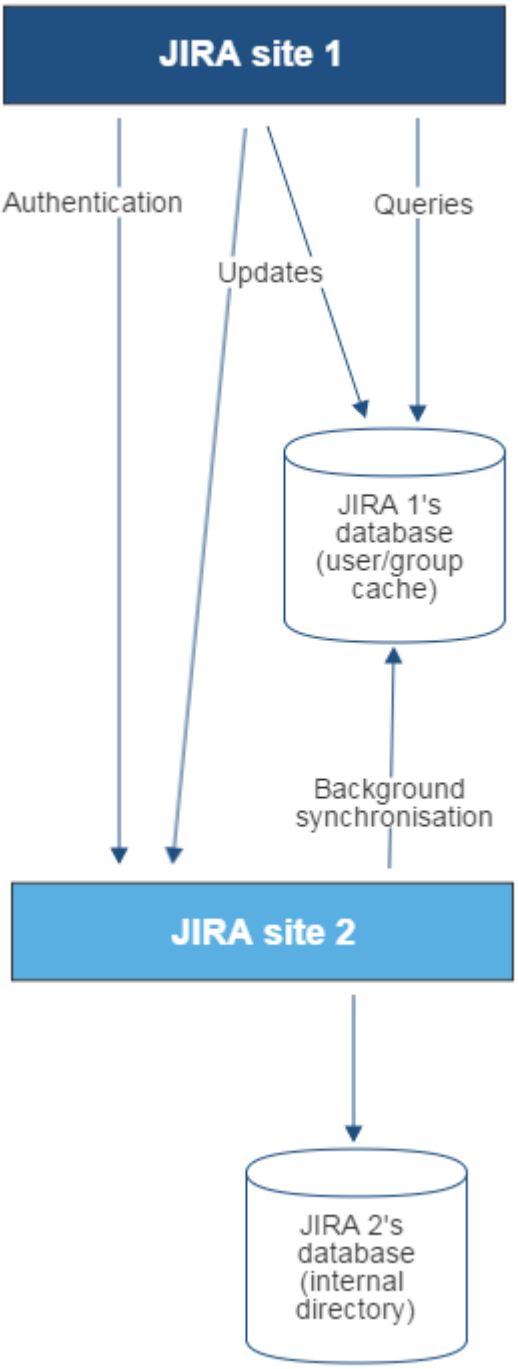


Diagram above: One JIRA site connecting to another for user management. JIRA site 2 does the user management, storing the user data in its internal directory.

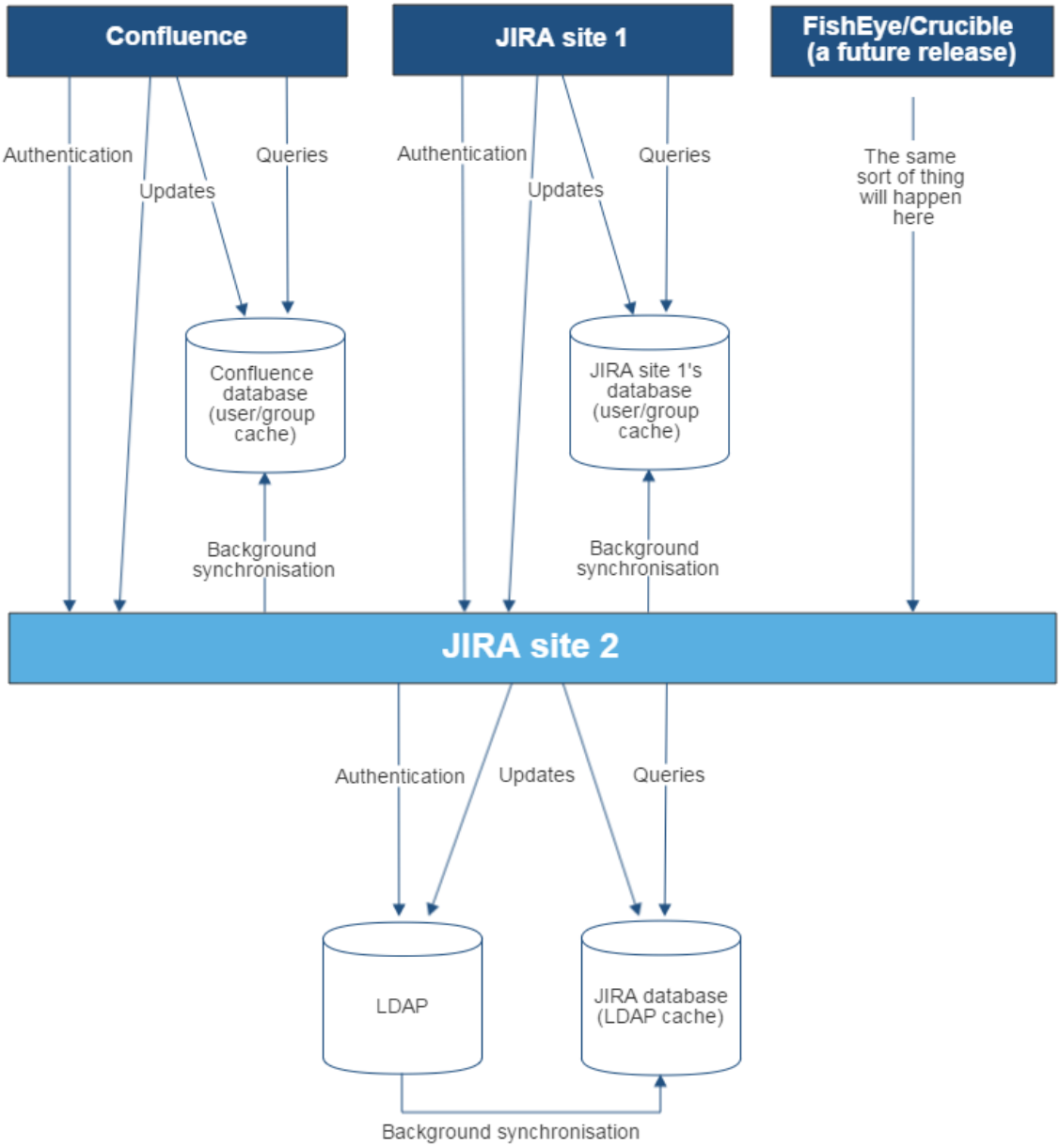


Diagram above: A number of applications connecting to JIRA (site 2) for user management, with JIRA in turn connecting to an LDAP server.

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.

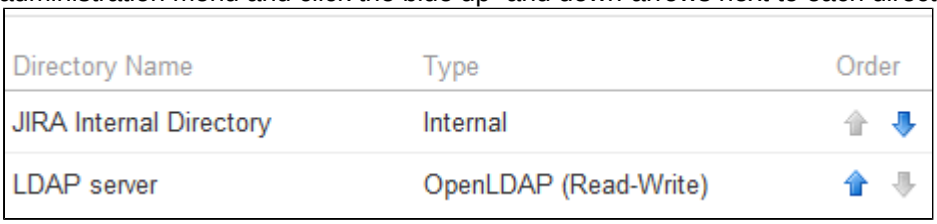
Here is a summary of how the directory order affects the processing:

- 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

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 summarises 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.
- All new users will be added to the LDAP directory. It is not possible to add a new user to the 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

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.

On this page:

- 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 th at 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.

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:

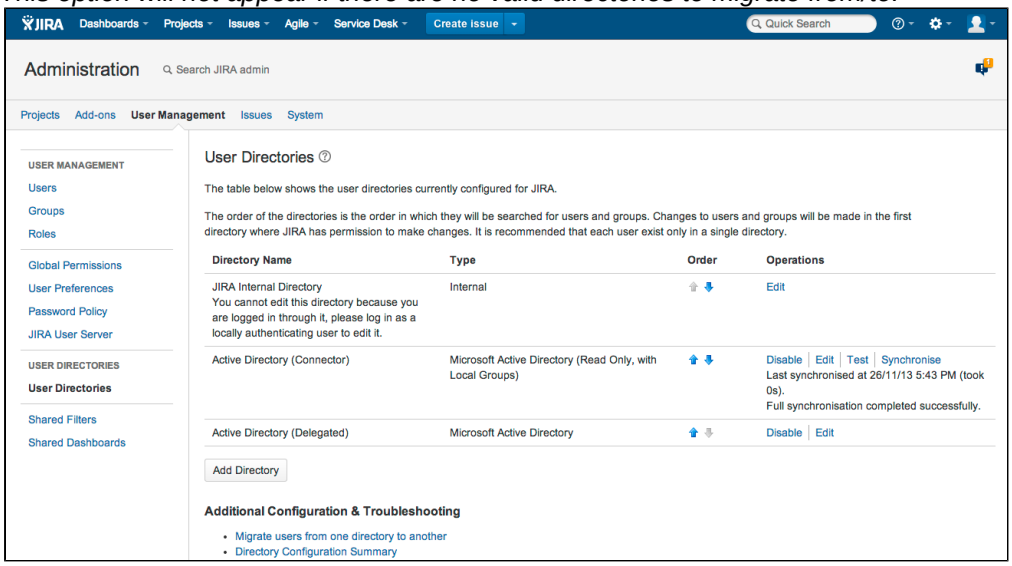
- Migrating a specific set of users or one single user from one directory to another
- Connector user directories — these can be easily identified, as they have a Synchronize option
- Migrating groups only
- Migrating users without their groups

It also has the following features:

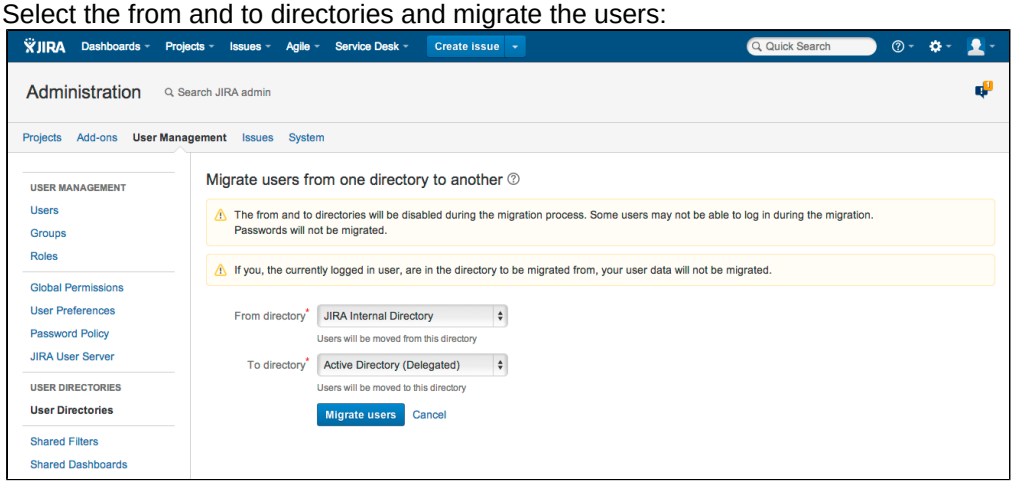
- If you, the currently logged-in user, are in the directory to be migrated from, your user data will **not** be migrated.
- Users and groups will not be migrated if they already exist in the target directory. For example, consider a user that exists in JIRA Internal and JIRA Delegated LDAP but has different 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:

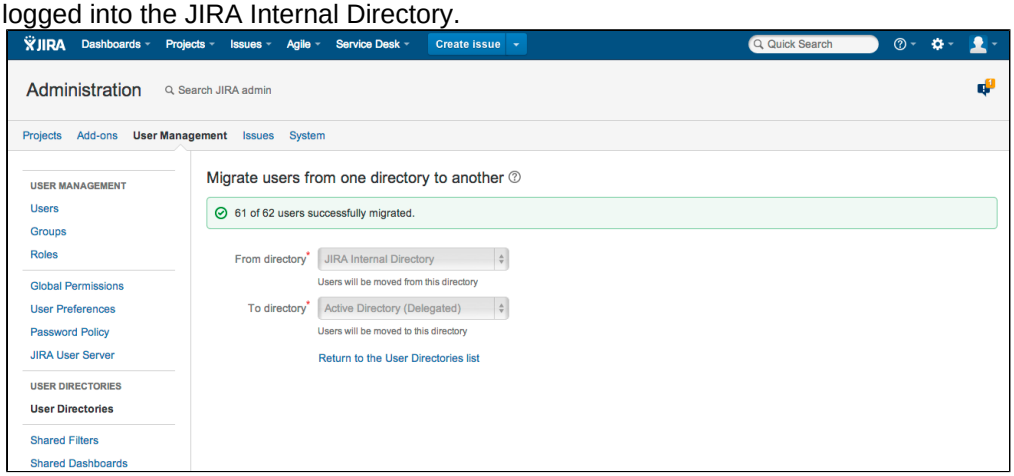
1. If the username needs to be changed as part of the migration, rename them (see Managing users for instructions).
2. Choose  **> User Management**.
3. Choose **User Directories**.
4. Choose **Additional Configuration & Troubleshooting** (section) > **Migrate users from one directory to another**.
5. *This option will not appear if there are no valid directories to migrate from/to.*



6. Select the from and to directories and migrate the users:



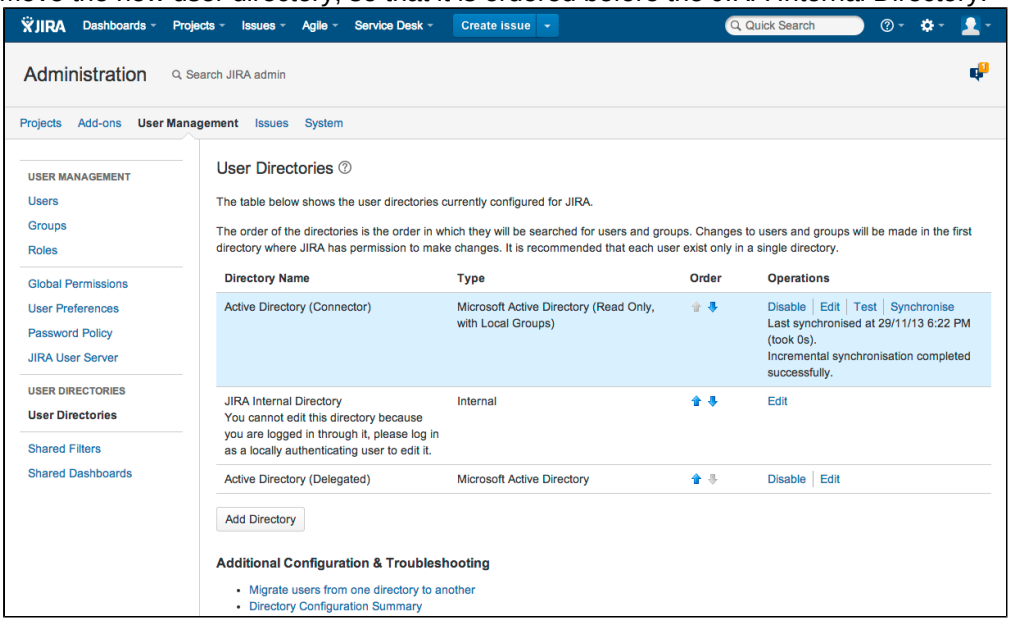
7. You will be shown a message telling you whether the migration was successful or not. In these example screenshots, only 61 out of 62 users could be migrated, as the user doing the migration was logged into the JIRA Internal Directory.



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:



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 JRA-27868 - Migrating users from one directory to another (part 2) OPEN 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

Data caching and synchronisation 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 synchronisation 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 synchronisation 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 synchronisation task has copied that user's details into the cache.

- A periodic synchronisation task will run to update the database with any changes made to the external directory. The default synchronisation interval, or polling interval, is one hour (60 minutes). You can change the synchronisation interval on the directory configuration screen.
- You can manually synchronise 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 Synchronise

The '**User Directories**' screen shows information about the last synchronisation operation, including the length of time it took.

Manually Synchronising the Cache

You can manually synchronise the cache by clicking '**Synchronise**' on the '**User Directories**' screen. If a synchronisation operation is already in progress, you cannot start another until the first has finished.

Screen snippet: User directories, showing information about synchronisation

Configuring the Synchronization Interval

You can set the '**Synchronisation Interval**' on the directory configuration screen. The synchronisation interval is the period of time to wait between requests for updates from the directory server.

The length you choose for your synchronisation 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 synchronise more frequently, then your data will be more up to date. The downside of synchronising 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

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 fresh set of schemes. These schemes are:
  - a permission scheme
  - a notification scheme
  - an issue security scheme
  - a workflow scheme
  - an issue type scheme
  - an issue type screen scheme
  - a field configuration scheme
- Sometimes you may wish to share schemes among your projects, so that editing one scheme changes that scheme in several projects at once.
- When you are creating a new project, 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**..
2. Locate the project that you'd like to change.
3. Select **Change project type** in the Operations column of the project you want to change.
4. Follow the wizard to change the project type. 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.

Configuring a project

1. Navigate to the administration page for the project by doing either of the following:
   - Choose  **> Projects**. Select your project.

- Navigate to the desired project's summary and click the **Administration** tab.

2. Use the tabs 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 **Edit Project** at the top of the **Project Summary** page and edit the project details as desired. 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 HTML in the project description: You can include HTML, but make sure all your HTML is valid. Please be aware that this is ***completely unfiltered HTML*** and as such, it is susceptible to cross-site scripting attacks.
- 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:

The project category is not edited in the Edit Project dialog. Instead, click the link next to the **Category** field (under the project name) on the project Administration page. Categories can be viewed/created via **Administration > 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

- Your desired project key must confirm 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. Navigate to the desired project, and access the project administration screen. Choose **Project Administration** (bottom of the project navigation sidebar) if you're on the Project summary screen.
2. Choose the **Edit Project** button.
3. Choose **edit key** next to the **Key** field.
4. Update the key and choose **Update**.

Note:
- If you update any other Edit Project 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

- 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	`([A-Z]{2}[0-9]{2})`	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

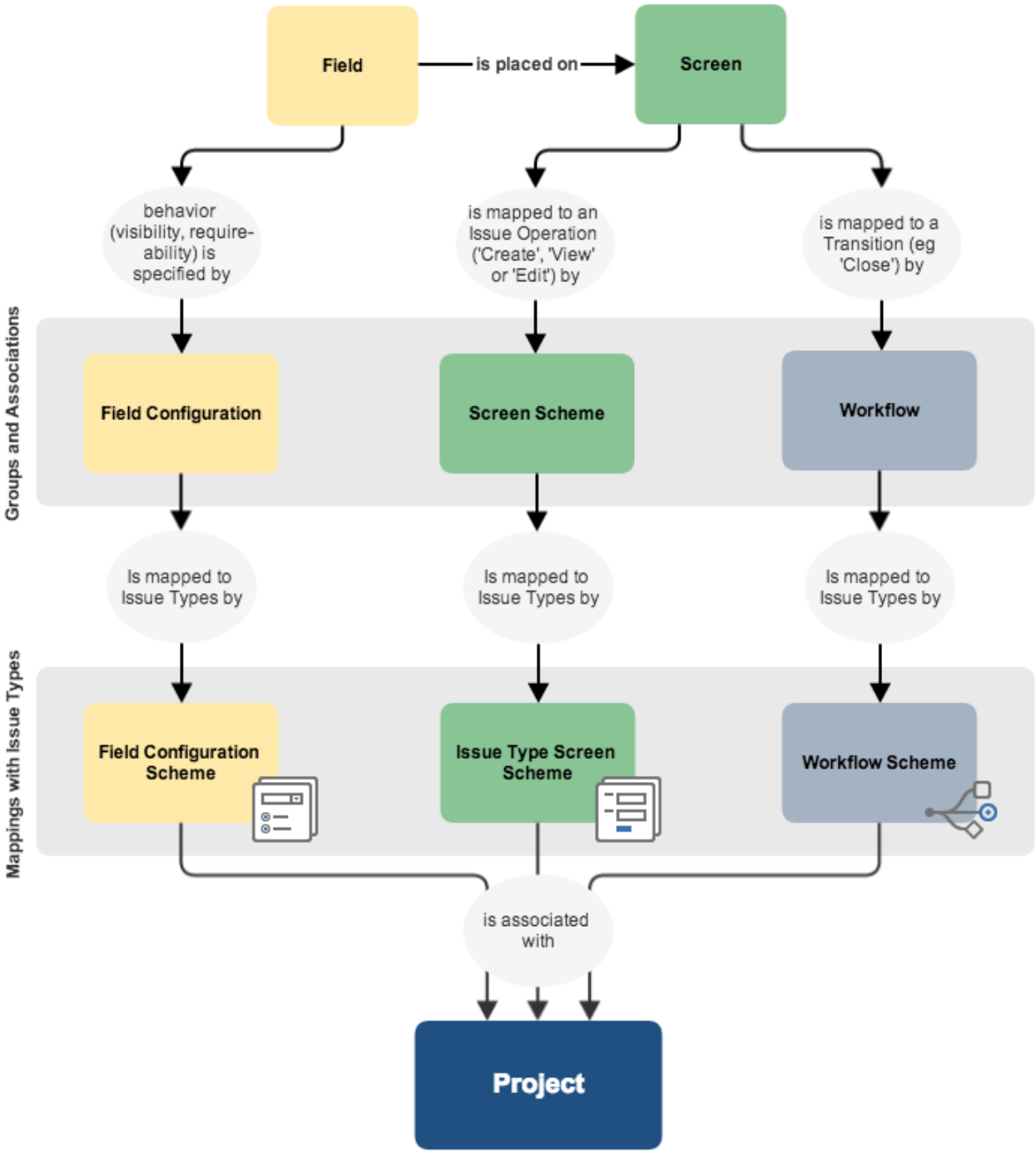
- **Changing the maximum project key length** — You can change the maximum characters allowed for a project key. Navigate to the General Configuration page of the JIRA administration console, as described on Configuring JIRA application options, and change the **Maximum project key size** field.
- Defining 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 workflow 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.
  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
  - 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
- Defining resolution field values
- Defining status field values
- Translating resolutions, priorities, statuses, and issue types

Defining issue type field values

On this page:
- Creating an issue type
- Deleting an issue type
- Editing an issue type

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.
  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.

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.
   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.

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.
  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:

Hence, you could create one scheme called *Development Issue Type Scheme* (with issue types *Bug* and *Fe ature*) 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*.

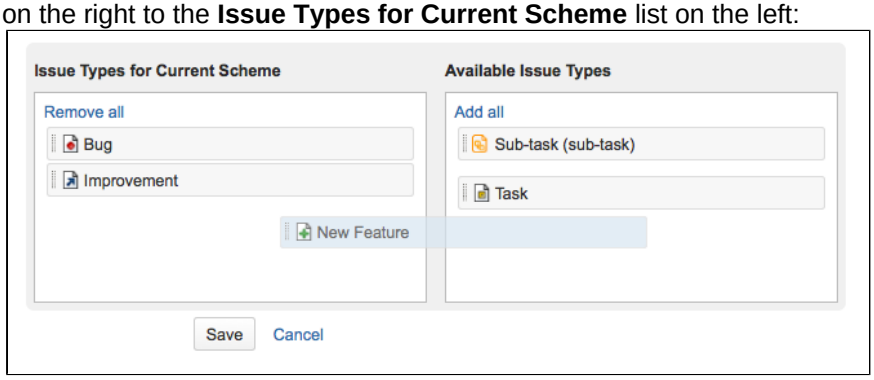
⚠ **Note:** For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

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 **I ssue 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.

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.

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.

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**.
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.

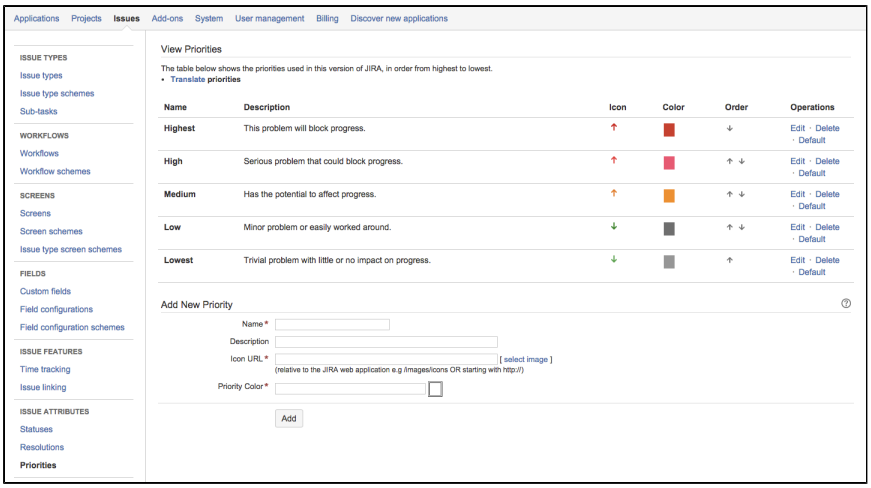
Defining priority field values

An issue's *priority* defines its importance in relation to other issues. JIRA applications come with a set of default priorities, which you can modify or add to, as explained on this page. To translate your priorities into another language, please see Translating resolutions, priorities, statuses, and issue types.

> For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

Defining a new priority

1. Choose  **> Issues**.
2. Select **Priorities** from the left menu to view the currently-defined priorities and the **Add New Priority** form :



3. Complete the **Add New Priority** form at the bottom of the page with the following fields:
    - **Name** — specify a word or two to describe your new priority. This name will appear in the drop-down field when a user creates or edits an issue.
    - **Description** — add a sentence or two to describe when this priority should be used.
    - **Priority Color** — 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.
4. Select **Add**.

Editing a priority

1. Choose  **> Issues**.
2. Select **Priorities** from the left menu.
3. From the Operations column, select the **Edit** link corresponding to the priority you wish to revise.
4. Update the fields as described under **Defining a new priority** (above), then select **Update**.

Re-ordering priorities

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

1. Choose  **> Issues**.
2. Select **Priorities** from the left menu.
3. To re-order the priorities, click the arrows in the **Order** column:
    - Click the up arrow to move a priority higher up in the list.
    - Click the down arrow to move a priority lower down in the list.

Deleting a priority

1. Choose  **> Issues**.
2. Select **Priorities** from the left menu.
3. From the Operations column, click the **Delete** link corresponding to the priority you wish to delete.

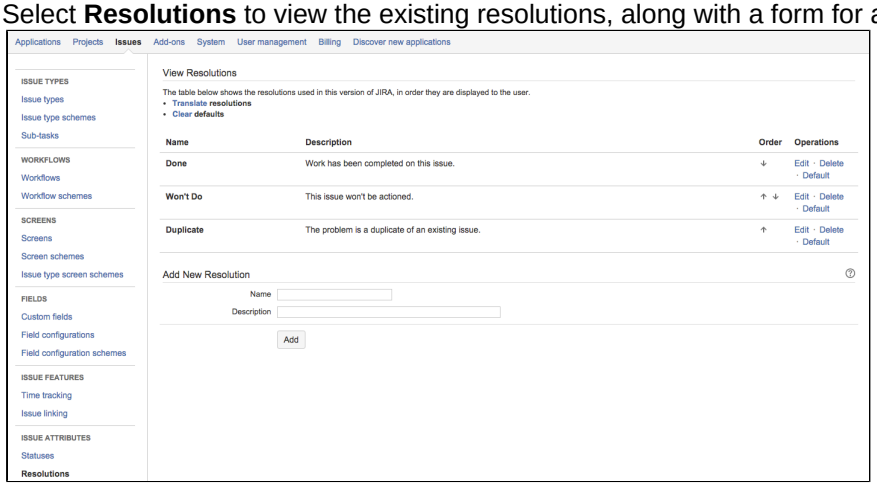
Defining resolution field values

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:


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.

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:

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

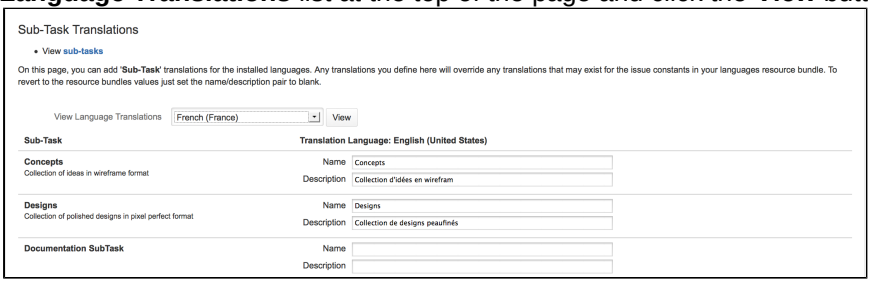
This allows you to specify a translation set for each available language — providing each user with a more complete translation in their own chosen language. The translated field names and descriptions appear throughout JIRA, e.g. in reports, gadgets and all issue views.

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) *(if applicable)*	Project version(s) for which the issue is (or was) manifesting.
Fix Version(s) *(if applicable)*	Project version(s) in which the issue was (or will be) fixed.
Component(s) *(if applicable)*	Project component(s) to which this issue relates.
Labels *(if applicable)*	Labels to which this issue relates.
Environment *(if applicable)*	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 *(if applicable)*	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 additonal 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

- 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 Operations 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 Operations 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 Operations row.
    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** pa ge.
4. Locate the appropriate security level and click its **Add** link (in the **Operations** 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**.
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** l abel 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 **Operations** 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 **Operations** 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 **Operations** 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:

On this page:

- permissions within JIRA applications themselves
- security in the external environment

Configuring permissions within JIRA applications

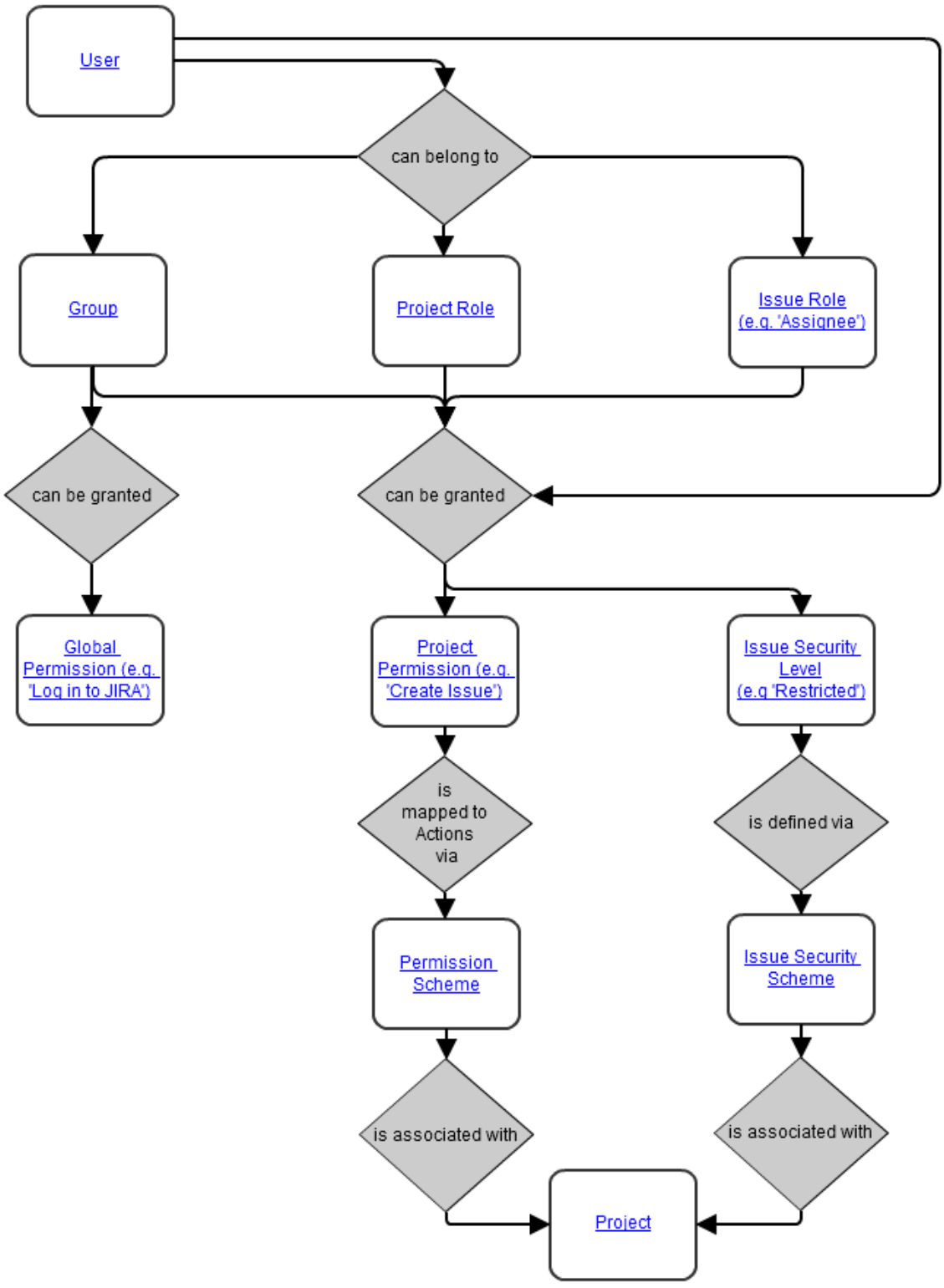
JIRA applications have a flexible security system which allows you to configure who can access JIRA applications, and what they can do/see within them.

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 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.

- 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 for manage permissions that apply to in dividual projects.

> For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

On this page:

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 my have restricted functions depending on their application access.
Browse Users	Permission to view a list of all JIRA user names and group names. 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 *all* accessible issues. Undoing such modifications may not be possible through the JIRA application interface and may require changes made directly against the database (which is not recommended).

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.

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

- Configure JIRA's SMTP mail server for notifications (but *they can* configure POP/IMAP mail servers fo r the receipt of email messages that create issue comments and new issues, and fully administer emai l 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).
- Change the index path (but *they can* reindex and optimise the index).
- Run the integrity checker.
- Access logging and profiling information.
- Access the scheduler.
- Export/backup JIRA data to XML.
- Import/restore JIRA data from XML.
- Import XML workflows into JIRA.
- Configure attachments (but *they can* set the size limits of attachments and enable thumbnails).
- 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 **JIR A 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  **> Add-ons**.
2. 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**.

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:

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').
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)	Permission to perform the following sprint-related actions for all projects in a board: • 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 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.
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' (a bove).
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.
2. Select the project of interest to open the **Project Summary** administration page for that project. See D efining 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.

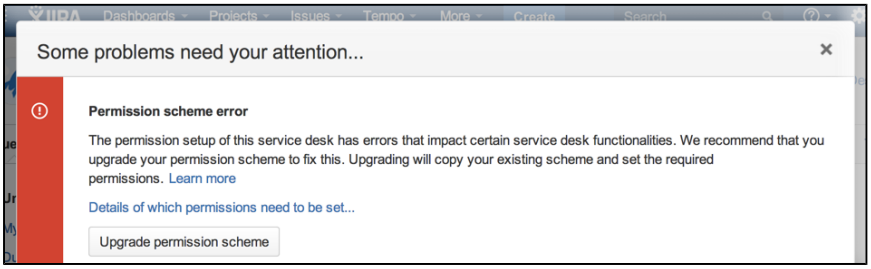
Mandatory permissions by project roles

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	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. 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.
Service Desk Team	• 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	The permissions for customers must be granted to the **Service Desk Customer - Portal Access** security type instead of the **Service Desk Customers** role. This configuration gives customers access to the Customer Portal only (not JIRA). The security type reads the role to determine who are customers. • Create Issues (This permission gives users the ability to create requests in a Customer Portal) • Browse Projects (This permission gives users access to the project in the Customer Portal) • Add Comments (This permission gives users the ability to add comments on their own requests.) • Create Attachments (This permission gives users the ability to add attachments when they create a request or add attachments to the request after it's been submitted) • Assign Issue (This permission is mandatory for the **Assignee** field to work. The **Assignee** field is an optional hidden field and it automatically channel issues to certain team members.) 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.

Resolving JIRA Service Desk permission errors

When you use a custom permission scheme, if the permission settings are different from those of the standard permission scheme, you will see a permission error similar to the following:



Explanation of permission scheme 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):** These errors either cause certain administration functionality to be disabled (for example you cannot add agents to your service desk), or impact the day-to-day use of your service desk (for example customers cannot log in to the Customer Portal). This table describes what JIRA Service Desk considers as critical errors. You cannot dismiss these error messages, as you must fix them in order for JIRA Service Desk to return to normal operation.
- **Non-critical errors (yellow):** Permission scheme differences that do not impact how JIRA Service Desk works are considered as non-critical errors. You can dismiss these error messages if you do not want to use the standard permission setup.

Resolving errors

You can resolve the permission errors by changing the permission scheme yourself or using the **Fix permissions** button in the error message.

What does the Fix permissions button do?

The **Fix permissions** button on the message disassociates your custom permission scheme with the service desk project, creates a copy of your permission scheme with the name of <your_permission_scheme *[number]*>, and associates this new scheme with the project. The new scheme fixes the errors by:

- Granting the standard permissions to the **Administrators** and **Service Desk Team** roles, and the **Service Desk Customer - Portal Access** security type as described on the Standard permissions page.
- Removing the **Service Desk Customers** role from all the permissions assigned.
- Leaving other permission setup as is.

Your original permission scheme	The new permission scheme

The name of the original one is 'JIRA Service Desk Permission scheme for Project OA'. The following permissions are set up differently from the standard permission scheme: • 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.	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. • 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.

What are critical permission errors?

Critical permission errors cause certain functionality of JIRA Service Desk to be disabled.

Error	Explanation
The **Administrators** role does not have the following required permissions: • Browse Projects • Administer Projects • Edit Issues	• 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.
The **Service Desk Customer - Portal Access** security type does not have the following required permissions: • Browse Projects • Create Issues • Add Comments	• 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.
The **Service Desk Customers** role is granted any permission directly.	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. 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.
The **Service Desk Team** role does not have the following required permissions: • Browse Projects • Edit Issues	• No **Browse Projects** permission = Agents cannot see the service desk. • No **Edit Issues** permission = Agents cannot edit issues.

The **Service Desk Team** role is granted the Administer Projects permission.	Granting the Administer Projects permission to your agents means that all agents become administrators for your service desk. 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.

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.

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 *only two projects*.

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>)`

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:

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.

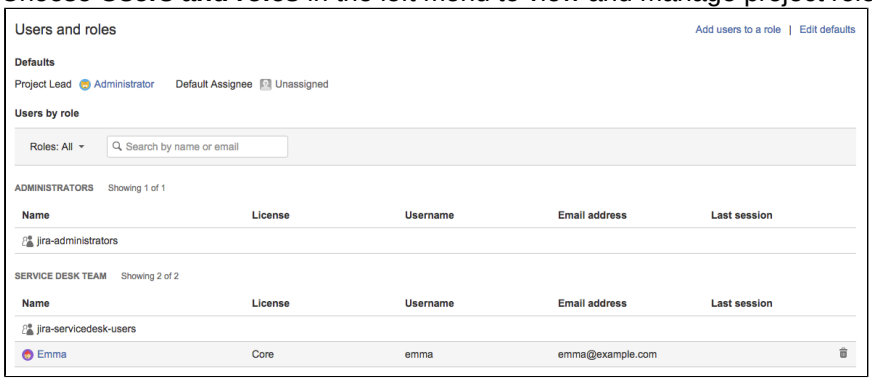
This page contains instructions for managing membership of *existing project roles*. For information on creating and using project roles, see Managing project roles.

On this page:

> For all of the following procedures, you must be logged in to JIRA as a project administrator.

Viewing project role members

1. Choose  **> Projects**.
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. There are two related actions:

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.

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:

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**.
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 pl anned release date for a version) can be 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** fr om 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** fr om 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 un-archived versions.
   In the 'Merging From Versions' select list, choose the version(s) whose issues you wish to move. Versi ons 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** fr om 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.

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.

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Project Administrator** global permission.

On this page:

Managing a project's components

1. Choose  **> Projects**.
2. Choose **Components** in the left menu. 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** a nd **Default Assignee** (see options below).

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.
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 per mission 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, hover over the relevant component to display the pencil icon.
2. Edit the component's **Name, Description, Lead,** and **Default Assignee**.
3. Click the **Update** button to save your changes.

Deleting a component

1. On the 'Components' screen, hover over the relevant component to display the **Delete** button.
2. You will be prompted to associate these issues with another component if you wish.

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 behaviour	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.

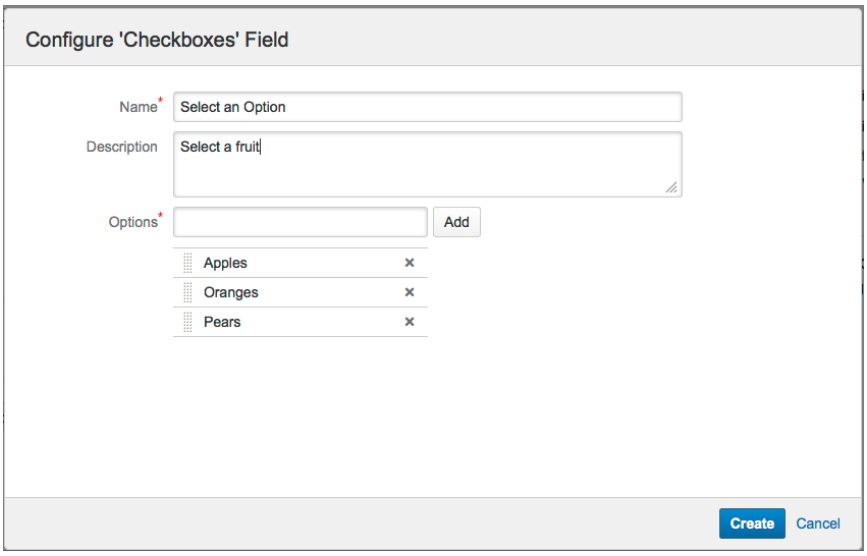
> For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

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:



> 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.

On this page:

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**.

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.
      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).
      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.

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-ADMINSITRATORS` 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.
   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**.
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.
- **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".

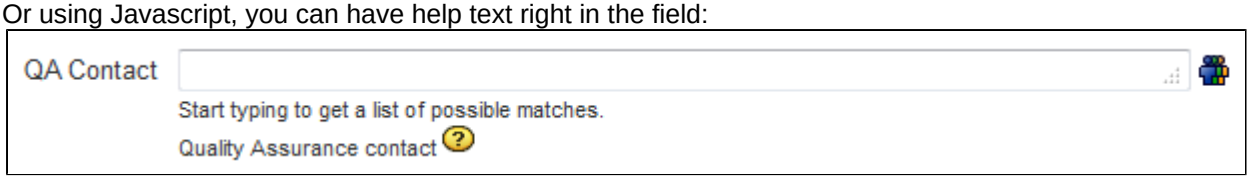
Creating help for a custom field

> Customizations to JIRA, such as including Javascript in the Custom Field description are not included in the scope of Atlassian Support.

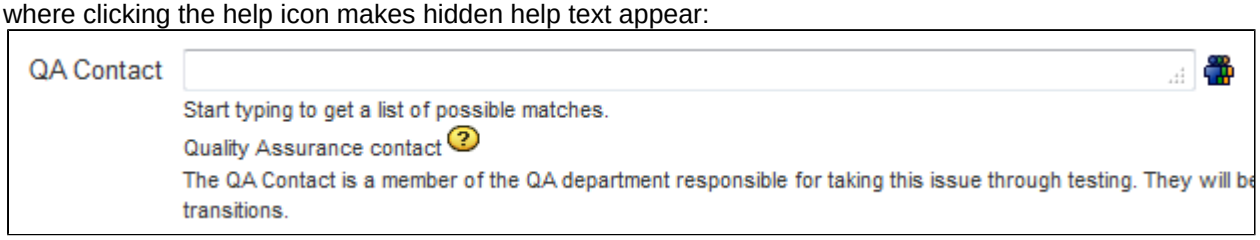
To provide online help for a custom field, use HTML or Javascript in the field's description, e.g. you can have a simple link to an external help page:

```
<a href="http://www.mycompany.com/jirahelp/fieldhelp.html">get help</a>
```

Or using Javascript, you can have help text right in the field:



where clicking the help icon makes hidden help text appear:



This can be done by entering the following as the field's description:

```
Quality Assurance contact
<script type="text/javascript">
  function showHelp() {
        var listenersDiv = document.getElementById("qaFieldHelp");
        if (listenersDiv.style.display == 'none') {
          listenersDiv.style.display = '';
        } else {
          listenersDiv.style.display='none';
        }
  }
</script>
<a href="#"  onclick="showHelp(); return false;"><img
src="/images/icons/ico_help.png"/></a>
<div id="qaFieldHelp" style="display:none">
  The QA Contact is a member of the QA department responsible for taking this issue
through testing.
  They will be notified by email of this and subsequent issue state transitions.
</div>
```

(Incidentally, Javascript in descriptions can also be used to set field values.)

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:

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.

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).
   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**.

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.
   You will be prompted to confirm this operation.

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.
   The (initial) field settings between the original and copied field configurations will be identical.

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.
   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).

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 the field does not appear on any screens (i.e. issue operation screens, workflow transition screens) where the field configuration applies.

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.
     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.
     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.

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 (JRA-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.

- 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

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 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.
   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.
   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.

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.
   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.

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.
   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.

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.
   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.

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.
   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

On this page:

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.

> 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 feature: if the text contains text that resolves to a JIRA issue key then an HTML link will be generated that points to that issue. Below is a sample of how description text looks when rendered through the default text renderer:

> 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.

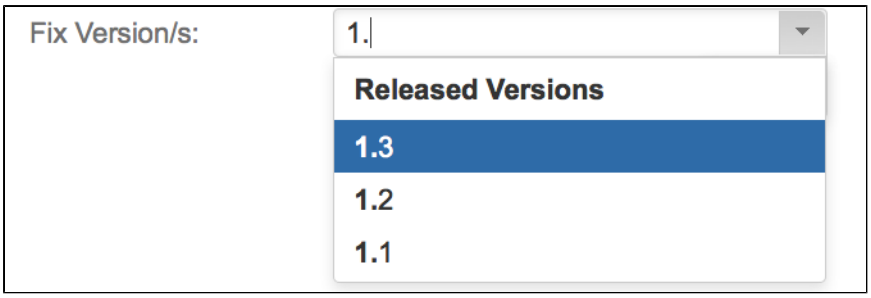
Wiki style renderer macro support

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:



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

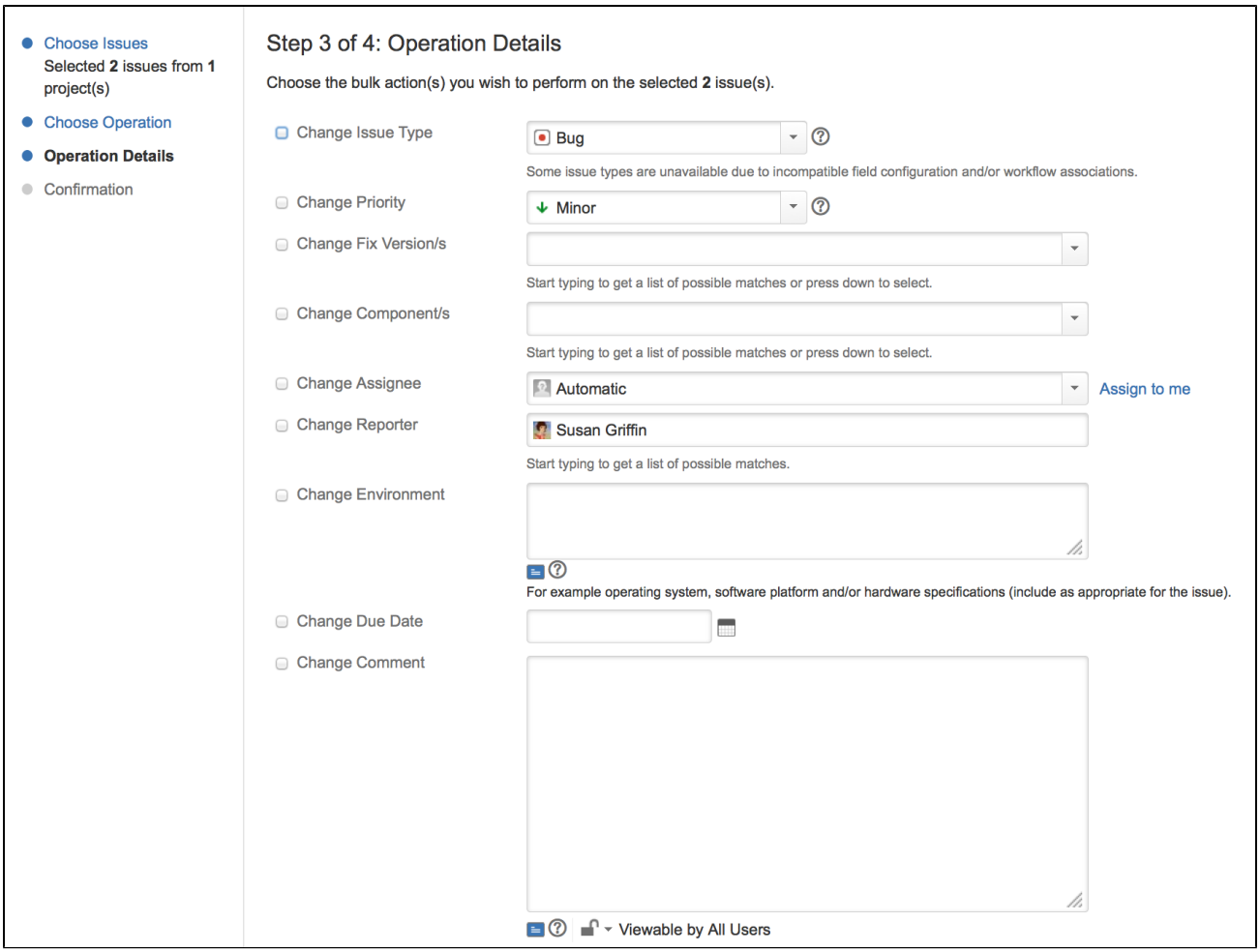
When bulk moving issues to an environment (project/issue type) that has a different renderer type defined for one of the fields being affected by the move, if any of the issues have a non empty value associated with the field, the move operation will present you with a warning so that you are aware of 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:



Email notifications

JIRA allows for extensive configuration in relation to email notifications, and cansend 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 wiew

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.

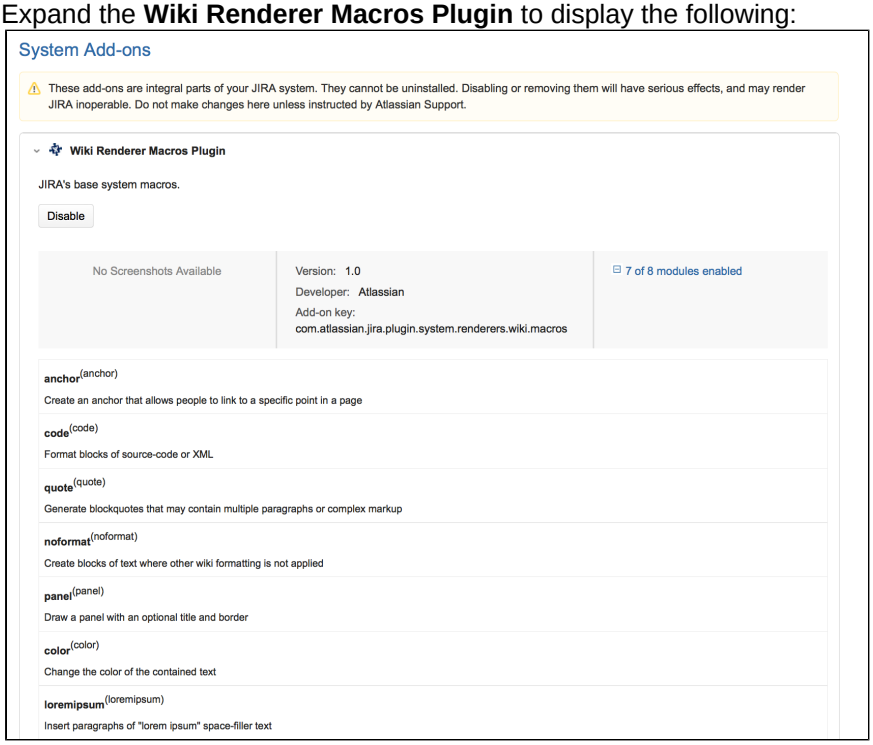
Configuring macro plugins for the wiki style renderer

The macros used by the wiki style renderer can be enabled or disabled as follows.

1. Choose 

> Add-ons.

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.
4. Expand the **Wiki Renderer Macros Plugin** to display the following:



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:

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:

1. are present on the screen associated with the issue's **Create Issue** issue operation
2. are also *not hidden* in the field configuration applicable to the issue (as defined by the project's field configuration scheme)
3. 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.
3. Click the **Copy** link next to the Screen you wish to copy. You will be directed to the **Copy Screen** pag e, where you can enter a name and a description for the new Screen.

Deleting 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.
3. Click the **Delete** link next to the screen you wish to delete. You will be prompted to confirm your deletion

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:

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 **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 clickAdd.
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	1. Hover over the tab name and click the **pencil icon**. 2. Enter the new name and click **OK**.
Delete a tab	Hover over the tab name and click the **X**.
Add a field	1. Click the tab that you want to add the field to. 2. 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. 3. Click **Add Field** to add it to the current tab.
Move a field	Hover over the dotted part of the field (next to the field name) and drag the field to the desired position. Move a field to a different tab by dragging it to the name of the tab and dropping it.
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 the m. 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.)

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:

Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** glo

bal 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.

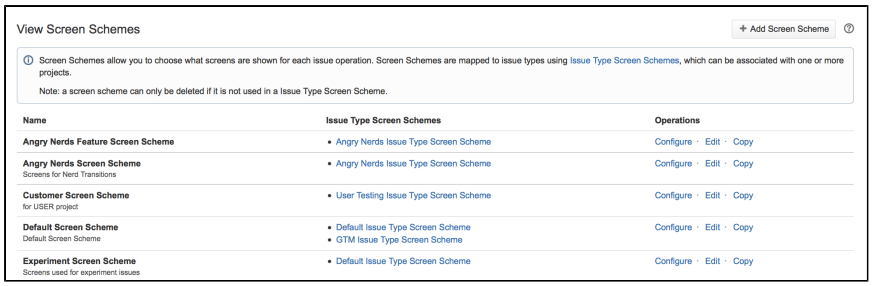
Copying a screen scheme

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:



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:

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.

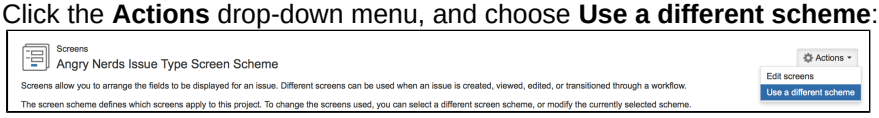
Copying an issue type screen scheme

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**.
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

On this page:

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.

You can use the same notification scheme for more than one project.

JIRA applications are pre-packaged with a notification scheme called **Defaul t Notification Scheme**. *This scheme isassociated 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 notifications 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

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**.
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.

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 *User Picker* or *Multi User Picker* 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 cust om field value	The value of a custom field of type *Group Picker* or *Multi Group Picker* that may have been associated with issues..

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

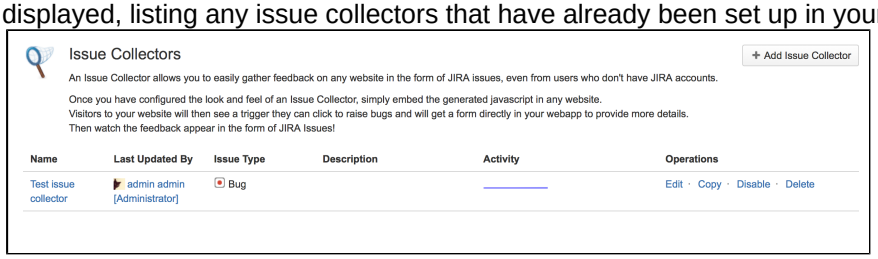
In JIRA applications, issue collectors are configured (and hence organized) on a per-project basis.

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.

To access issue collectors belonging to a specific project:

1. Choose  **> Projects**.
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:



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**.
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?	Select either of the following:
	• **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:
	  • 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.
	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.

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	Choose from the options provided. Typically, your choice would reflect the type of issue being created (i.e. chosen above). You can choose:
	• 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.
	  • Please note that if a field is *required* for the chosen issue type but that field has:
	  • 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:
	  • *Standard Fields*: Summary, Description, Components, AffectsVersion, Environment, Priority, Attachment
	  • *Custom Field Types*: 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	Type a message, which appears in the blue 'information' panel along the top of the dialog box.

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 a frameset will not close the prompt window automatically.

This is a known limitation for the Issue Collector plugin, and has been tracked at

JRA-29886 - Issue Collector Cannot Be Closed When Placed Inside a Frameset RESOLVED .

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:

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.

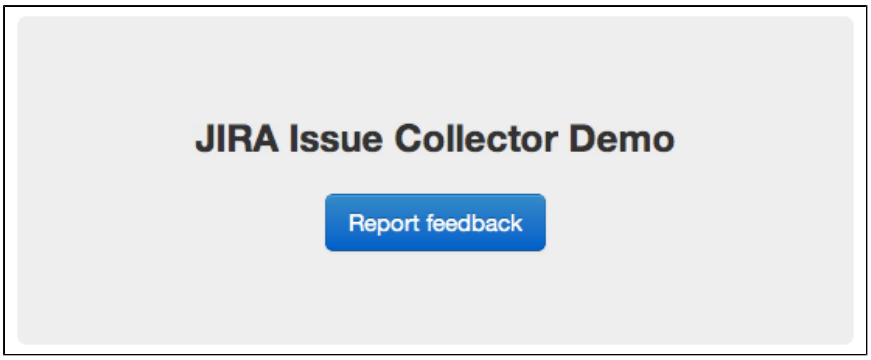
Customization of the issue collector is done by creating/extending the global object **ATL_JQ_PAGE_PROPS**. This allows you to add a custom trigger, set default values for fields and more.

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-418945332/803/10
88/1.2/_/download/batch/com.atlassia
n.jira.collector.plugin.jira-issue-c
ollector-plugin:issuecollector/com.a
tlassian.jira.collector.plugin.jira-
issue-collector-plugin:issuecollecto
r.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) {
    //Requries that jQuery is
available!

jQuery("#feedback-button").click(fun
ction(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

determine 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.jspa?projectKey=<PROJECT_KEY>&col
lectorId=<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 Include Macro. Note that using the HTML Include 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:

CONF-26104 - Some JavaScripts are not executed if included in User Macro
OPEN

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
  $('.footer-link').append(feedbackButton);

  $('#feedback-button').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>

   <!-- JIRA Issue Collector - append this at the bottom of <body> -->
 <script type="text/javascript" src="https://<JIRA
URL>/s/en_US-ydn9lh-418945332/803/1088/1.2/_/download/batch/com.atlassian.jira.co
llector.plugin.jira-issue-collector-plugin:issuecollector/com.atlassian.jira.coll
ector.plugin.jira-issue-collector-plugin:issuecollector.js?collectorId=d03d7bd1">
</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 ====
      '<collectorId_1>' : {

     // === 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

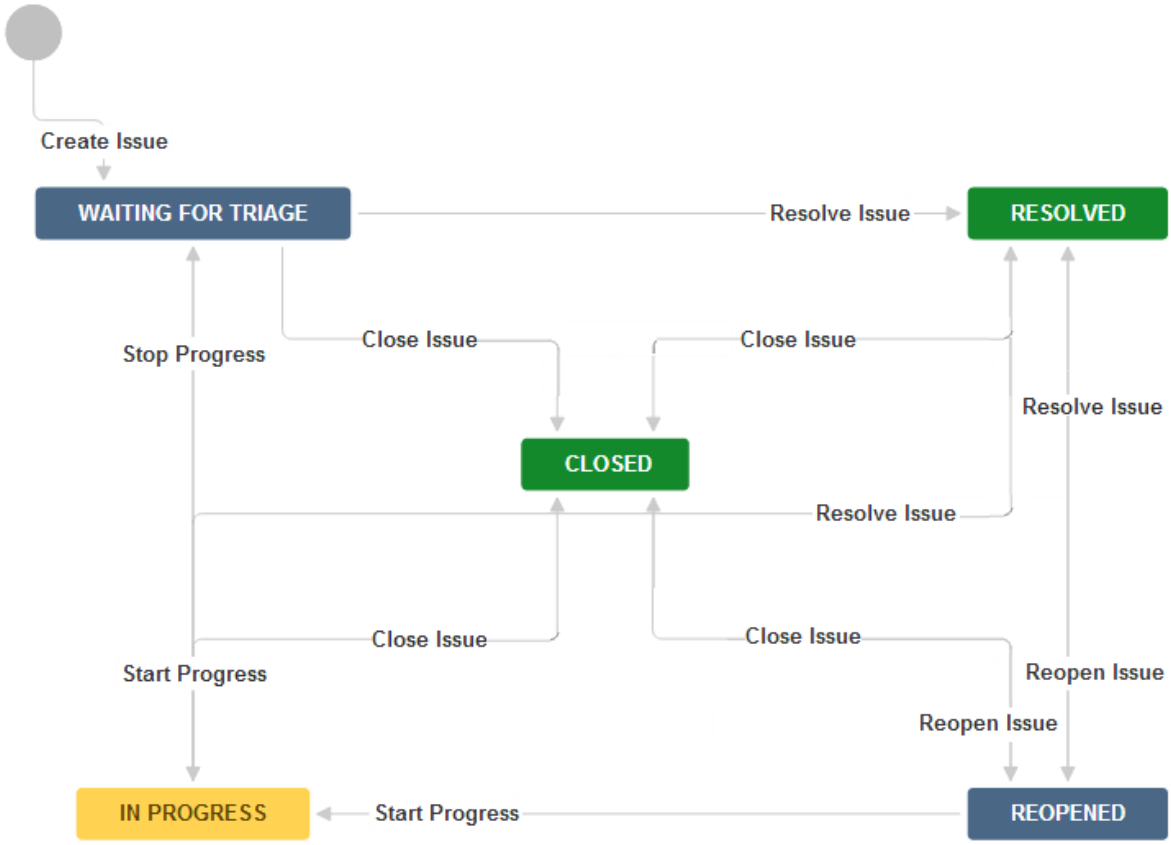
A JIRA workflow is a set of *statuses* and *transitions* that an issue moves through during its lifecycle and typically represents processes within your organization. There are default built-in workflows that cannot be edited; however, you can copy and use these workflows to create your own.

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 System Administrators' global permission to access and manage workflows.

On this page:

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 *inactive workflow* 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	An *active workflow* 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. The following limitations apply when editing the draft for an active workflow: Editing limitations... • It is not possible to edit the workflow name (only the description) if a workflow is active. • Workflow statuses cannot be deleted. • If a status has no outgoing transitions (Global transitions are not considered), it cannot have any new outgoing transitions added, regular or global. • The step ID cannot be changed. See Cannot Add Transitions or Delete Steps in Draft Workflows. 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.

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...

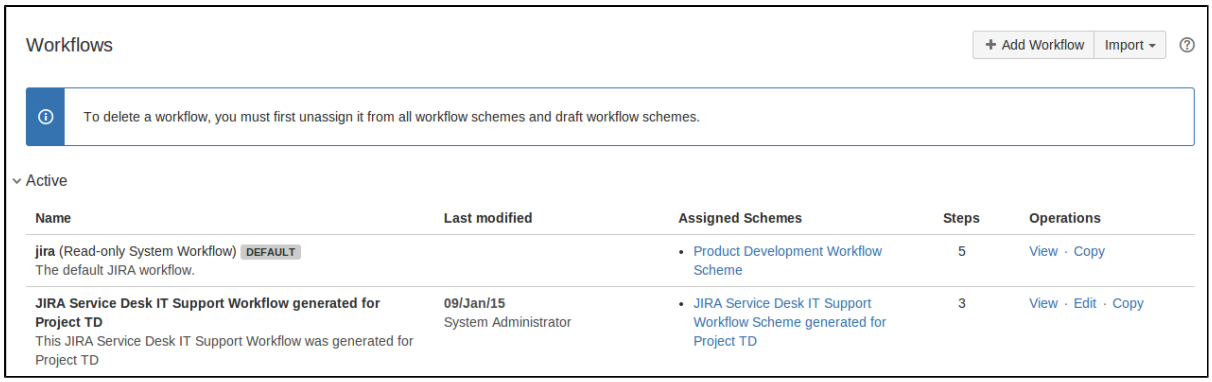
- Statuses are *global objects*. Changing the name of a status on one workflow also changes it in *all workflows that use that status*.
- Hover over a transition or a status to see the relevant transition labels.
- Zoom the diagram with your mouse wheel. Pan the diagram by clicking and holding the mouse 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.

Clone an existing workflow

1. Choose  **> Issues**.
2. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.



3. Copy an existing workflow using the **Copy** link in the Operations column (shown above). Enter a name and description and select the **Copy** button.
4. 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

1. Click **Workflows** in the left-hand nav panel, then **Add Workflow** at the top of the screen.
2. 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**.
3. 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.

1. Choose  **> Projects**.
2. On the Administration page for the project, click **Workflows**.
3. 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.
4. You can now edit your draft workflow. Click on a status or transition to see editing options in the panel that appears.
5. When you are finished, click **Publish Draft**. 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 cancelled*. 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 closes, 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

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 fo r 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**. 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. 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 add-ons.

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.
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:

JRA-37358 - Workflow import creates duplicate custom fields

OPEN

for the request to improve this.

Configuring workflow schemes

On this page:

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.

This page describes how to configure workflows and issue type workflow associations in the scheme.

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	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	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 ins tead. 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	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	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

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.

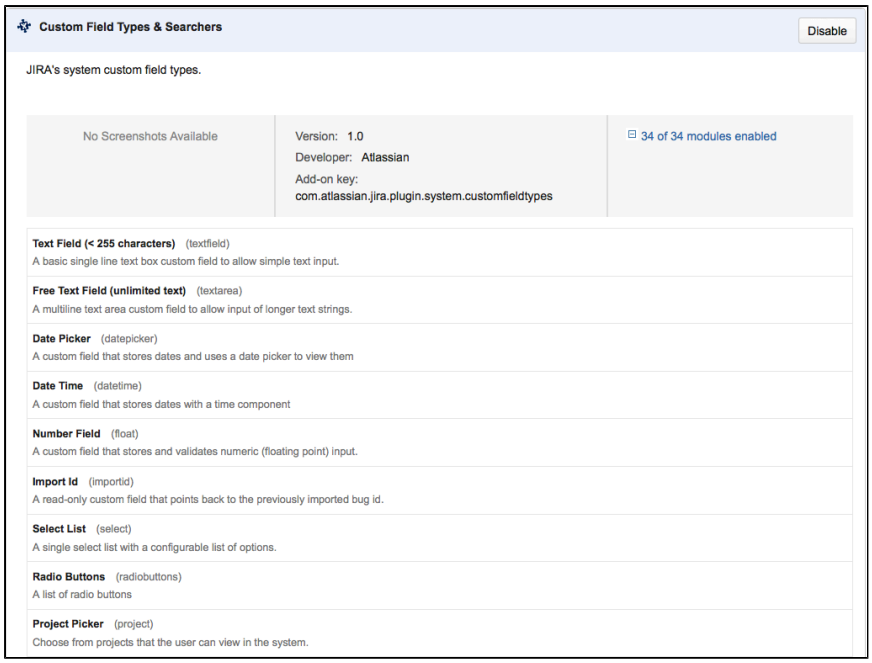
1. Click on the Workflows section in the left-hand panel.
2. Select **Import > Import Workflow**.
3. Select a workflow from your computer to upload, and then click **Next**.
4. JIRA automatically generates a workflow name, but you can change this if you like. Click **Next**.
5. Next, you are presented 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:
   JRA-37358 - Workflow import creates duplicate custom fields
   OPEN
   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 add-ons.

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'.

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 v alidator 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 cllck **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: • Assignee • Description • Environment • Priority • Resolution • Summary • Original Estimate • Remaining Estimate This post function cannot update custom fields and must be positioned after the other optional post functions.

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')

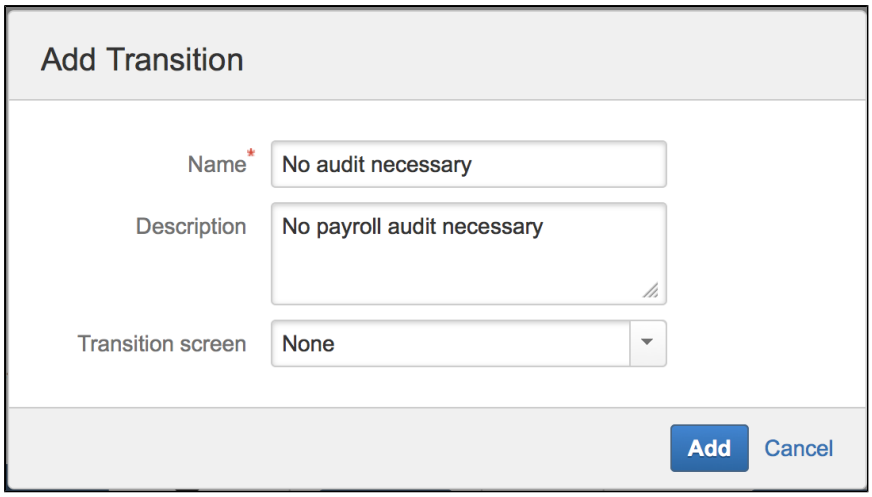
Using a post function to set a field

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.

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.

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:

To access the workflows page:

1. Choose  **> Issues**.
2. Select **Workflows** to open the Workflows page, which displays all of the workflows in your system.

Make sure the Text, rather than Diagram, button is selected, so that your workflow appears in Text edit mode. A list of existing steps that comprise the workflow and each step's Linked Status and Outgoing Transitions (under Transitions (id)), is shown.

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.
   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.

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.
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.
   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 f or 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 Workf low 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* wit hin JIRA is associated with a particular *event* - e.g. the `Issue Created` ev ent is fired when an issue has been created.A listener can execute a specified action once it has been notified that a particular event has been fired. For example, the MailListener can send an `Issue Created` email to a list of recipients defined in the appropriate notification scheme, whenever an issue is created.

Some events are fired by JIRA internally — e.g. an `Issue Updated` or `Issue Moved` event. Other events are fired from within workflow transition post functions — e.g. an `Issue Resolved` event, or a custom event (see below).

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 organisation might need you to modify the default workflow by adding a workflow step called 'QA_Inspection' (e.g. between **Resolve Issue** and **Close Issue**). You would typically also need to generate an email notification to the QA team whenever an issue progresses to the 'QA_Inspection' step of the workflow.

There are three overall steps to achieve this:

1. Add a custom event to the system (e.g. 'Issue Awaiting QA').
2. Configure the notification scheme to send an email when the custom event is fired.
3. Configure the workflow transition post function to fire the custom event.

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** li nk (under Operations).
4. Add the recipients for the custom event as required. See Creating a notification scheme for more information.

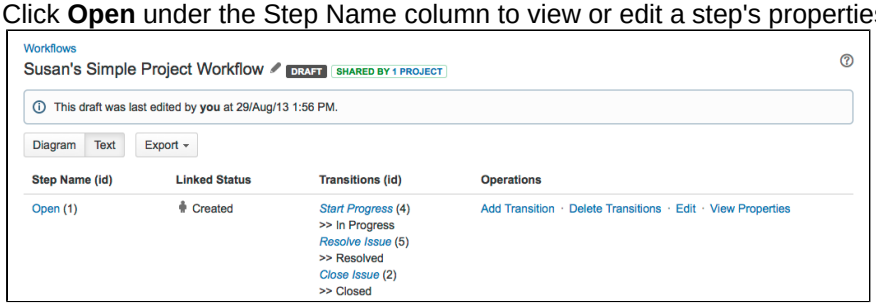
Configuring a post function to fire the custom event

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 Adv anced 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.

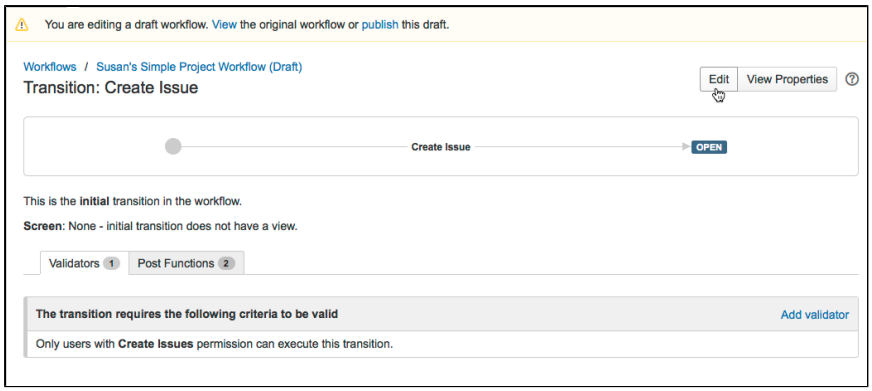
1. Click **Open** under the Step Name column to view or edit a step's properties:



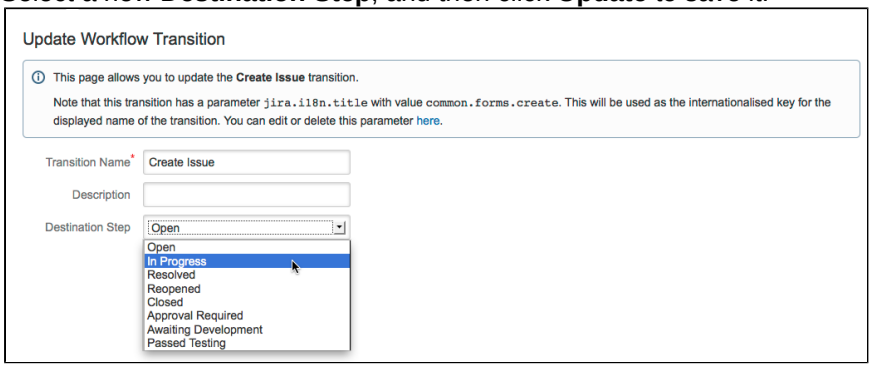
2. Click the **Create Issue** incoming transition:

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:



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

On this page:

While this page on triggers applies to all JIRA applications, triggers are designed to work closely with development tools, and are most powerful when used with JIRA Software.

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.

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. At a minimum, you will need a JIRA Server or JIRA Cloud instance, plus at least one of the following:

- Bitbucket Server (Stash 3.2.0 or later)
- FishEye/Crucible 3.5.2 (or later)
- GitHub Enterprise 11.10.290 (or later)
- Bitbucket
- GitHub

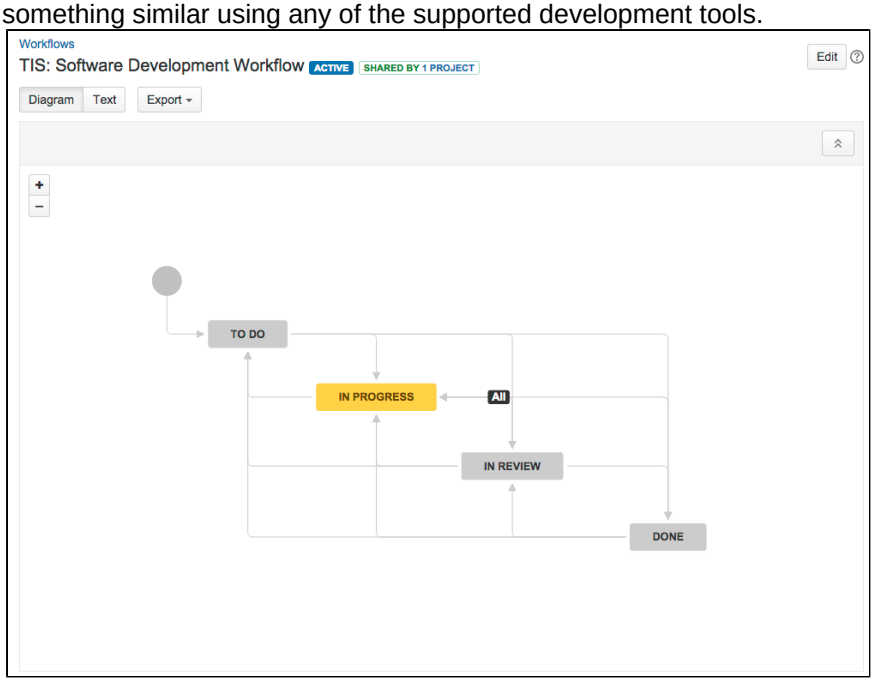
For instructions on how to connect these tools to JIRA, see Integrating with development tools. This page also includes details on other functionality you can enable by connecting the various development tools Atlassian offer.

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

The screenshot and table below show a workflow and triggers similar to what you will be configuring. They re flect the typical interactions between JIRA and development tools in a software development lifecycle. JIRA (6.3.4), 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 (***To Do*** ***In Progress***)	Branch created (Bitbucket Server) Commit created (Bitbucket Server)
Start review (***In Progress*** ***In Review***)	Pull request created (Bitbucket Server) Pull request reopened ((Bitbucket Server) Review started (Crucible)
Restart progress (***In Review*** ***In Progress***)	Pull request declined (Bitbucket Server) Review rejected (Crucible) Review abandoned (Crucible)
Done (***In Review*** ***Done***)	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.

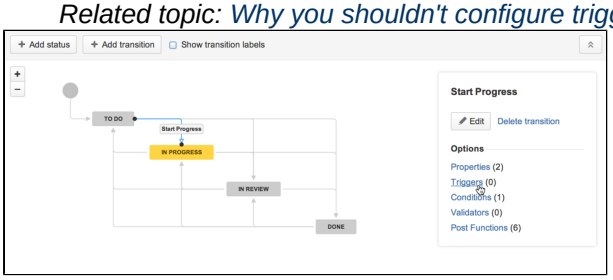
If you already have a similar workflow, navigate to it and edit it: JIRA administration console > **Issues** > **Work flows** > **Edit**

Step 2. Add a trigger to a transition

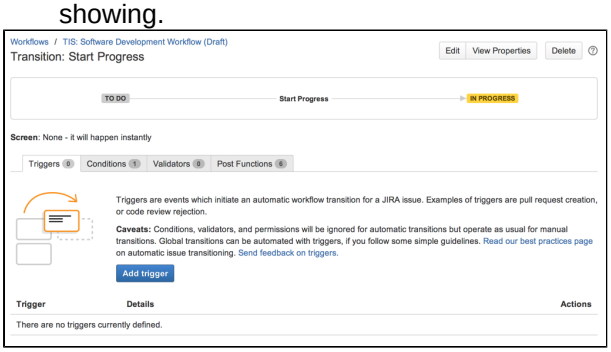
We'll start by adding a 'Commit created' trigger to the 'Start progress' transition. Ensure that you are editing (not viewing) the workflow.

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.

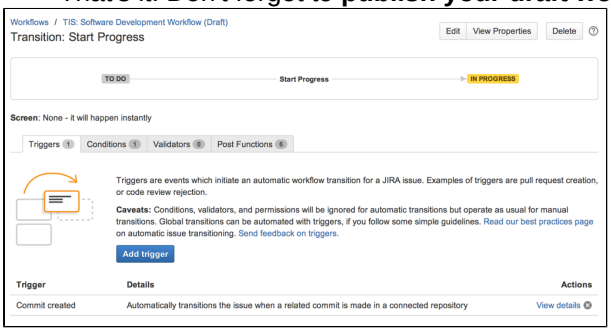


3. Click **Add trigger**, then select **Commit created** in the dialog that appears. A diagnostics window will display — you'll notice that the trigger will be added for all development tools that JIRA is connected to.

   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**.

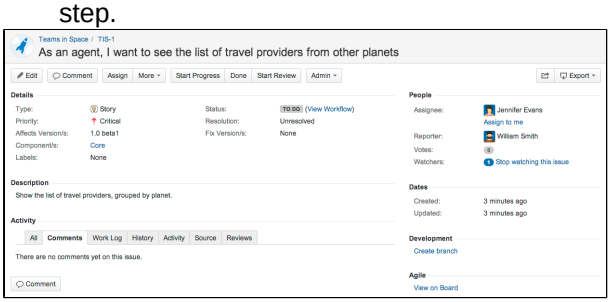
   That's it! Don't forget to **publish your draft workflow**.



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.

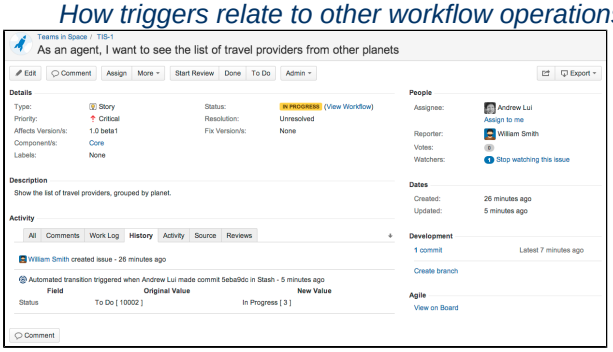
In this example, the issue key is `TIS-1`, which is referenced in the commit message shown in the screenshot.

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	• Pull request created • Pull request merged • Pull request declined (Bitbucket only) • Pull request reopened (Bitbucket Server only) • Commit created • Branch created	• Review started • Submitted for approval • Review rejected • Review abandoned • Review closed • Review summarized	• Commit created • Branch created

There is a known issue where the 'Branch created' event isn't supported for GitHub, which is being tracked under **DCON-432** - Implement 'Create Branch' feature in DVCS connector plugin for Github integration CLOSED — 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 behaviour 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	Include the issue key in the commit message. For example, a commit message like this "TIS-1 Initial commit" will automatically transition the TIS-1 issue from 'To Do' to 'In Progress'.
Create branch	Include the issue key in the branch name, when you create the branch. For example, if you name your branch "TIS-2-feature", it will automatically transition the TIS-2 issue from 'To Do' to 'In Progress'.
Create/Reopen/Decline Merge pull request	Ensure that the pull request includes commits that reference the issue (in their commit messages). 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.

Start/Reject/Abandon/Close review	Include the issue key in the review title, when you create the review. 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.

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. *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***
Your development tools are not connected to JIRA correctly	Check the configuration of your connection: • **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. *All triggers are unique per transition, that is, you can only add a trigger to a transition once.*

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	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. *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.*
Your DVCS repository is not linked to the synchronised DVCS account	Navigate to the JIRA administration console > **Add-ons** > **DVCS Accounts** and enable your repository. *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.*
Your commits are too old	Only commits less than 21 days old will cause a transition. This is to prevent bulk uploads from causing bulk transitions. *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:* `jira.devstatus.commitcreated.age.timeout=P2D` *where P2D is an example ISO-8601 duration representing 2 days.*
The operation is not permitted for anonymous users	Check that each user in your development tools maps to a JIRA user. *Certain issue operations will throw exceptions when the transition is performed by an anonymous user. These are:* • *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* *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.*

The maximum number of automatic transitions permitted for an issue has been exceeded	Check that your workflow transitions do not end in an infinite loop. *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:* `jira.automatic.transitioning.issue.limit`
Automatic issue transition events are incorrectly suppressed by the development tool	Change the repository/project settings to allow events to be sent. *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.* *You shouldn't suppress repository events from Bitbucket Server or FishEye, unless duplicate events are causing issues to transition incorrectly.*

The issue transitions but not as expected:

Possible causes...

Cause	Solution
You have configured a trigger on a global transition	Investigate how the trigger event affects issues in different statuses. Consider removing the trigger from the global transition. *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 behaviour of the issue. See Triggers and global transitions above for more information.*
Workflow conditions, validators and permissions are intentionally ignored for automatic issue transitions	Do nothing. *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.*
Your workflow is shared across multiple projects	You may need to copy your workflow, if you want triggers to apply to the workflow for some projects but not others. *Triggers apply to the workflow. If a workflow is shared across multiple projects, it will include all triggers that have been configured for it.*
Duplicate automatic issue transition events are being sent by multiple development tools	Change the repository/project settings in one (or more) of your development tools to prevent events from being sent. *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).* *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.*

The information recorded for the transition is not correct:

Possible causes...

Cause	Solution
The users in your development tools do not map to users in JIRA	Check that each user in your development tools maps to a JIRA user. *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.*
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.	Do nothing. *This is a known issue that will be fixed in a future release.*

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!**

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
`jira.field.resolution.exclude`	Resolution `id`		

`jira.field.resolution.include`	Resolution `id`	JRA-16443	
`jira.i18n.submit`	i18n property key	JRA-6798	
`jira.i18n.title`	i18n property key	JRA-6798	
`jira.issue.editable`	`true`, `false`		Working with workflows
`jira.permission.*`	user1, user2 / group1, group2 / role/ ...?	JRA-6381 JRA-34621 JRA-35917	• WorkflowBasedPermissionManager clas documentation) • Permissions based on Workflow Status • For link permissions • `jira.permission.edit.group=jir` means that only JIRA administrators car
`opsbar-sequence`	Integer value greater than or equal to 0		Advanced workflow configuration (Customiz

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 other tools	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 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.

Our website highlights some top reasons why people migrate from other issue trackers to JIRA.

On this page:

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 Bugzilla
- Importing data from FogBugz for your Server
- Importing data from FogBugz On Demand
- Importing data from Mantis
- Importing data from Pivotal Tracker
- Importing data from Trac
- Importing data from CSV
- Importing data from Redmine
- Importing data from Bitbucket
- Importing data from Github
- Importing data from JSON
- Importing data from Axosoft
- Importing data from YouTrack
- Importing data from VersionOne
- Importing data from Excel
- Importing data from Rally
- Importing data from TFS or Visual Studio
- Importing data from BaseCamp
- Importing Data from Asana

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 Bugzilla

The JIRA Importers plugin, which is bundled with JIRA, allows you to import data from **Bugzilla** by connecting to a live Bugzilla database.

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 4.0.2. 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.

JIRA is able to import data from Bugzilla 2.20 **only** if it's using a supported database – in this case, MySQL 5.1 or higher.

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

On this page:

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 whilst 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.

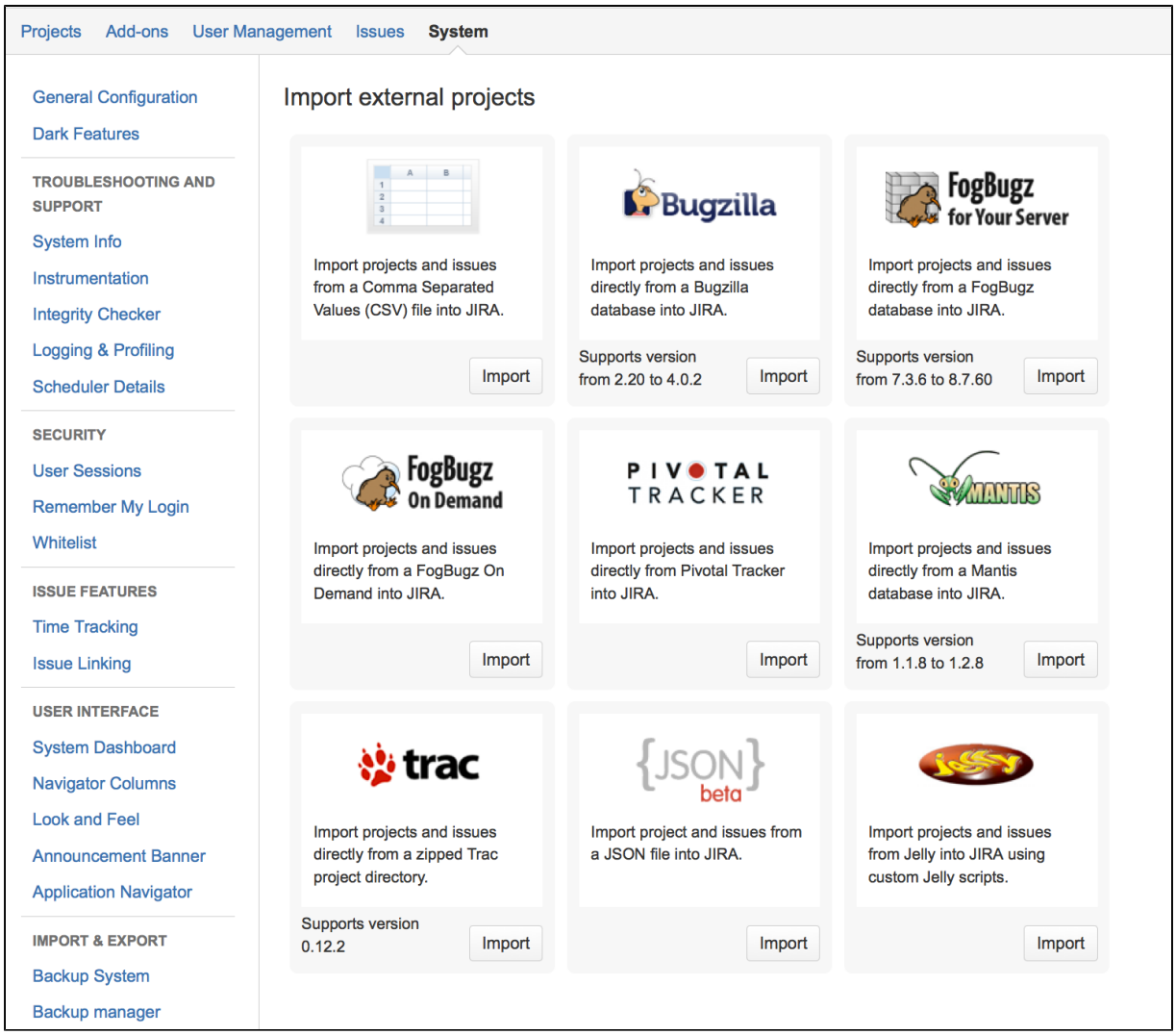
Please note:

- 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.

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 the **Import** button associated with the **Bugzilla** option 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: • 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, `/etc/bugzilla/`
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, `/etc/bugzilla/`
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, `/etc/bugzilla/`
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. **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 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)	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. If you chose MySQL (above), the Bugzilla Import Wizard will add several additional connection parameters by default.

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.
   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**
        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.
   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**.
     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.
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.
  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).
    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.

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. • 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. • 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	You can choose to have the importer automatically create JIRA users for any Bugzilla users who do not already exist in JIRA. • 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 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 8.7.60. 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:

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

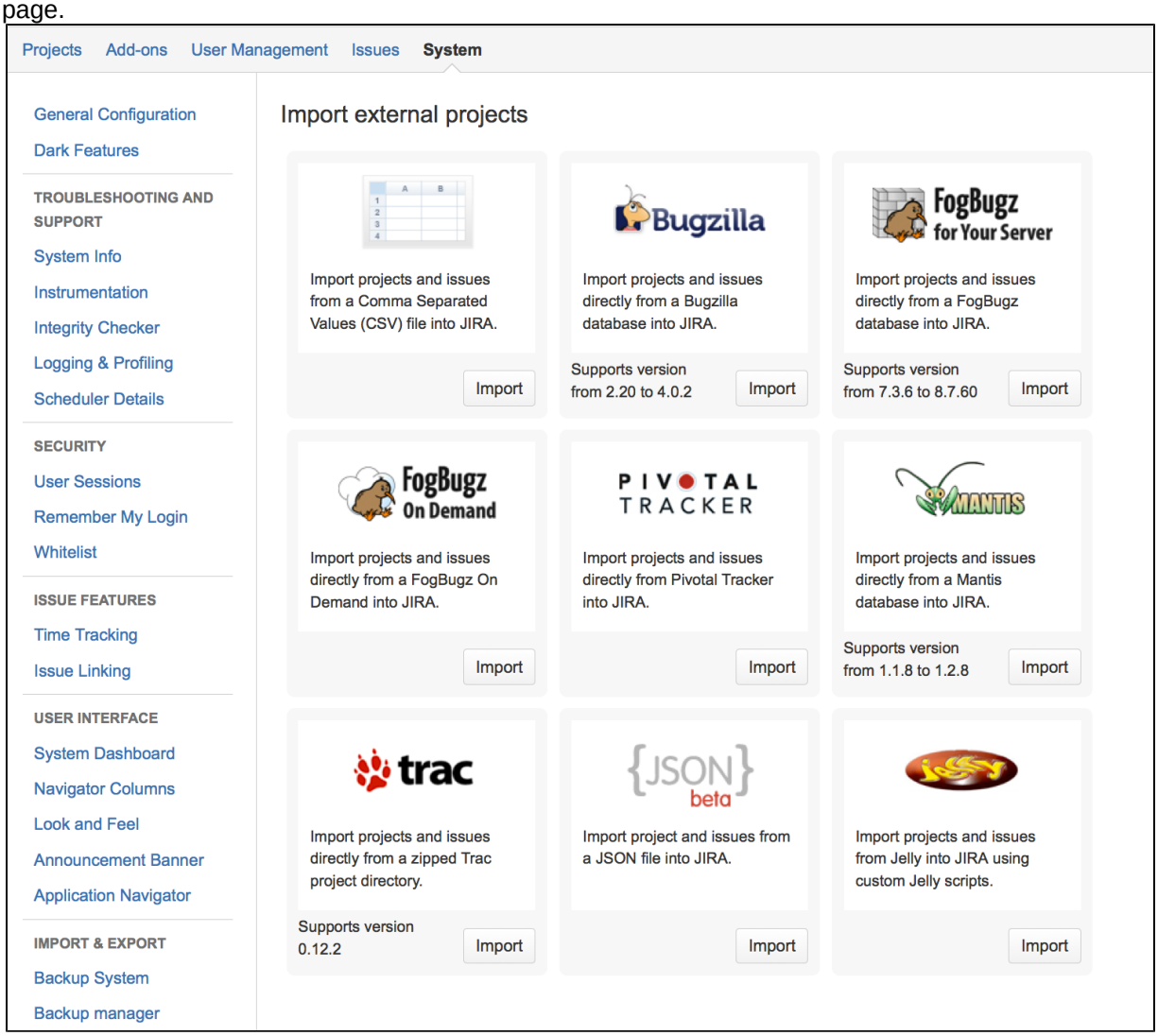
configuration file, which contains the settings you configured whilst running through the FogBugz Import Wizard. This is useful if you need to test your FogBugz import on a test JIRA server first before performing the import on a production system.

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.

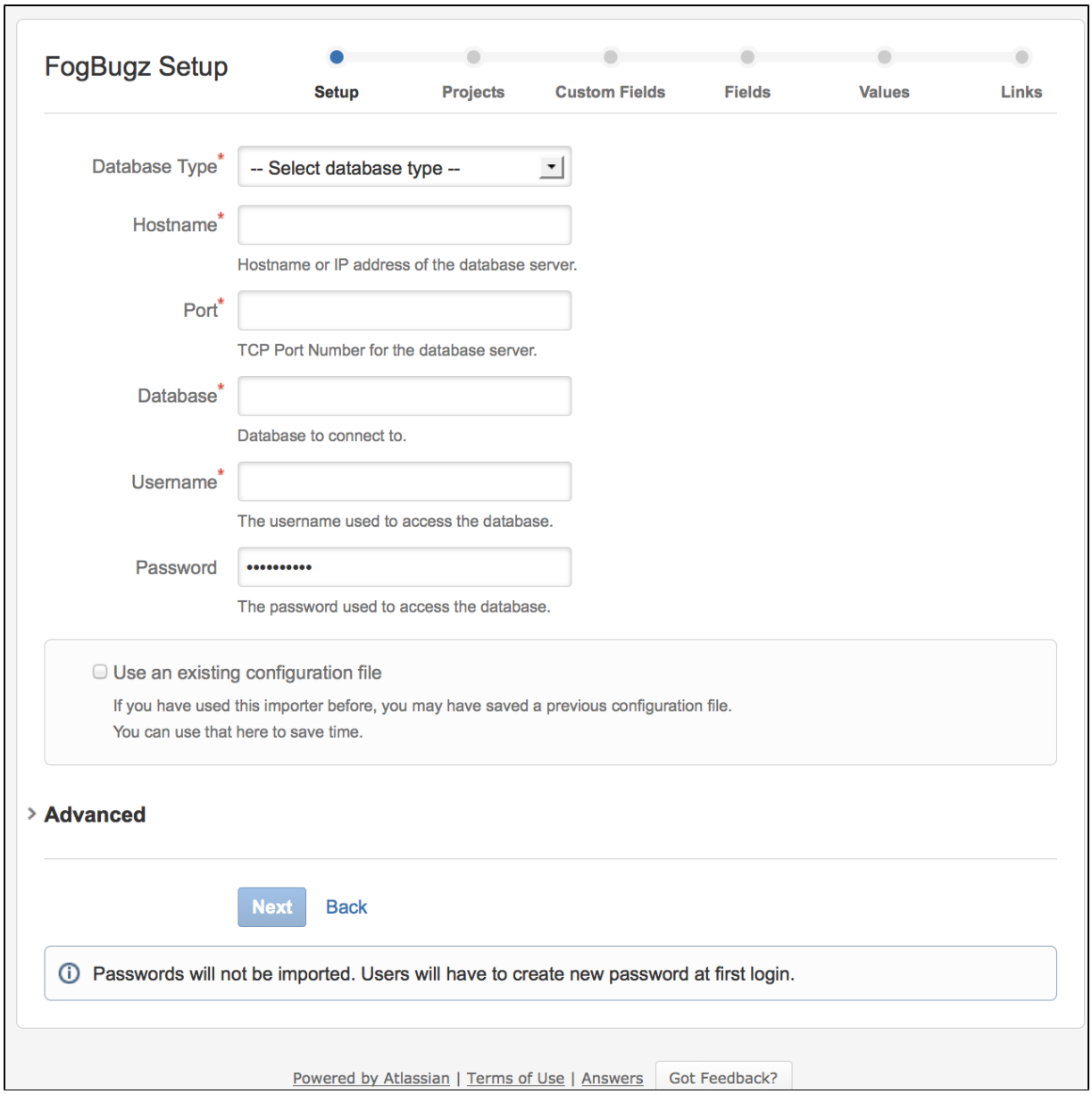
Running the FogBugz for your Server Import Wizard

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 the **Import** button associated with the **FogBugz for Your Server** option to open the **FogBugz Import Wizard: Setup** page.



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: • 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 `fogbugz;instance=sqlexpress` (where `fogbugz` is the name of your FogBugz database and `sqlexpress` 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	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. **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 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)	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. If you chose MySQL (above), the FogBugz 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 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.
  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).
    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.
    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 issueId" or "case issue id". After import, any string matching this pattern will be rewritten to their new JIRA key. For example, a comment "Please see case 100" may be rewritten to "Please see IMP-100".
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. • 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. • 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	You can choose to have the importer automatically create JIRA users for any FogBugz users who do not already exist in JIRA. • 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	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 *FogBugz Custom Field plugin* is not supported.

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.

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:

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.

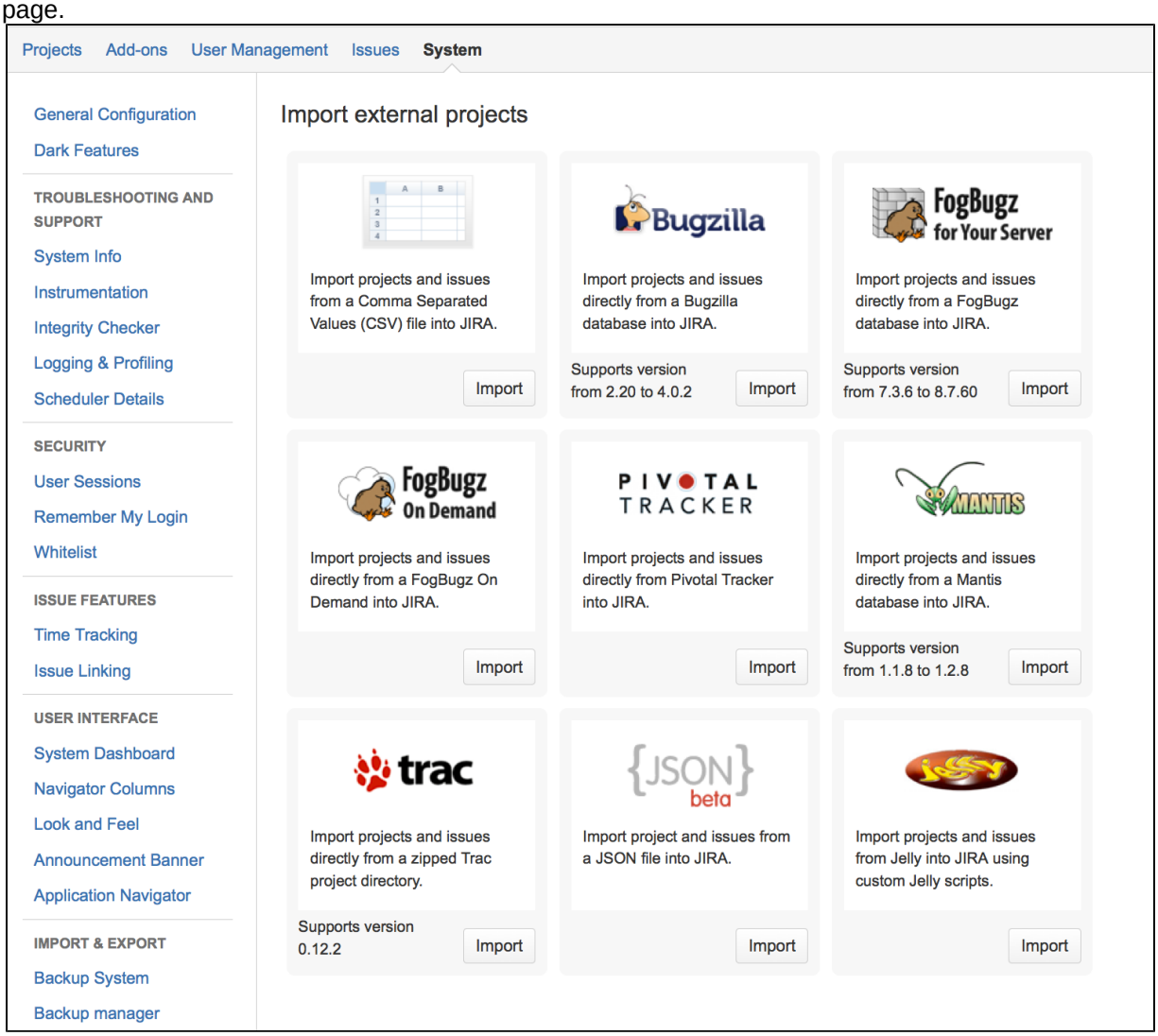
Tip: FogBugz On Demand supports hierarchical issues. During the FogBugz On Demand Import Wizard, you are given the option to recreate this issue hierarchy through JIRA issue links. Hence, before commencing the FogBugz On Demand Import Wizard, you may wish to configure a custom issue link to replicate 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 the **Import** button associated with the FogBugz On Demand option 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 `http://myfogbugzondemand.fogbugz.com`
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.
14. 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 issueId" or "case issue id". After import, any string matching this pattern will be rewritten to their new JIRA key. For example, a comment "Please see case 100" may be rewritten to "Please see IMP-100".
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. • 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	You can choose to have the importer automatically create JIRA users for any FogBugz users who do not already exist in JIRA. • 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	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 *FogBugz Custom Field plugin* is not supported.

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.

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 1.2.8. 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.)

On this page:

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 whilst 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.

Running the Mantis Import Wizard

Before you begin, please back up your JIRA data.

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 the **Import** button associated with the **Mantis** option 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** fiel ds, into which you should specify these user credentials.
Database Type	Select the type of database that your Mantis installation uses: • 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 **Da tabase 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 `config_inc.php`. (Typically, the default username is "root" and the default password is empty). See also http://www.mantisbt.org/manual/manual.confi guration.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:** • 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** ta b)	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.
    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.
      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).
    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.
    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. • 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. • In JIRA, you can configure different types of links (please see Configuring issue linking).
CC List	Watchers	
User	User	You can choose to have the importer automatically create JIRA users for any Mantis users who do not already exist in JIRA. • 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	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.

Importing data from Pivotal Tracker

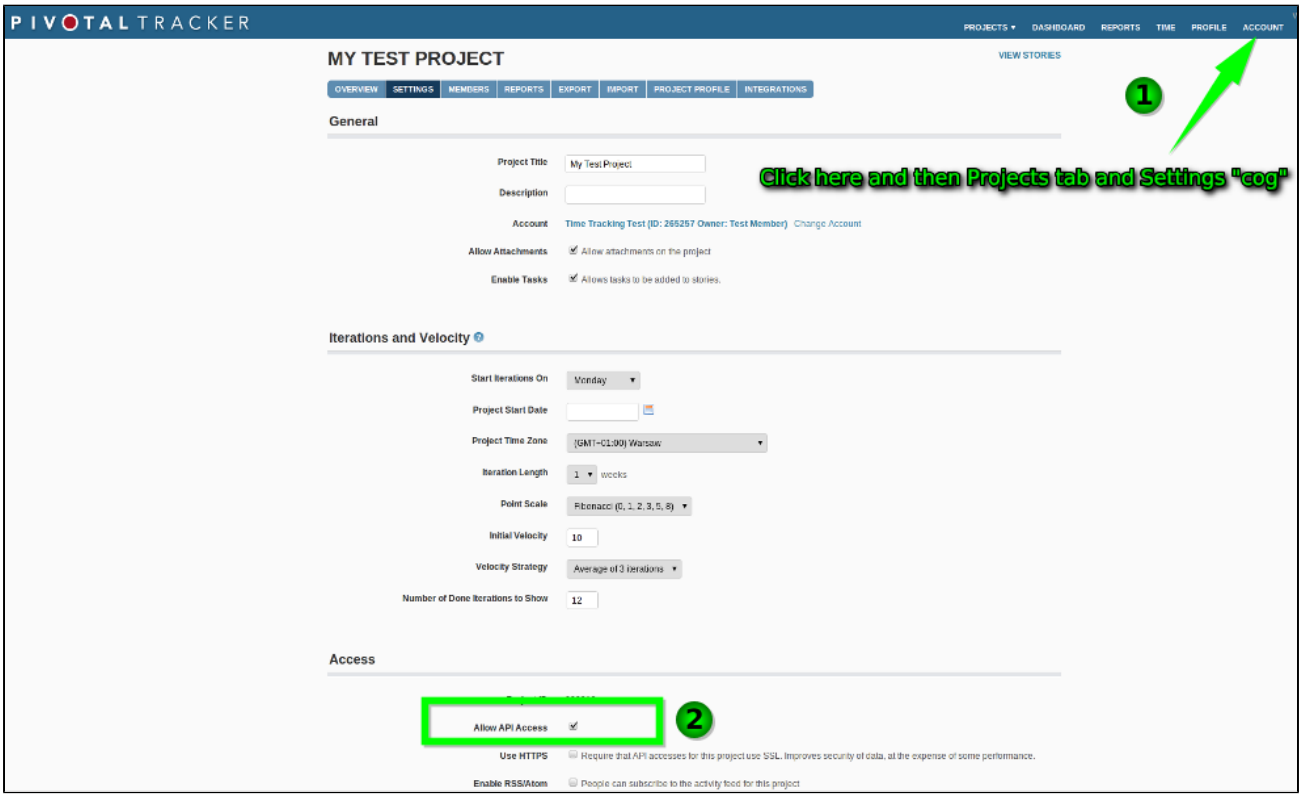
The JIRA Importers plugin, which is bundled with JIRA, allows you to import data from **Pivotal Tracker**, a 'Software as a Service' (SaaS) issue tracker application.

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

In Pivotal Tracker, please ensure you have switched on **Allow API Access** in your Pivotal Project's Settings.



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 the **Import** button associated with the Pivotal Tracker option 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)	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. **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 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).

5. Click the **Next** button to proceed to the **Setup project mappings** step of the Pivotal Tracker Import Wizard.
6. On the **Setup project mappings** page, select which Pivotal Tracker projects you wish to import into JIRA.
   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. 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**.
   - 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.
     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:
     a. 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**.
     b. *(Optional)* Delete this project if you do not intend to use it any further.
     c. 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.)
     d. Re-use the Pivotal Tracker Import Wizard to import your issues into this existing JIRA project.
7. Click the **Next** button to proceed to the **Setup user mappings** step of the Pivotal Tracker Import Wizard.
   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	Story ID and Story 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	The importer will automatically create JIRA users for any Pivotal Tracker users who do not exist in JIRA. • 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 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 version 0.12.2.

On this page:

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 > Import** button associated with the Trac option to open the **Trac Import Wizard: Setup** page.
3. On the **Trac Import Wizard: Setup** page, select your compressed Trac environment file, which you prepared above.
4. 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).

5. Click the **Next** button to proceed to the **Setup project mappings** step of the Trac Import Wizard.
6. 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**.
7. 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.
8. **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.
9. 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).
10. **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.
11. 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	The importer will automatically create JIRA users for any Trac users who do not exist in JIRA. • 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	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.

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).

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 whilst 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.

On this page:

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.

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.

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"
```

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.
  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 (JRA-34202 - Allow CSV import to support Multi-Level Cascading Select plugin fields OPEN).

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 > Import** button associated with the Comma-separated values (CSV) option to open the **CSV File import** page.
3. 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).
   ℹ **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.
4. 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.
   ℹ **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).
5. Click the **Next** button to proceed to the **Setup project mappings** step of the CSV file import wizard.
6. 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	Choose either of the following:
	• **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**.
	• **Defined in CSV**. Ensure that every issue in your CSV file includes data for the JIRA **Project Name** and **Project Key**.
	✅ 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.

7. Click the **Next** button to proceed to the **Setup field mappings** step of the CSV file import wizard.
8. 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.
9. 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.
10. 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.
11. Click the **Next** button to proceed to proceed to the **Setup value mappings** step of the CSV file import wizard.
12. 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.
      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.
13. 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.
    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	You can import issues with multiple labels by entering each label in a separate column.
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. • 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
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.doImpor
t(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
```

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 exists 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 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 2.0+.

Before you begin

- Ensure that you are using Redmine versions 1.3.0+ and 2.0+.
- 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 add-ons.
- Install the JIRA Redmine Importer plugin, if you haven't installed it already. For instructions on how to install a plugin, see Installing add-ons.
- Enable the REST web service in Redmine in **Administration > Settings > Authentication > Enable REST web service**, if you haven't already enabled it.
  (click for larger image)

- 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 the **Import** button associated with the Redmine option.
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	You can configure the mapping of specific Redmine values to specific JIRA values, provided you create your workflows in JIRA before running the importer. • 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	You can choose to have the importer automatically create JIRA users for any Redmine users who do not already exist in JIRA. • Users who interacted with the Redmine system will be created as active accounts in JIRA. Other users will be imported into a special group called "redmine-import-unused-users" and will be deactivated. • Passwords from Redmine are not imported. Users from Redmine 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.
Other fields	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 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 add-ons.
- 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 add-ons.
- 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 Import.**
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	You can configure the mapping of specific Bitbucket values to specific JIRA values, provided you create your workflows in JIRA before running the importer. • 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	You can choose to have the importer automatically create JIRA users for any Bitbucket users who do not already exist in JIRA. • 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.

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

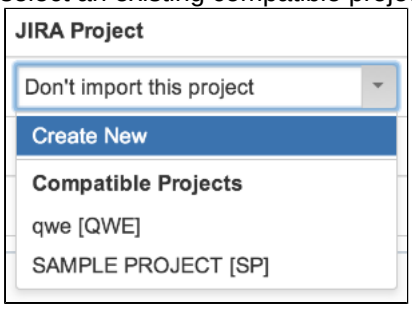
personal, public and starred repositories, the GitHub importer will display all repositories it can access and you can pick and choose which ones you want to import. If your GitHub instance has 2 factor authentication, you will be required to either provide the 6 digit access code that you will be sent, or a back-up code.

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.

Running the GitHub 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. Choose  **> System**. Select **Import & Export > External System Import** to open the Import external projects page.
3. Select the **Import** button associated with the **GitHub** option to open the **GitHub Import Wizard**.
4. On the **GitHub Setup** page, select which type of GitHub you are using. If you are using GitHub Enterprise you will also be required to provide your GitHub Enterprise URL. You also need to provide either your GitHub username and password, or a GitHub Personal Access Token. Note if you have used the GitHub import wizard before and saved a previous configuration file, you can select the configuration file here to speed up your import.
5. Click **Next**.
   Have 2 factor authentication? Click here..
   If you have 2 factor authentication on your GitHub account, you will be prompted to enter your 6 digit code now, and then click **Next**.

   The Authentication page displays, verifying your authentication has been successful.
6. Click **Next**. The **Map projects** page displays, and will show a list of all your public and private repositories, as well as any repositories you have starred.
7. On the **Map projects** page, select the repositories you want to import data from, and where you want to import it to.
   All GitHub projects are initially set to "Don't import this project". To import a repository, 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**
      This will be used as the prefix for all issue IDs in your JIRA project.
   c. The **Project Lead**.
8. Click **Next**. The **Fetching Data** page will display, updating you on the progress of your import.
9. The **Labels** page will display when your import has completed. As GitHub only uses labels, you can now map the labels to an issue type and/or a resolution. You do not have to map every label, and if you would like to create JIRA labels to correspond to the GitHub labels you have *not* mapped, ensure the "Add JIRA labels..." tick box is checked.
10. Click **Next**. The **Workflow status mapping** page displays, and allows you to select which workflow you wish to apply to your imported GitHub repository. You must also select what to map your GitHub open and closed issues to. The default is open to open and closed to closed, but you can select from the issue states available in your JIRA workflow.
11. Click **Begin import**.
12. Success! You have completed importing your GitHub 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.

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 GitHub Import Wizard process.
- If you need to import data from another GitHub repository with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a GitHub 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 JSON

On this page:

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.

JSON File Example Expand source

```
{
    "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",
```

```
            "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 Expand source

```
 "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	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.	**User Example** Expand source `"users": [ { "name" : "someuser", "groups" : [ "jira-users", "my-custom-group" ], "active" : true, "email" : "user1@example.com", "fullname" : "User 1" } ]`
Project Key and Issue Key	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".	**Project Key and Issue Key Example** Expand source `{ "projects": [ { "name": "Sample data", "key": "SAM", "issues": [ { "key" : "SAM-123", "status" : "Open", "reporter" : "admin", "summary" : "Parent case", "externalId": "123" } ] } ] }`

Comments	This example shows how you can import multiple comments for an issue.	**Comment Example** Expand source

```
{
    "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"
                        }
                    ]
                }
            ]
        }
    ]
}
```

Worklogs	This example shows the syntax to import worklog detail.	**Worklog Example** Expand source `"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" } ]`
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 JIR A Importers plugin will always create a new component with "Default Assignee" switched to "Project Default", as you are unable to specify a "Default Assignee".	**Component Example** Expand source `"components": [ "Component", //Component specified only by name { // Component specified by object "name": "SomeName", "lead": "admin", "description": "Some description" } ],`

Issues with Time Tracking	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. Please ensure Time Tracking is enabled in JIRA before you start your import, otherwise the data will be ignored by the JIR A Importers plugin during the import.	**Issues with Time Tracking** Expand source `"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" }] }]`

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 the **Import from JSON** button associated with the JSON option 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 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 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 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 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 chose 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 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 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 imorter. 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 Spartez
- 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 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 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.

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 the **Import** button associated with the **Asana** option 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.
   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**
      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.

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.

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

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

You can get a copy of JIRA Cloud data by using the Backup Manager. This is useful if you want to import cloud data into JIRA Server installations or into other sites. You can also use this approach as a way of taking occasional offline backups.

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).

On this page:

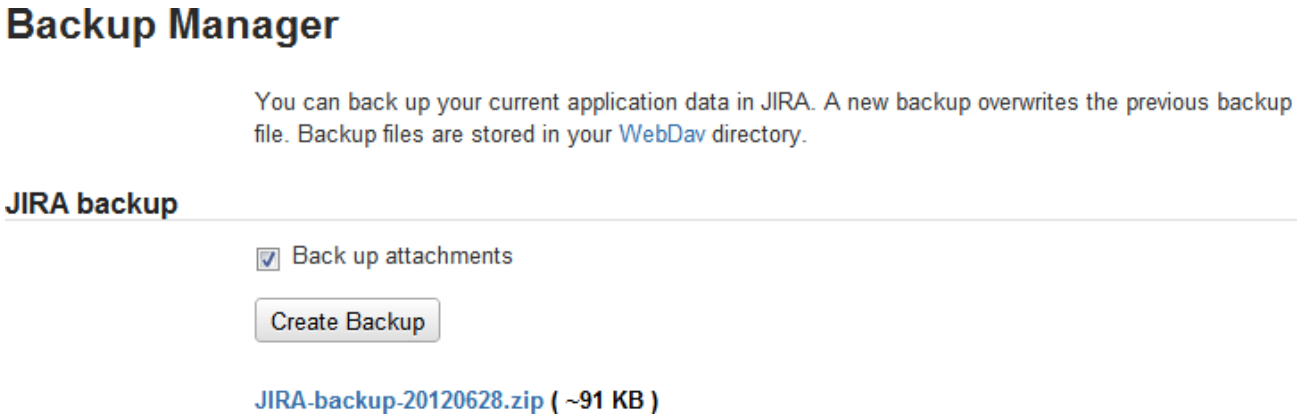
JIRA Backup Manager

Only one backup file is stored at a time and an existing backup is overwritten by new ones.

You can only generate a new backup every 48H, from the time the backup completes.

1. Choose  **> System**.
2. In the **Import & Export** section, select **Backup manager**.
3. If you want to back up attachments, select the option for it.
4. Click **Create Backup** to start the backup.
   After the backup is complete, you will see the link to the backup file. You can download the file by following the link.

Screenshot: Backup Manager in JIRA Cloud



JIRA Backup Manager Known issues

On a subset of instances that have JIRA Cloud only, the **JIRA Backup Manager** menu cannot be found on the user interface.
Workaround: Access the two menus with one of the following URLs.
`https://<domain_name>.atlassian.net /plugins/servlet/ondemandbackupmanager/admin`
`https://<account_name>.jira.com/plugins/servlet/ondemandbackupmanager/admin`

Retrieving the backup data

The backup files are stored in your WebDAV directory: `https://<domain_name>.atlassian.net/webdav/backupmanager`.

Screenshot: The backup file in the WebDAV directory

What data is backed up

The backup includes the following data:

- Issues
- Users and their group settings
- Avatars
- Attachments if selected

Screenshot: Inside the backup

Importing the backup data into JIRA Cloud

> See below for how to structure your export file prior to importing into JIRA

To import the backup data to JIRA Cloud:

- To import the backup into JIRA Cloud, follow the instructions on importing issues.

To log in for the first time after the import:

- Use the following credentials to log in:
  - The administrator credentials you use to log in to your source JIRA site.

Importing the backup data into JIRA

> See below for how to structure your export file prior to importing into JIRA.

You can import the backup data into local JIRA systems.

To do this:

- Refer to the instructions on the Migrating from JIRA Cloud to JIRA Server page.

To log in for the first time after the import:

- Use the following credentials to log in:
  - user: sysadmin
  - password: sysadmin

Make sure that you change the password after you log in.

Note, the 'sysadmin' user is created automatically during the backup process. The reason for the creation of this user is that you must log in as a user with the 'JIRA System Administrator' global permission after restoring data in JIRA, and this permission is not available to customers in JIRA Cloud.

Notes

Structure of JIRA Cloud export for importing into JIRA

You will need to extract the data directory from the JIRA Cloud export and ensure the structure of the archives are as follows:

JIRA XML zip:

- entities.xml
- activeobjects.xml

JIRA attachments zip:

- Project directories (PROJ, JRA, etc)

JIRA avatars zip:

- image files

JIRA logos zip:

- image files

Cross-application links point back to your site

Cross-application links, i.e. a link to a source file page from a JIRA issue, point back to the corresponding source cloud location after the import.

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

> This import procedure overwrites all the existing data and configuration in JIRA cloud applications and cross-application settings.

Before you begin

- ⚠ 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 **JIRA Import** in the **System** section.
   The JIRA Import Wizard appears. Read the prerequisites to view the supported formats for the data to be imported.
3. In your JIRA 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.
4. *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

1. Upload the files to the cloud by using WebDAV. (For backup files larger than 4GB, refer to Uploading Large files to WebDAV)
2. Log in as an administrator. Then at the top right of the screen, choose  **> System**.
3. Then choose **JIRA Import** in the **System** section.

4. 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 '**JI RA 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.

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

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 JIRA's 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.

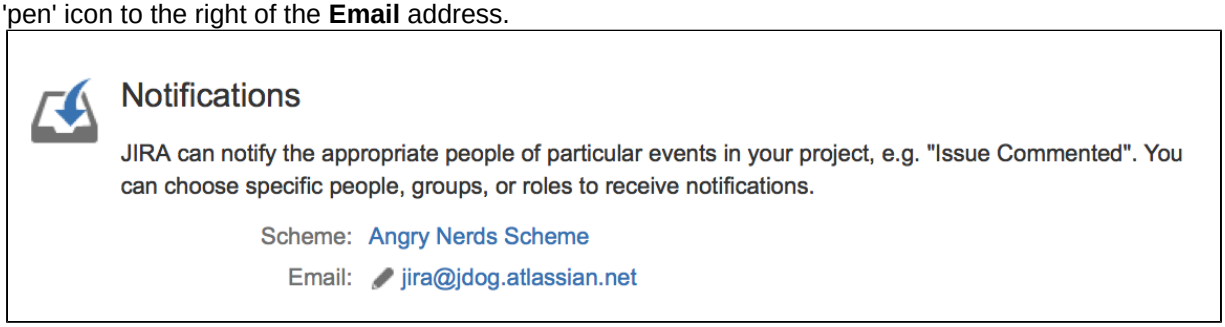
Configuring a project's email address

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** s pecified 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**. 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.



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** configuratio n) 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 JIRA's 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.

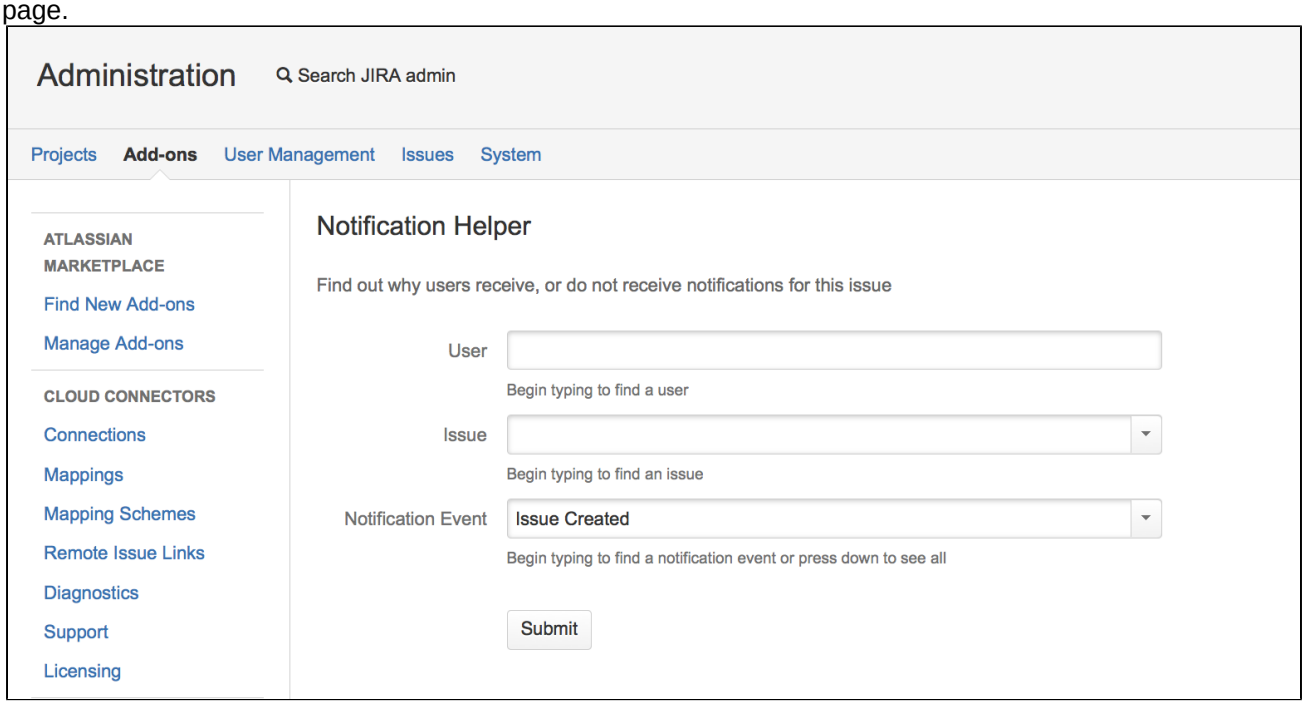
To diagnose why a user is or is not receiving notifications for an issue:

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.



Configuring JIRA's 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 Administrators** global permission.

On this page:

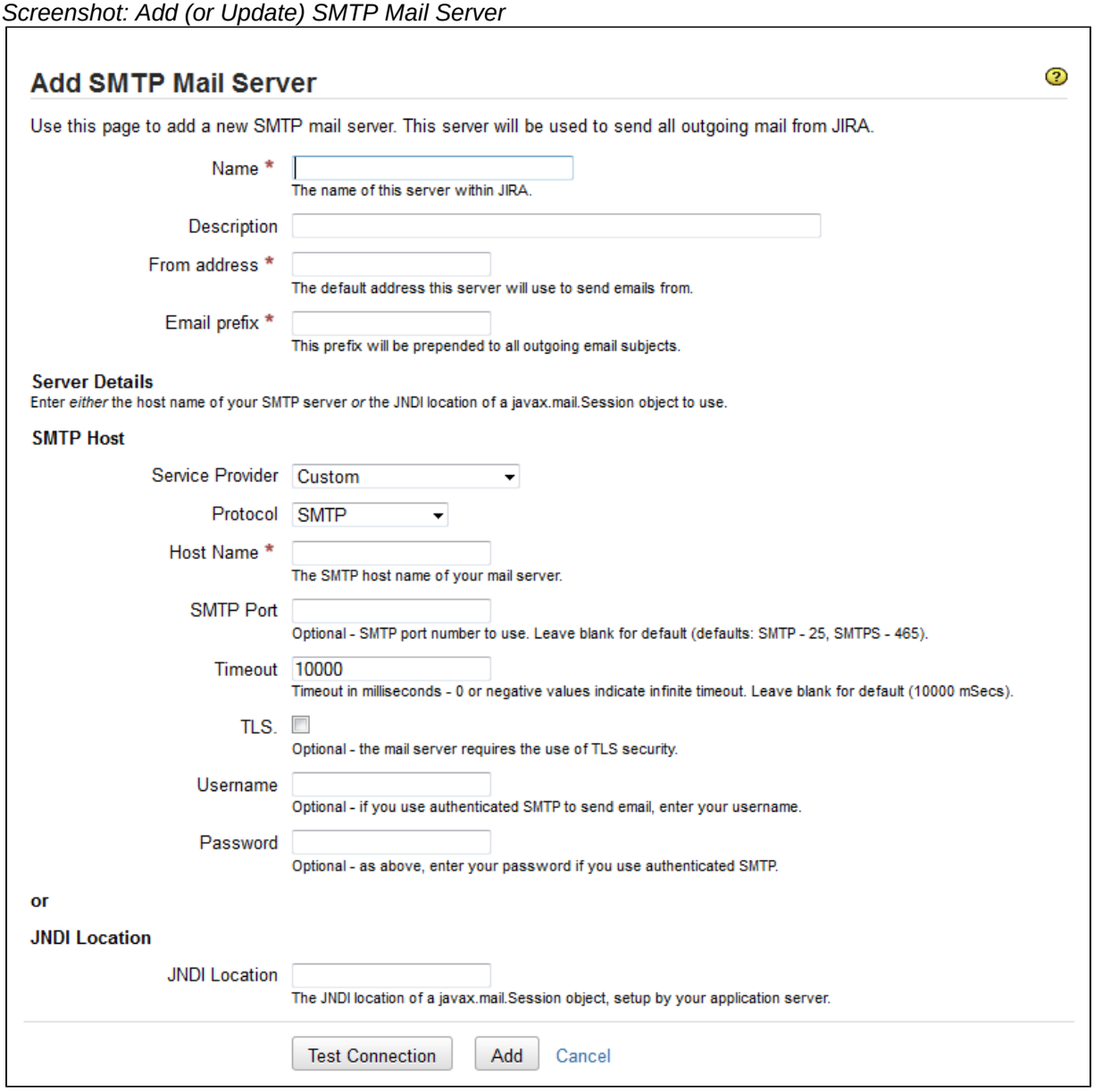
Define or edit the SMTP mail server

1. Choose

> System.
2. Select **Mail > Outgoing Mail** to open the SMTP Mail Server page.
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. `jira@example-company.com`). 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



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.
  - **Centralised 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. `smtp.yourcompany.com`
SMTP Port	*(Optional)* The SMTP port number, usually 25 for SMTP or 465 for SMTPS, either of which are assumed if this field is left blank.
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 SMTP server to respond.
TLS	*(Optional)* Select this checkbox if your SMTP host uses the Transport Layer Security (TLS) protocol.
Username	*(Optional)* 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	*(Optional)* Again, if your SMTP host requires authentication, spcify 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 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

JIRA can be configured to automatically create issues or comments on existing issues based on incoming messages received by a mail server or external mail service.

This is especially useful in a helpdesk or support scenario, where users send support queries via email that you wish to track with JIRA. Subsequent email messages about the issue (for example, responses to email notifications) can be automatically recorded as comments. Additionally, any attachments in the emails can automatically be attached to the issue (when enabled).

> If you're looking to set up and manage incoming emails to a service desk project, note that JIRA Service Desk uses a different, built-in email processor from the one described here for JIRA Software and JIRA Core issues. To learn more about receiving service desk requests sent by email, check out this page in the documentation.

On this page:

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 JIR A Home Directory's `import/mail` subdirectory, which is scanned periodically.

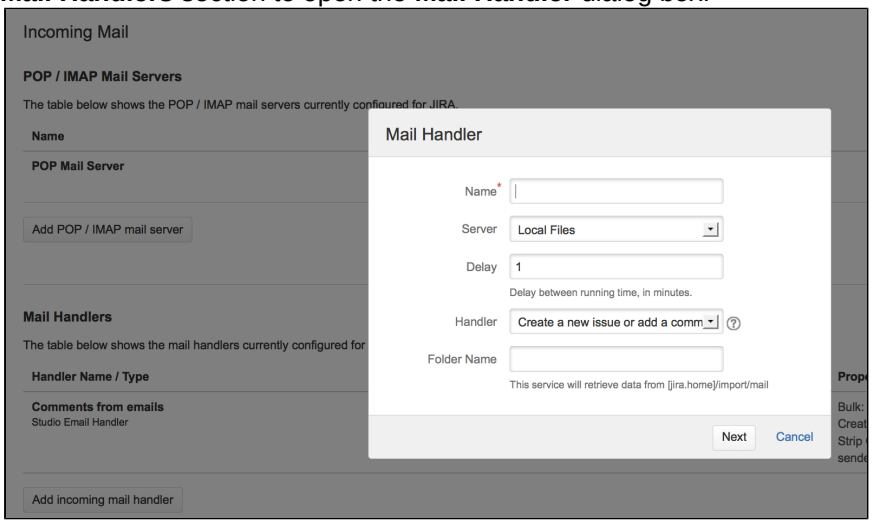
Please also be aware that JIRA expects only one message per file, so your external mail service should be configured to generate such output.
Note — how JIRA handles messages on a mail server/service:

- 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 IMAP mail server) or file system (i.e. for file system messages).
- 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.



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:** • 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.
Catch Email Address	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` 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). **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. 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 **JRA-41831** - Duplicate issues creation fails - Creating multiple issues by one Email RESOLVED .
Bulk	This option only affects 'bulk' email messages whose header has either its **Precedence:** field set to **bulk** or its **Auto-Submitted** field 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: 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. 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).
Forward Email	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.

Create Users	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. 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. 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** fiel d option below and as such, choosing the **Create Users** option will hide the **Default Reporter** option.
Default Reporter	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 `emailed-r eporter` **Note:** • 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:** fi eld address is appended in a brief message at the end of the issue's **Description** fi eld, so that the sender can be identified.
Notify Users	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. **Note:** this option only functions if the **Create Users** checkbox has been selected.
CC Assignee	Select this checkbox if you want JIRA to automatically assign the issue created to a JIRA user: • 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** (spe cified above).
CC Watchers	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. Please note that when an issue is created, new JIRA users created by the **Create Users** option (above) *cannot also be added* to the issue's watchers list by this **CC Watchers** option. JIRA users must *already* exist in JIRA's userbase, and must have an email address.

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	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` 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). **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.
Bulk	This option only affects 'bulk' email messages whose header has either its **Precedence:** field set to **bulk** or its **Auto-Submitted** field 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: 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.
Forward Email	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.
Create Users	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. 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. 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.

Default Reporter	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 `emailed-reporter` **Note:** • 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	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. **Note:** this option only functions if the **Create Users** checkbox has been selected.

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	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` 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). **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.
Bulk	This option only affects 'bulk' email messages whose header has either its **Precedence:** field set to **bulk** or its **Auto-Submitted** field 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: 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.
Forward Email	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.

Create Users	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. 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. 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** fiel d option below and as such, choosing the **Create Users** option will hide the **Default Reporter** option.
Default Reporter	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 `emailed-repo rter` **Note:** • 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	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. **Note:** this option only functions if the **Create Users** checkbox has been selected.

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	Specify the project key of the default project to which new issues are created by this handler — for example, `JRA`. **Note:** • 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.

Catch Email Address	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` 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). **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.
Bulk	This option only affects 'bulk' email messages whose header has either its **Precedence:** field set to **bulk** or its **Auto-Submitted** field 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: 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.
Forward Email	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.
Create Users	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. 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. 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.
Default Reporter	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 `emailed-reporter` **Note:** • 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	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. **Note:** this option only functions if the **Create Users** checkbox has been selected.

CC Assignee	Select this checkbox if you want JIRA to automatically assign the issue created to a JIRA user: • 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	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. Please note that when an issue is created, new JIRA users created by the **Create Users** option (above) *cannot also be added* to the issue's watchers list by this **CC Watchers** option. JIRA users must *already* exist in JIRA's userbase, and must have an email address.

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	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.
	
	Please Note:
	
	• 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.
	
	For example:
	
	`/----\s*Original Message\s*----/`
	
	or
	
	`/_____*/`

Catch Email Address	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` 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). **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.
Bulk	This option only affects 'bulk' email messages whose header has either its **Precedence:** field set to **bulk** or its **Auto-Submitted** field 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: 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.
Forward Email	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.
Create Users	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. 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. 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** fiel d option below and as such, choosing the **Create Users** option will hide the **Default Reporter** option.

Default Reporter	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 `emailed-reporter` **Note:** • 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	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. **Note:** this option only functions if the **Create Users** check box has been selected.

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 'JRA' 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 Disable 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 Outlooks' '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

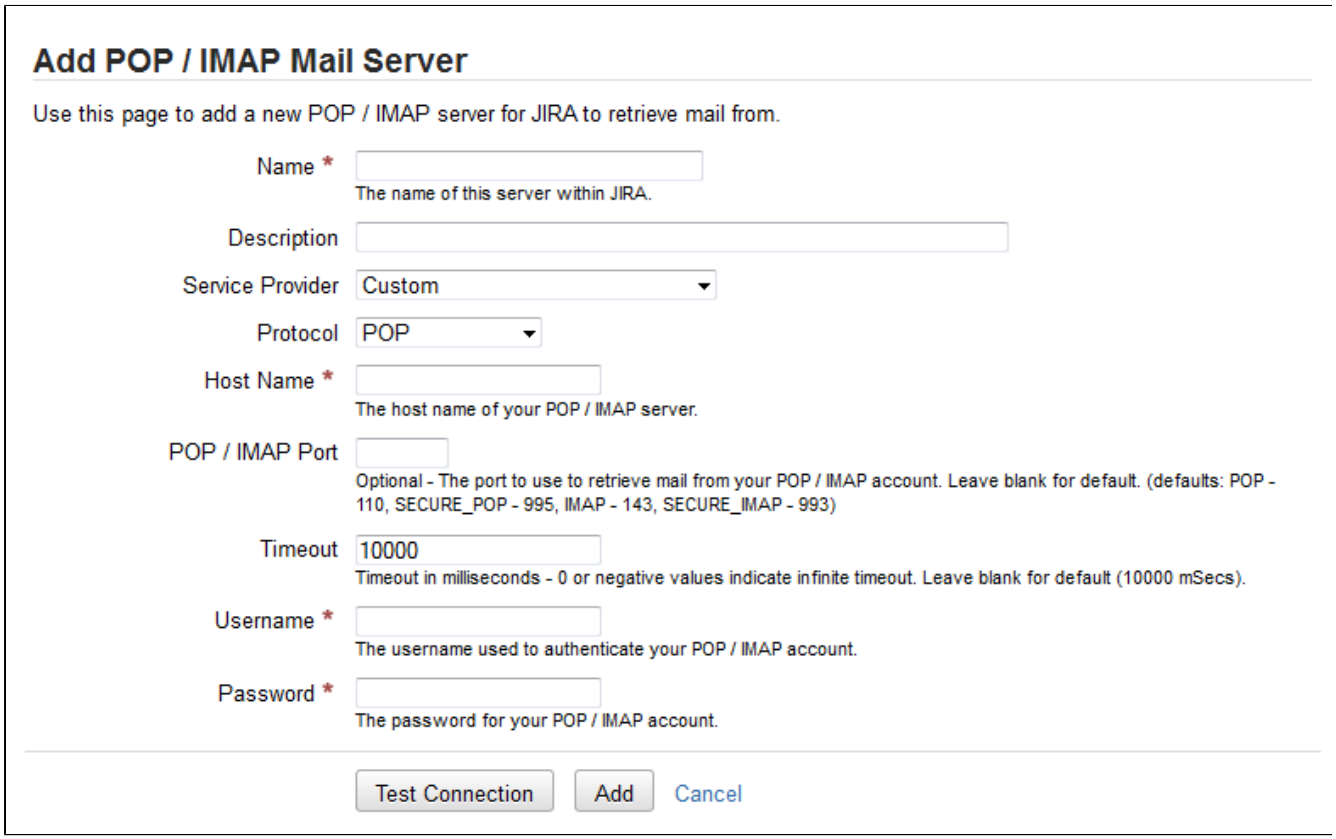
1. Choose  **> System**.
2. Select **Mail > Incoming Mail** to open the Incoming Mail page.
3. Click either the **Configure new POP / IMAP mail server** button to define a new POP / IMAP mail server, or the **Edit** link at the right of an existing POP / IMAP mail server configuration, which will open 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. 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. `pop.yourcompany.com` or `imap.yourcompany.com`
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



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':

Server 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 optimise 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
- 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

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.
3. 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** scr een 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.

On this page:

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.

Step 1: Diagnosis

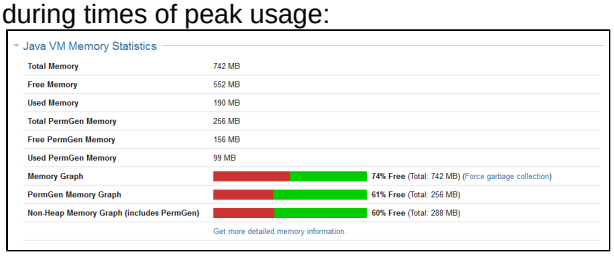
Expand to see diagnosis section

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 **Troubleshooting and 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:

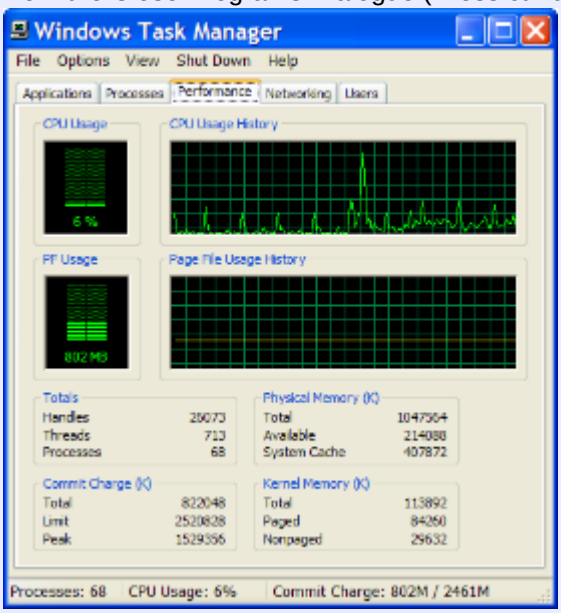


This server has been allocated a maximum of 768 MB and a minimum of 256 MB (typically defined in the `setenv` script which is executed by running the `start-jira` script). If you are trying to see whether your settings are being picked up by JIRA applications, this is where to look. Here, you can see 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 Dialogue (Press ctrl-alt-delete), select the Performance tab:



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

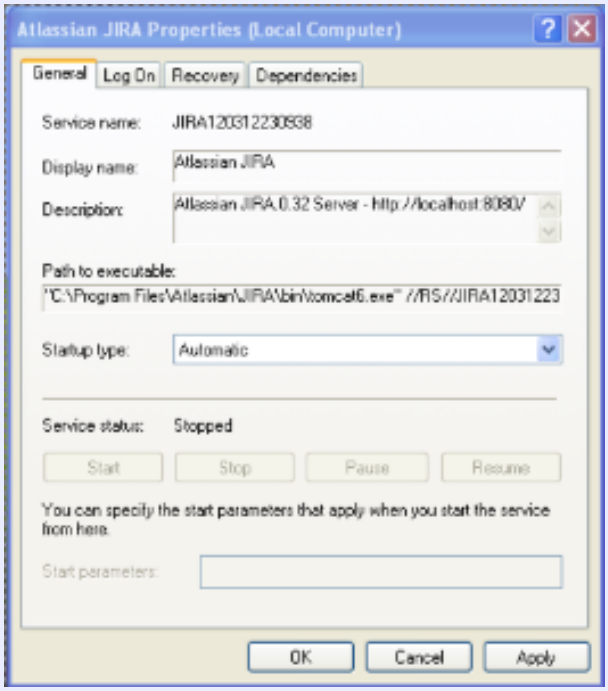
Expand to see Windows service instructions

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):



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
```

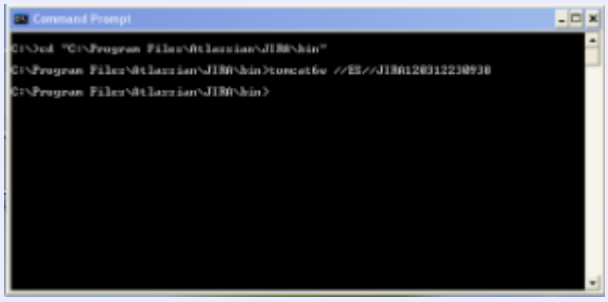
4. For JIRA 5.1 or below:

```
tomcat6w //ES//%SERVICENAME%
```

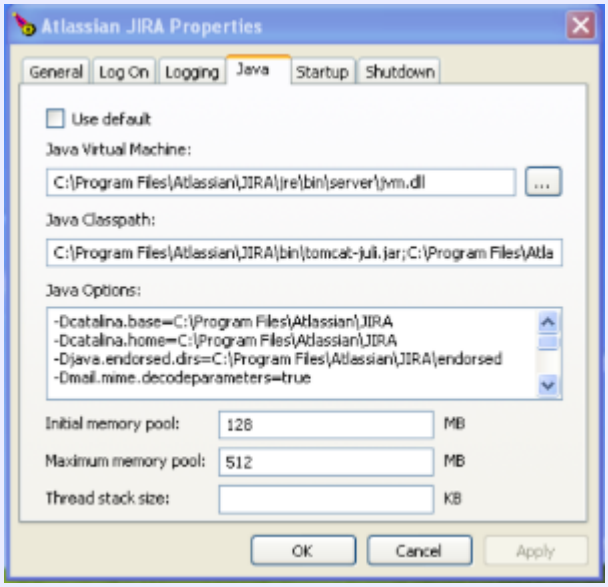
For JIRA 5.2 or above:

```
tomcat7w //ES//%SERVICENAME%
```

In the above example, it would be `tomcat6w //ES//JIRA120312230938`



5. Click on the `Java` tab to see the list of current start-up options:



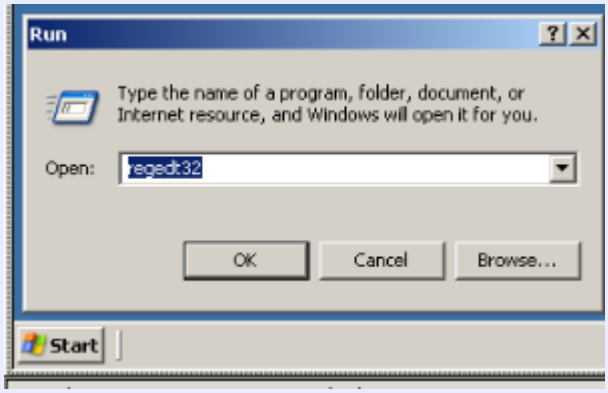
6. Set the maximum memory allocation here

Setting properties for Windows services via the Windows registry

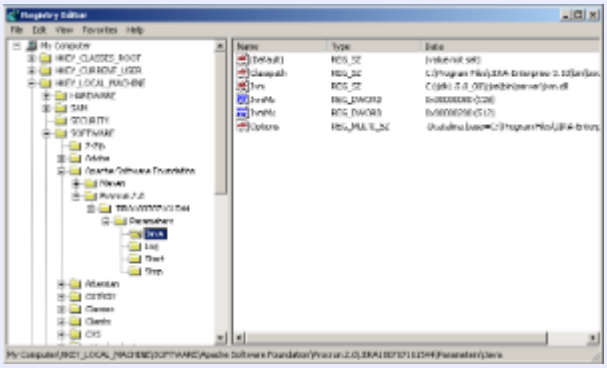
In some versions of Windows, there is no option to add Java variables to the service. In these cases, you must add the properties by viewing the option list in the registry.

To set properties for Windows services via the Windows registry

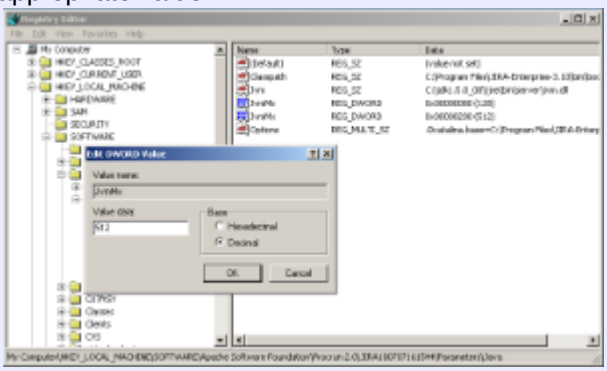
1. Go to `Start > Run`, and run "regedit32.exe".



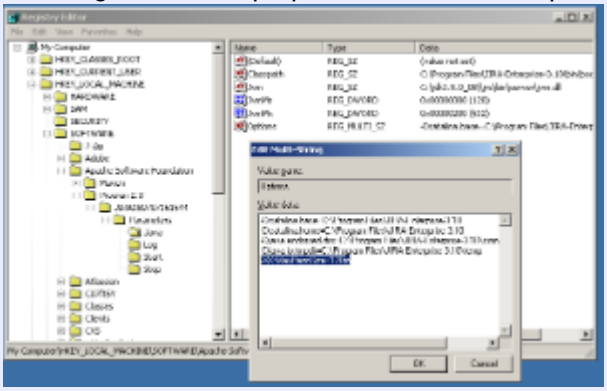
2. Find the Services entry:
   32-bit: `HKEY_LOCAL_MACHINE > SOFTWARE > Apache Software Foundation > Procrun 2.0 > JIRA`
   64-bit: `HKEY_LOCAL_MACHINE > SOFTWARE > Wow6432Node > Apache Software Foundation > Procrun 2.0 > JIRA`



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.

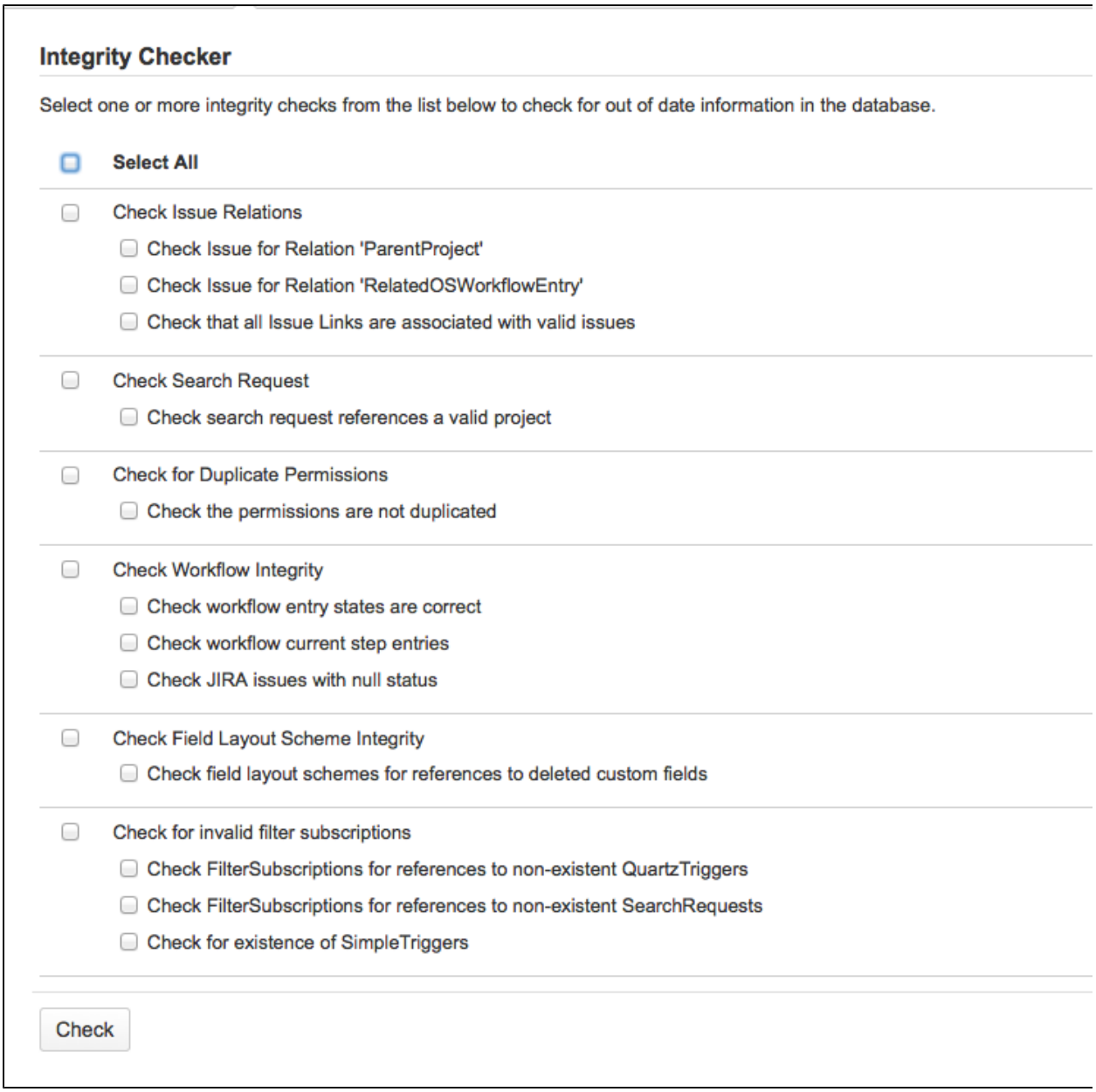
- Before migrating a project to a new workflow
- An external program is modifying JIRA's databasee
- Troubleshooting a server crash

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 **Troubleshooting and 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.



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.
   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

On this page:
- Logging
- Profiling

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:

   ```
   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 that 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 **Troubleshooting and 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`).
   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.

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 analyse the performance traces that JIRA will produce for every request. An example of a profiling trace is shown below:

```
[Filter: profiling] Turning filter on [jira_profile=on]
[116ms] - /secure/Dashboard.jspa
  [5ms] - IssueManager.execute()
    [5ms] - IssueManager.execute()
      [5ms] - Searching Issues
  [29ms] - IssueManager.execute()
    [29ms] - IssueManager.execute()
      [29ms] - Searching Issues
        [28ms] - Lucene Query
          [23ms] - Lucene Search
```

Profiling can be enabled either

- temporarily — profiling will be enabled until you next restart JIRA, or
- permanently — profiling will remain enabled, even after you restart JIRA.

Temporarily enabling profiling

1. Choose  **> System**.
2. Select **Troubleshooting and 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-cl
ass>
            <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-8O
        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.

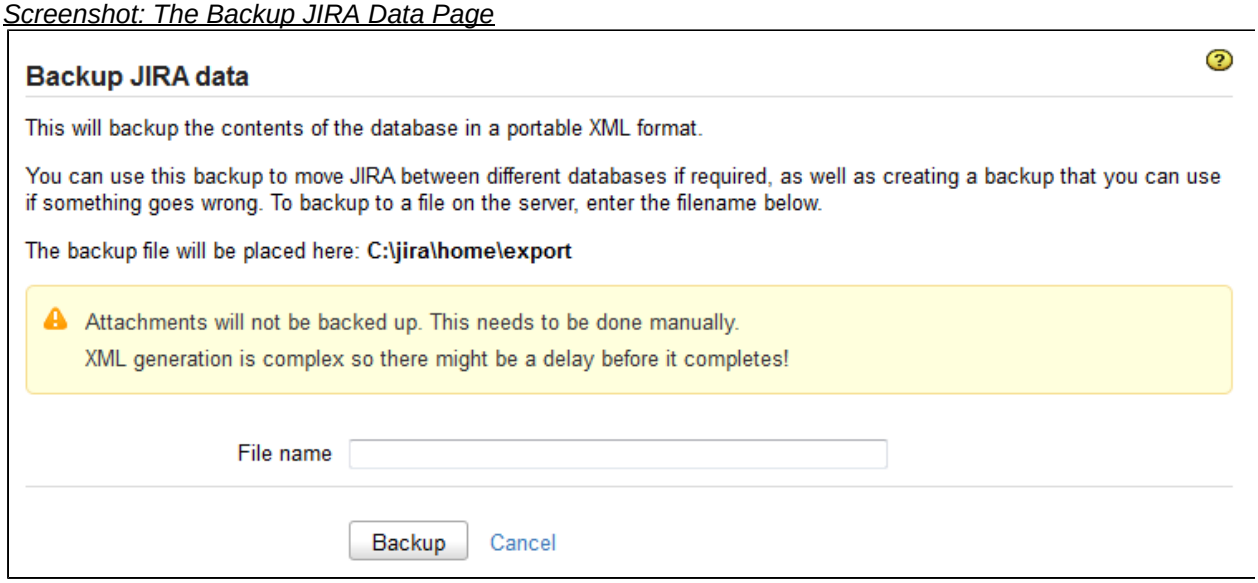
Using JIRA's XML backup utility

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



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.
Ensure that JIRA has the necessary file system permissions to write to this location. See the Create a dedicated operating system account section for more information.
4. Click the **'Backup'** button and wait while your JIRA data is backed up.
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

The `data` directory is a sub-directory of your JIRA application home directory. It contains application data for JIRA, e.g. if you have attachments enabled, all files attached to JIRA issues are stored in the `data\attachments` directory (not in the database).

To back up the `data` directory, you need to create a snapshot of the `data` directory (including all files and subdirectories), then back up the snapshot. Note that the directory structure under the `data` directory **must** be preserved in the snapshot.

Creating this snapshot is an operating system-specific task, e.g.:

- On MS Windows, a batch script copying the directory can be written and scheduled periodically (Programs > Accessories > System Tools > Scheduled Tasks).
- On Linux/Solaris, it is best to write a small shell script, placed in `/etc/cron.daily` , backing up files to a directory like `/var/backup/jira` . It is best to copy an existing script in `/etc/cron.daily` to ensure local conventions (file locations, lockfiles, permissions) are adhered to.

Your "attachments" directory may be located elsewhere
If you have put your `attachments` directory in a custom location rather than inside the `data` directory, you will also need to back up your `attachments` directory using the snapshot method described above.

Automating JIRA application backups

JIRA applications can be configured to automatically create an XML backup of JIRA application data on a routine basis.

Please note:

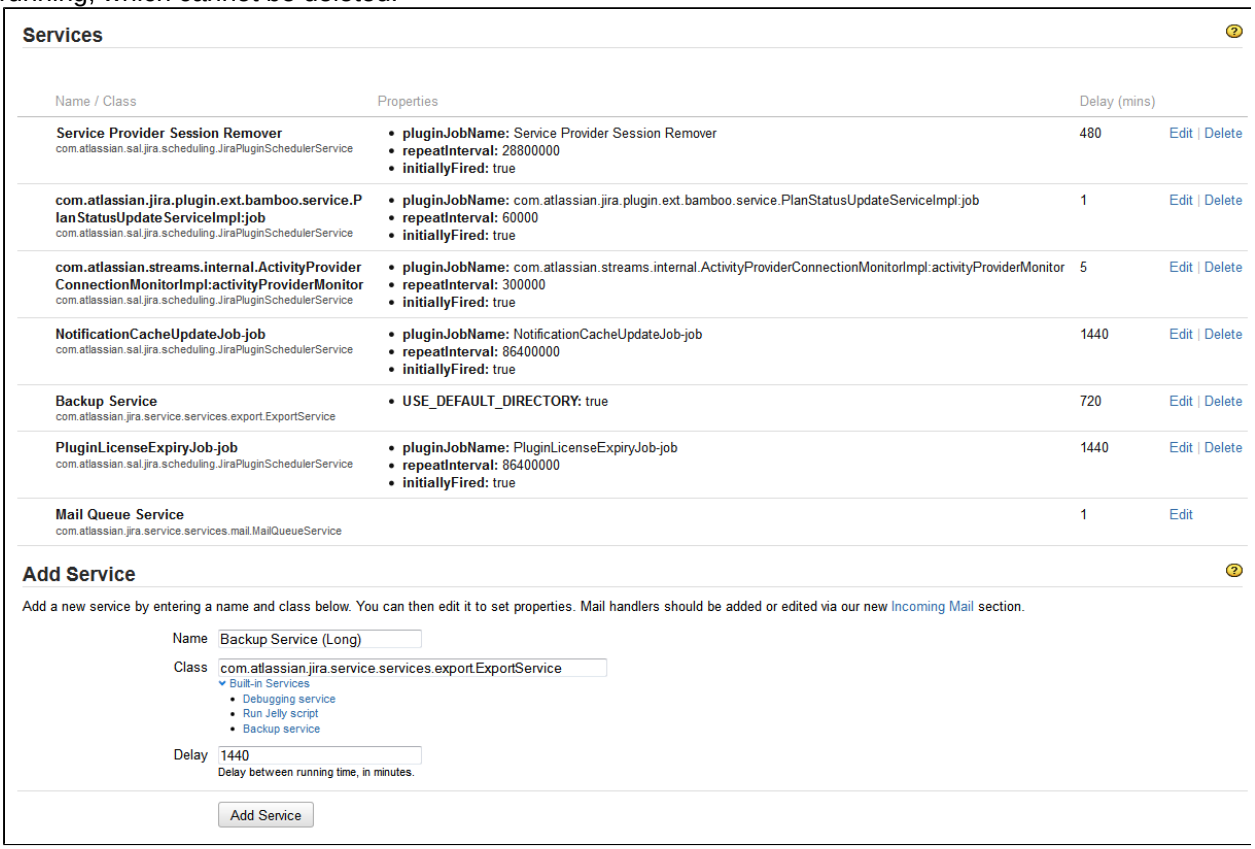
- The XML backup includes all data in the database. However, it **does not include** your attachments directory, JIRA application home directory, or JIRA application installation directory, which are stored on the

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.

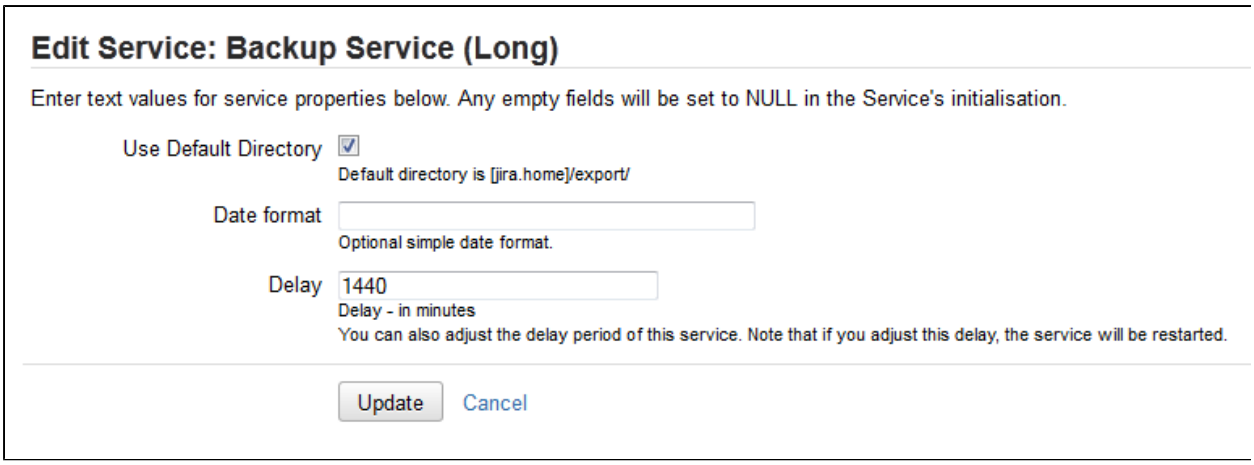
> For production use or large JIRA application installations, it is **strongly recommended** that you use native database-specific tools instead of the XML backup service. XML backups are not guaranteed to be consistent, as the database may be updated during the backup process. Inconsistent backups are 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.



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:
     - Select the **Backup service** from the list of JIRA application **Built-in Services**. To do this:
       a. Click the **Built-in Services** link below the **Class** field to expand the list of JIRA application built-in service classes.
       b. 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`'
     - 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.



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 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.
  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

The Project Import tool will automatically create all issue links between issues within your backed up project. It will also try to create links between the backup project and another project, as long as the other project already exists in your target JIRA instance with the relevant issue keys. If the source/target of a link cannot be found (i.e. the entire project or the particular issue may be missing), the link will not be created although the project will still be imported.

Note that the Project Import tool will create issue links between projects in either direction (source to target, or target to source). This means that if you import two projects from the same backup file, the second project import will create all of the links between the two projects that were missing from the first project import.

Once you have completed as many of the setup tasks as you are able to, run the Project Import tool.

Project Import

Restoring your project is a four step process:

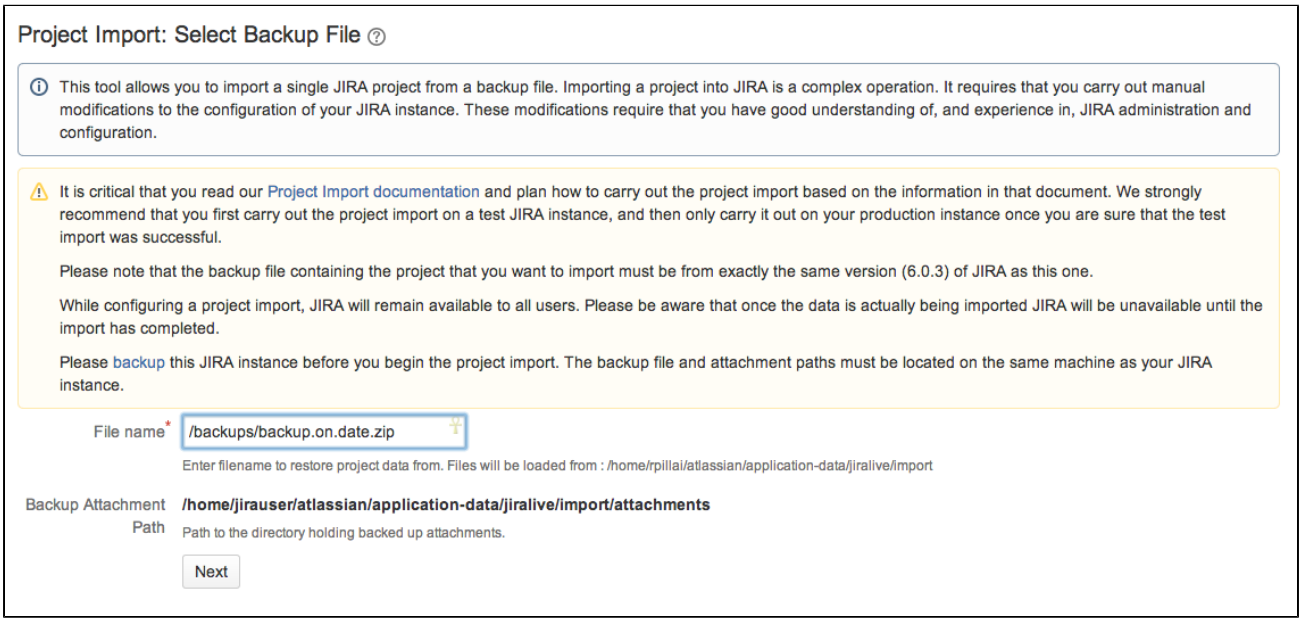
1. Specify the backup file
2. Select a project
3. Review data mapping validations
4. Verify the restored project

If you start the Project Import tool, we strongly recommend that you complete all steps of the wizard before performing any other activities in JIRA. Please be aware that it can take some time to validate the data mappings and then import the project.

You will most likely need to navigate away from the Project Import wizard to correct your JIRA configuration, as advised by validation errors in the wizard. If you have to navigate to other pages in JIRA to correct your JIRA configuration or for other activities, you should:

- ***(recommended)*** open a separate session of JIRA in a new browser window/tab. When you return to the Project Import wizard in the original browser window/tab, you can use the **'Refresh validations'** b utton 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

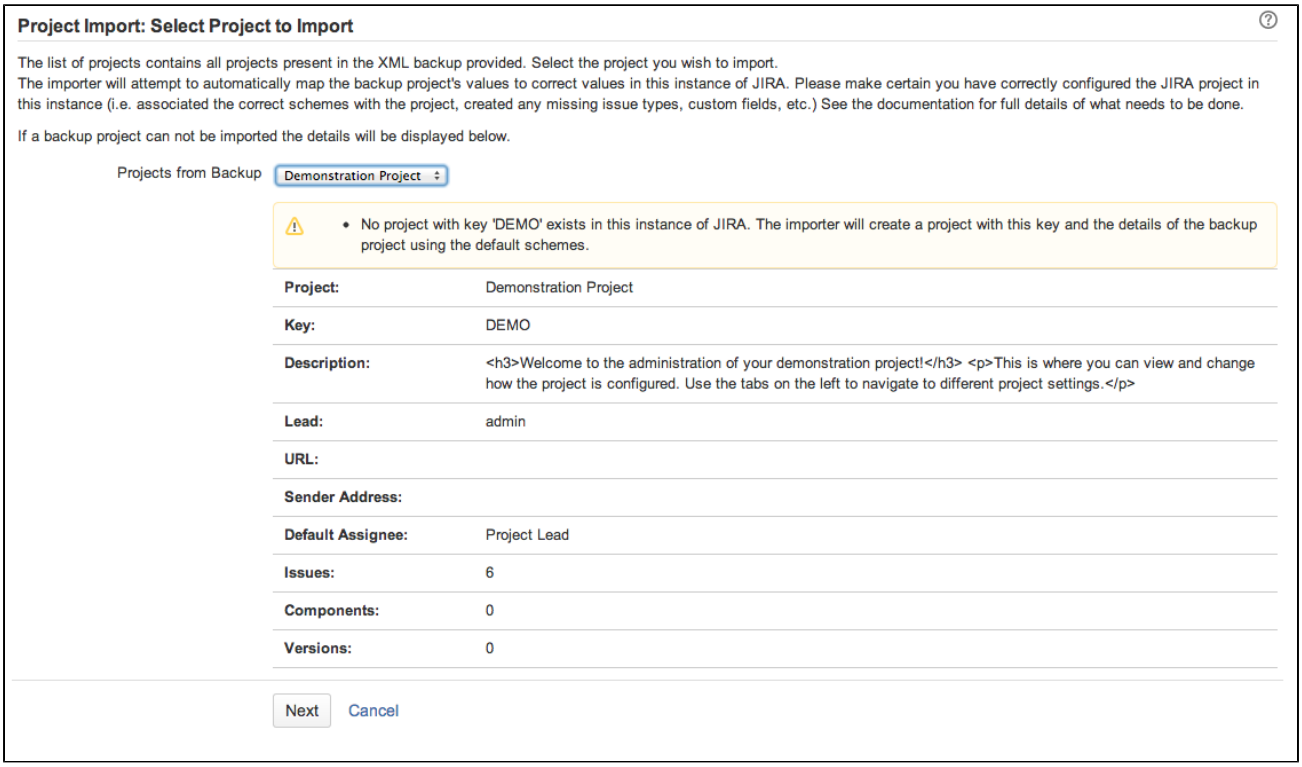


To start the Project Import tool:

1. Choose  **> System**.

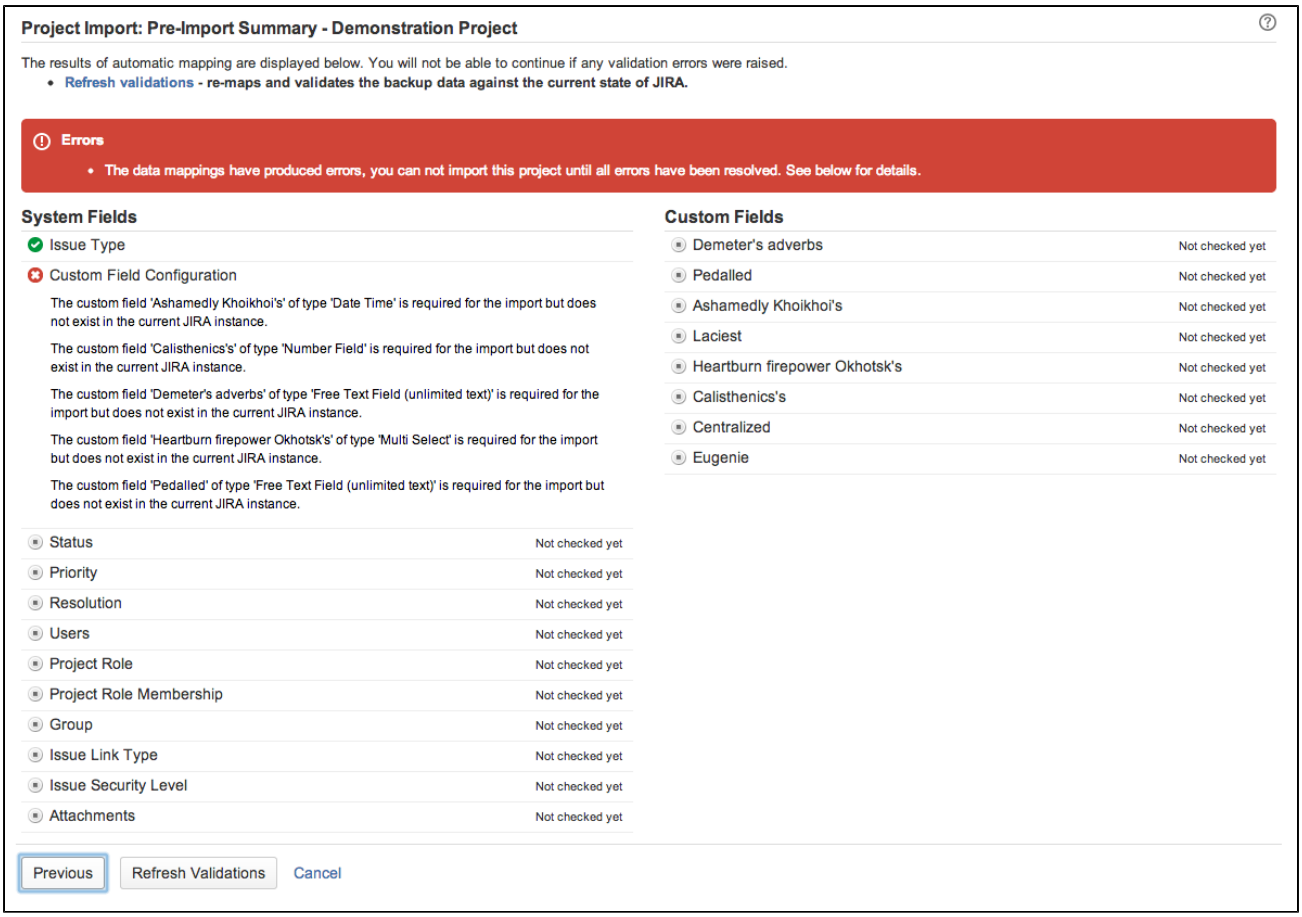
2. Select **Import & Export > Project Import** to open the Project Import wizard page.
3. Specify the path and name of your backup file in the 'File name' field. Your backup file must be an XML or ZIP file (as exported by JIRA).
4. Copy the attachments from the path where you have backed up the attachments to the **'Backup Attachment Path'** shown in the import window. This path is under the JIRA home directory of the instance. Please not 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



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.

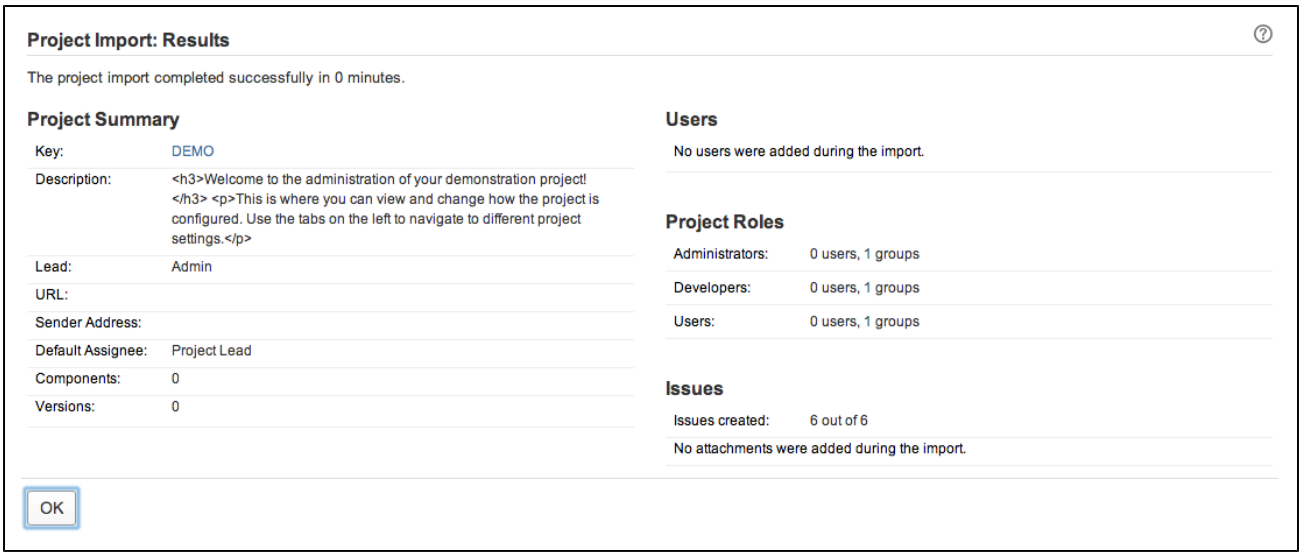
3. Review data mapping validations



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



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'** se ction 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 **Administr ation** tab), so that we know the details of your JIRA configuration.

You can anonymise 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 anonymising tool to handle this particular scenario.

Anonymising 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 anonymised - please do not attempt to anonymise 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 Anonymiser.
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. Anonymise the backup file with the below commands:

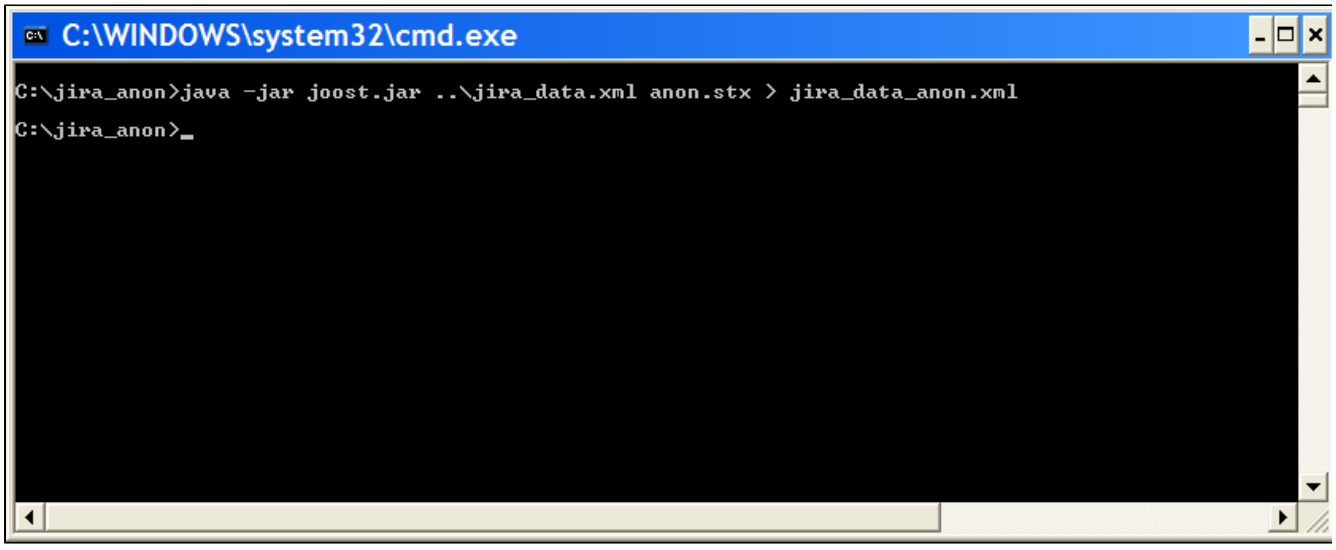
```
$ java -Xmx512m -jar joost.jar <JIRA BACKUP>.xml anon.stx > <NAME
OF ANONYMISED BACKUP>.xml
```

   For example, this would be anonymising 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 anonymised 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:



Information about the Anonymiser

The anonymiser 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 anonymised backup, `anon-backup.xml`, to ensure it's clean enough for the needs of your organisation 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.senddisabled=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.fetchdisabled=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):

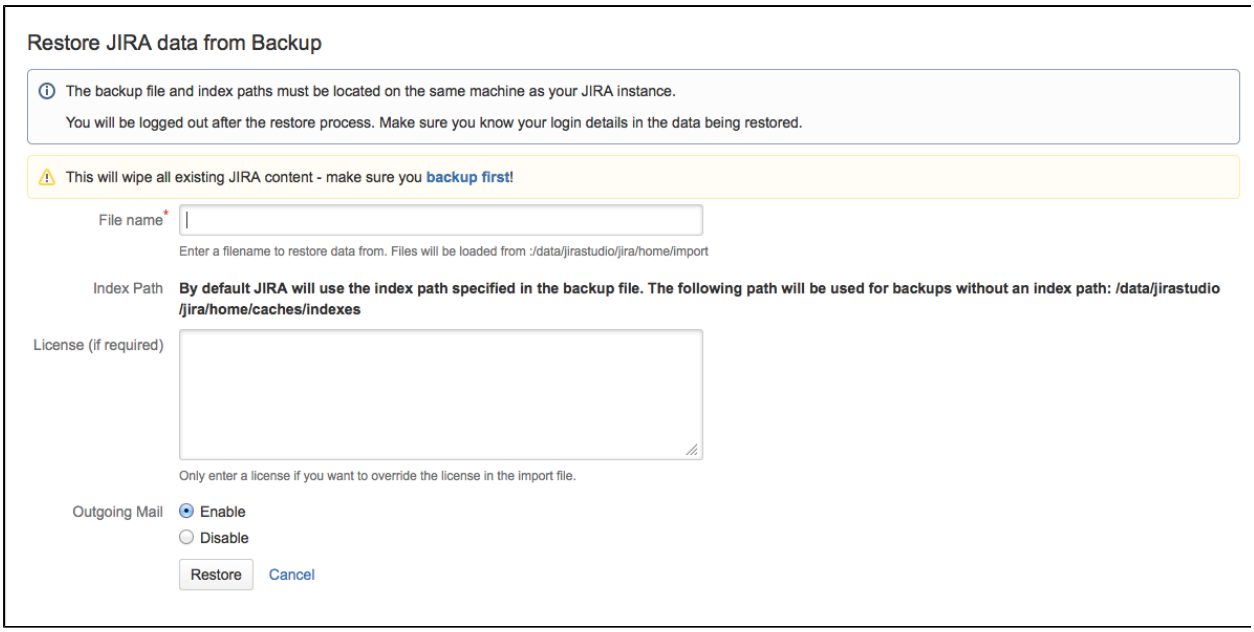
```
set DISABLE_NOTIFICATIONS=" -Datlassian.mail.senddisabled=true
-Datlassian.mail.fetchdisabled=true -Datlassian.mail.popdisabled=true"
cd bin
startup.bat
```

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.



3. In the **'File name'** field, type the file name of the zipped XML backup file generated by JIRA.
   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.
   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.
   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.

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.
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.senddisabled=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.fetchdisabled=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.fe
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 if issues have been manually entered into the database, or the index has been lost or corrupted.

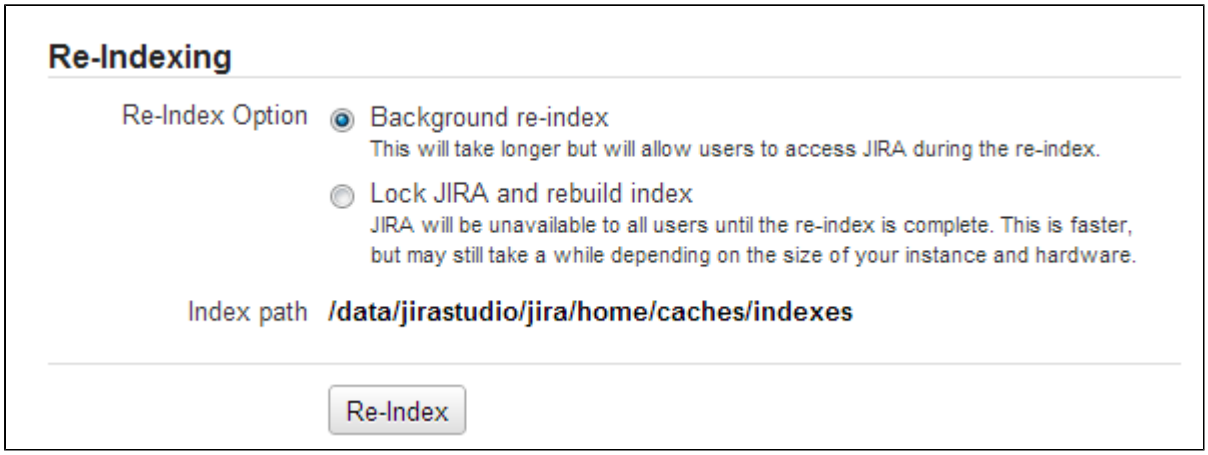
See Re-indexing after major configuration changes for more information on when you should re-index.

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



Which re-indexing option should I use?

The **Background re-index** option should be used in the majority of circumstances, particularly following changes to the configuration. It will generally take significantly longer to perform than the **Lock JIRA and rebuild index** option, but it allows JIRA to remain usable while it is being done. There will however be a performance impact on JIRA as a whole. *We recommend that you perform this option during a low usage period*. The actual impact of running the **Background re-index** option will depend upon the customer's particular hardware and software installation as well as how many issues are in the system.

The **Lock JIRA and rebuild index** should be used when:

- the indices are corrupt, which may be caused by a system or disk failure or
- it is more important to have the re-index completed quickly than to have JIRA continuously available. The **Lock JIRA and rebuild index** option may be in the order of twice as fast as a background re-index.

The following table summarises the differences between the two options:

Background re-index	Lock JIRA and rebuild index
Slower to complete.	Faster to complete (may be up to twice as fast).
JIRA can be used by users during re-index.	JIRA cannot be used by users during re-index.
Can be cancelled at any time.	Cannot be cancelled once started.

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 `caches/indexes` subdirectory of the JIRA application home directory.

NFS storage for JIRA indexes is not supported. See Supported platforms for more information.

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-INSTALL/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/sr/`. However, you still need to put `robots.txt` file in the root directory, i.e. `jira.mycompany.com/robots.txt` (not `jira.mycompany.com/jira/robots.txt`).

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

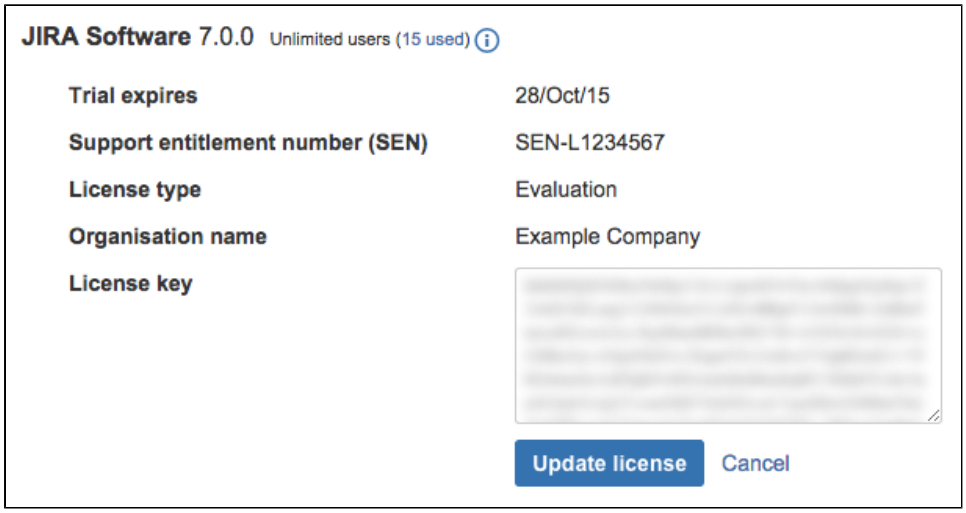
license expires)
- upgrade your user tier to accommodate new users
- add a new license when your old license has expired
- add a new license for a newly installed application

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.



The updated license must be compatible with any other license(s) on your JIRA server.

To update your JIRA license key manually

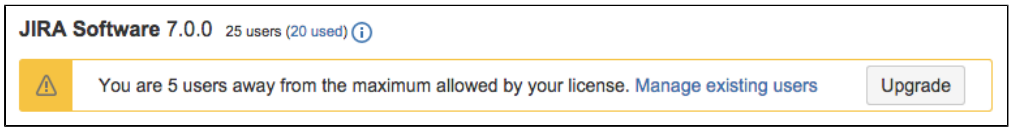
1. Obtain the license key you want to update (you may do this by logging in to http://my.atlassian.com separately, or contacting your organization's member who deals with licensing for IT products).
2. Choose
   > Applications.
3. Select **Versions and licenses** to view your existing JIRA applications 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 **Troubleshooting and 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 version.
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 f or 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 beeen 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% utilisation)** 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 licences, 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 `entityengine.xml` 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.

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 with the **JIRA Administrators** global permission.

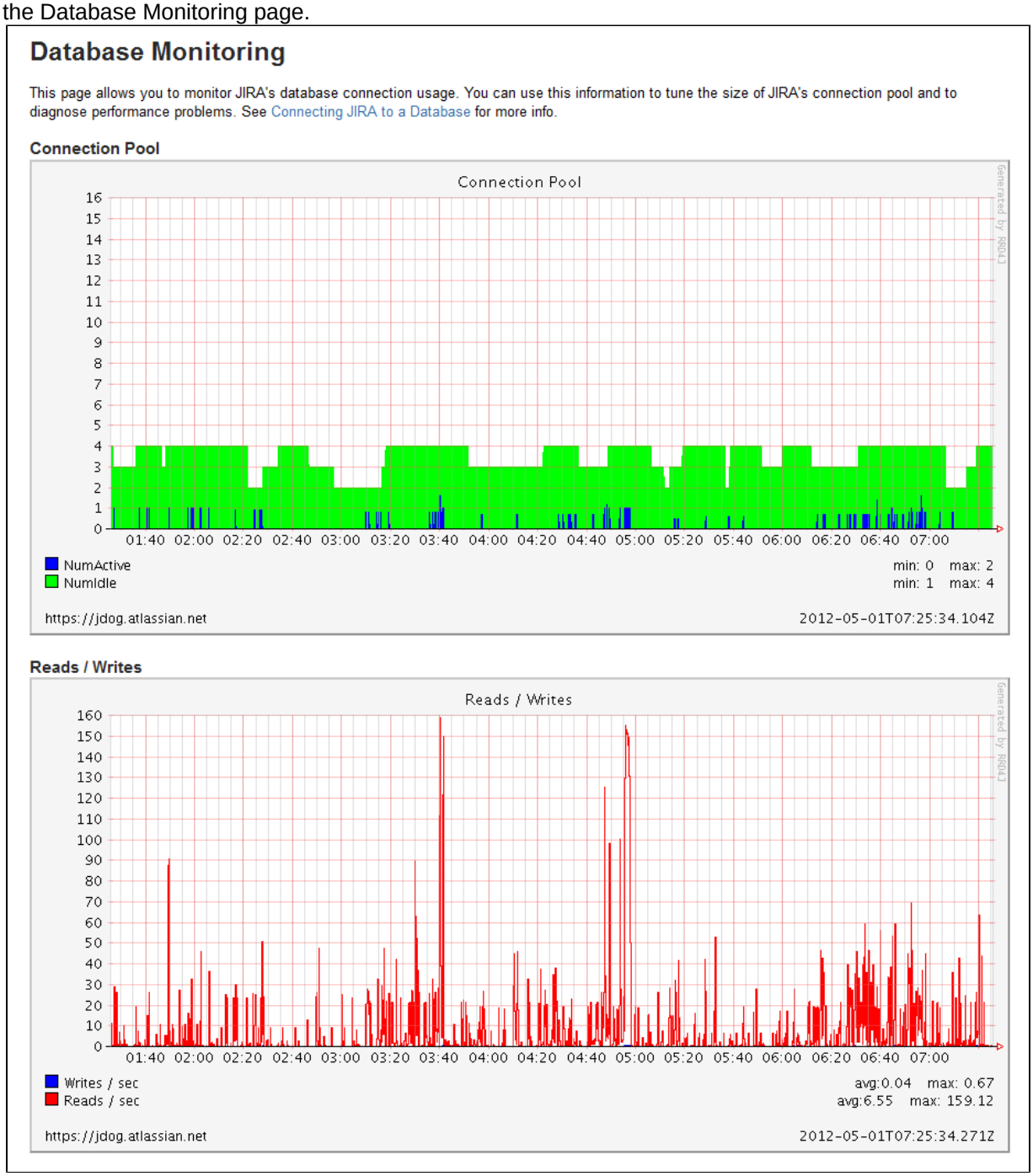
On this page:

Related pages:

- Tuning database connections
- Enterprise Resources

Accessing the Database Monitoring page

1. Choose  **> System.**
2. Select **Database Monitoring,** which can be found under **Troubleshooting and 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 optimise 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 **Troubleshooting and 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.**

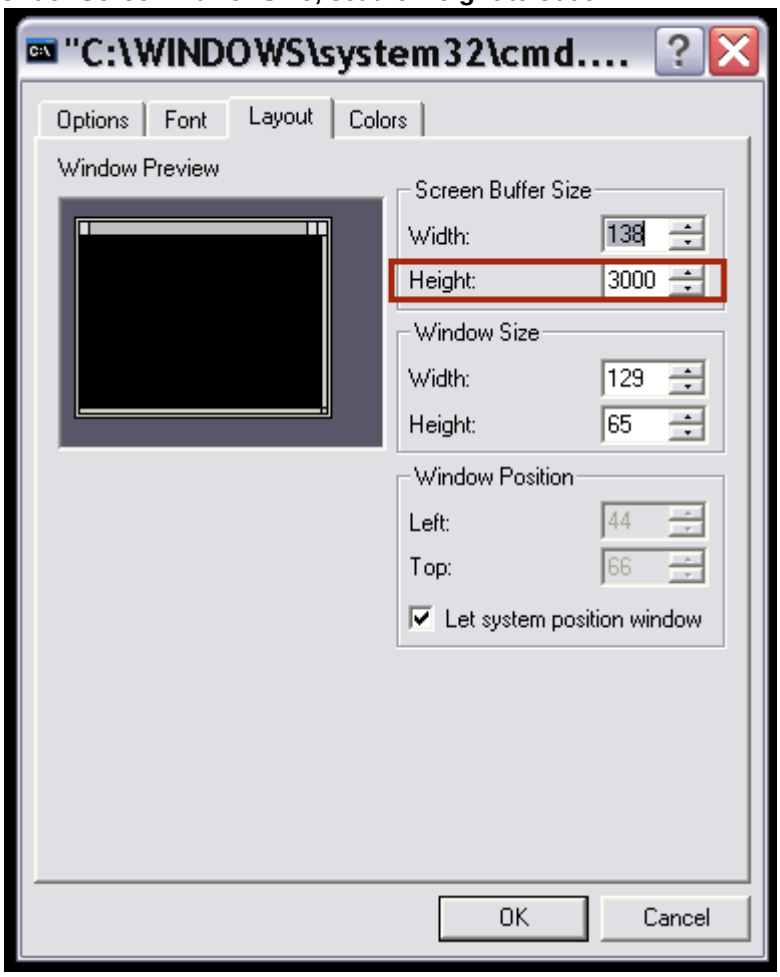
The steps necessary to retrieve the **thread dump** are dependant on the operating system JIRA is running in — please follow the appropriate steps below.

On this page:

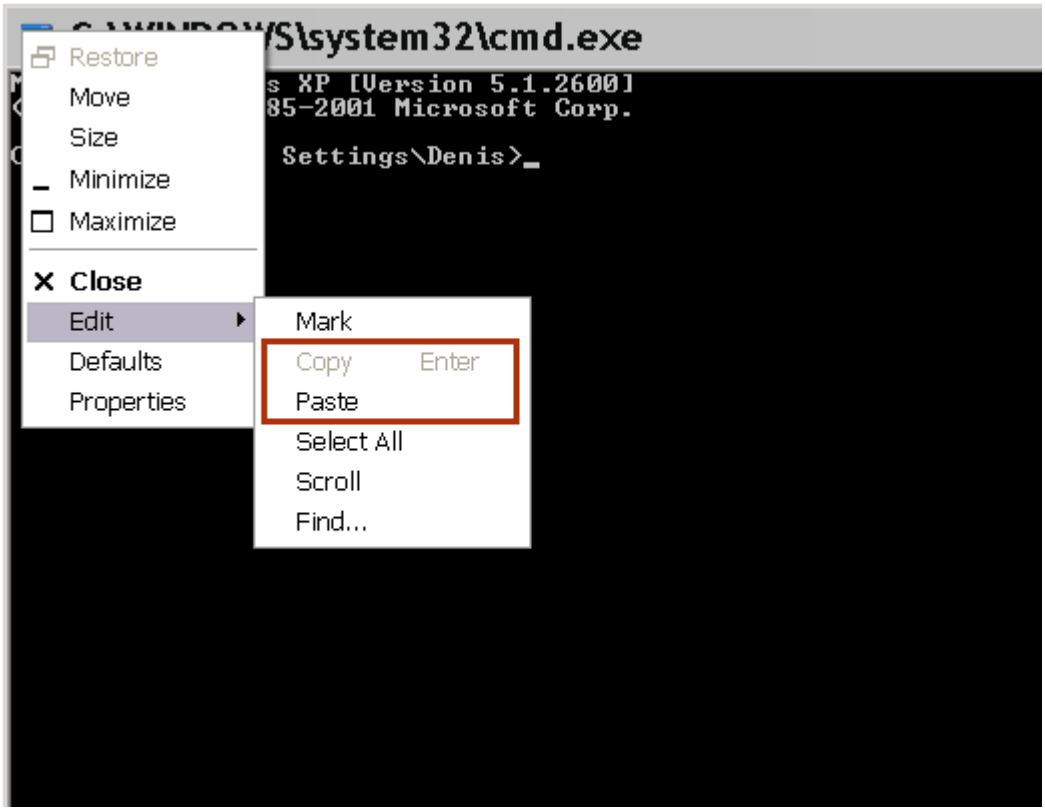
Windows environment

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` chec k 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.ClassLoaderLogManager
-Djava.awt.headless=true
-Djava.endorsed.dirs=/tmp/atlassian-jira-enterprise-3.6-standalone/common/e
ndorsed
-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

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.

Check the known thread dump knowledge base articles:

- Poor performance due to limited database connection pooling
- Searching, Indexing, and filters troubleshooting
- JIRA Deadlocks when Running Tomcat 6.0.24
- OutOfMemory or Poor Performance due to XML View of a Filter
- JIRA applications performance tuning
- JIRA applications crash due to OutOfMemoryError Java heap space

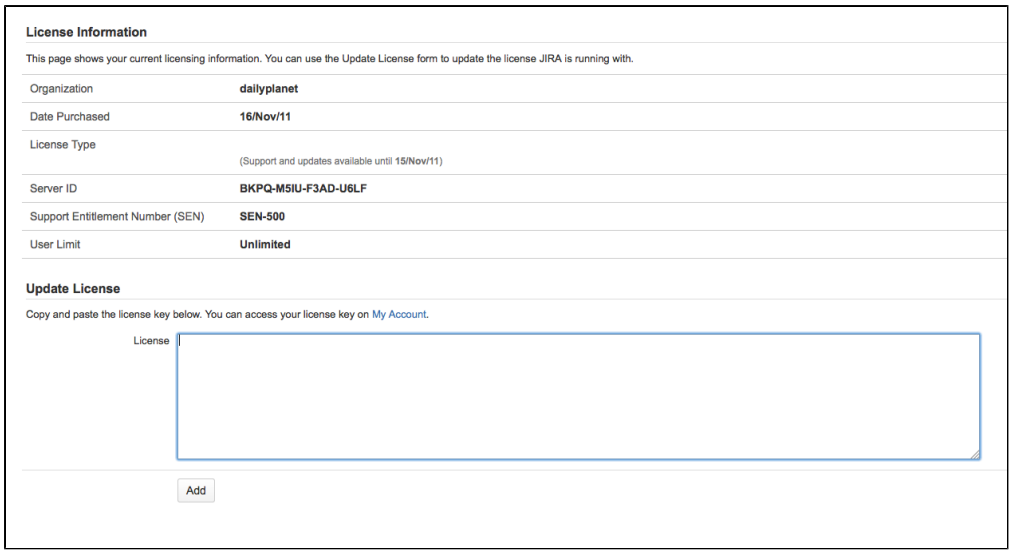
Finding your JIRA application Support Entitlement Number (SEN)

There are three ways to find your Support Entitlement Number (SEN).

See Finding Your Support Entitlement Number in the support space for more general information about how Atlassian Support uses this number.

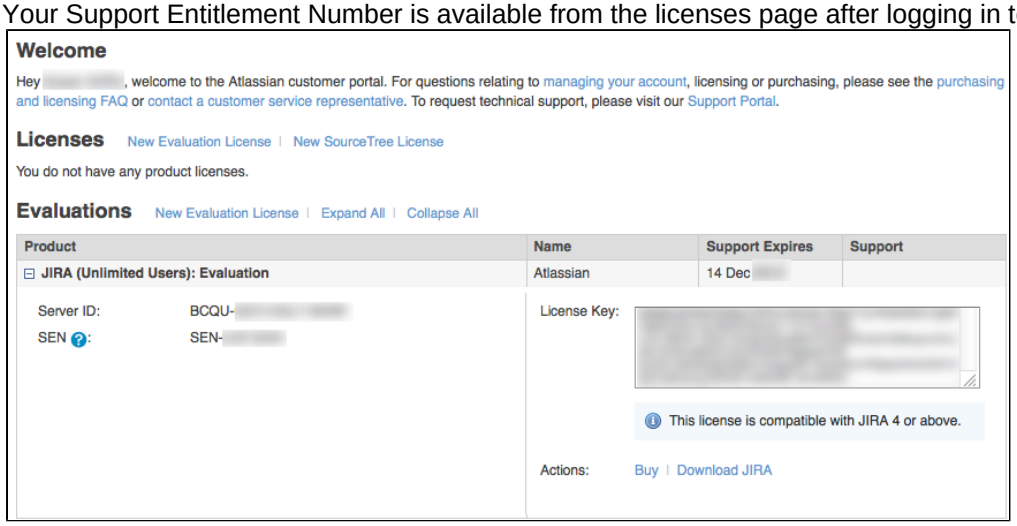
Method 1: Check in the JIRA administration interface

Access the JIRA license page, as described in Licensing your JIRA applications. The JIRA license page will show your Support Entitlement Number (SEN).



Method 2: Check my.atlassian.com

Your Support Entitlement Number is available from the licenses page after logging in to http://my.atlassian.com:



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

On this page:

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

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

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', 'Linux Installer', or from an 'Archive File'.
> - 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 datasource.

> 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.

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 util 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', 'Linux Installer', or from an 'Archive File'.
> - 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` director y. 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 datasource.

> 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.

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 util 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 used by one JIRA installation.

On this page:

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 your 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 **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.

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
4. *(Optional)* Configure IIS to forward Confluence requests to Tomcat (if you are using both Confluence and JIRA)

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"
```

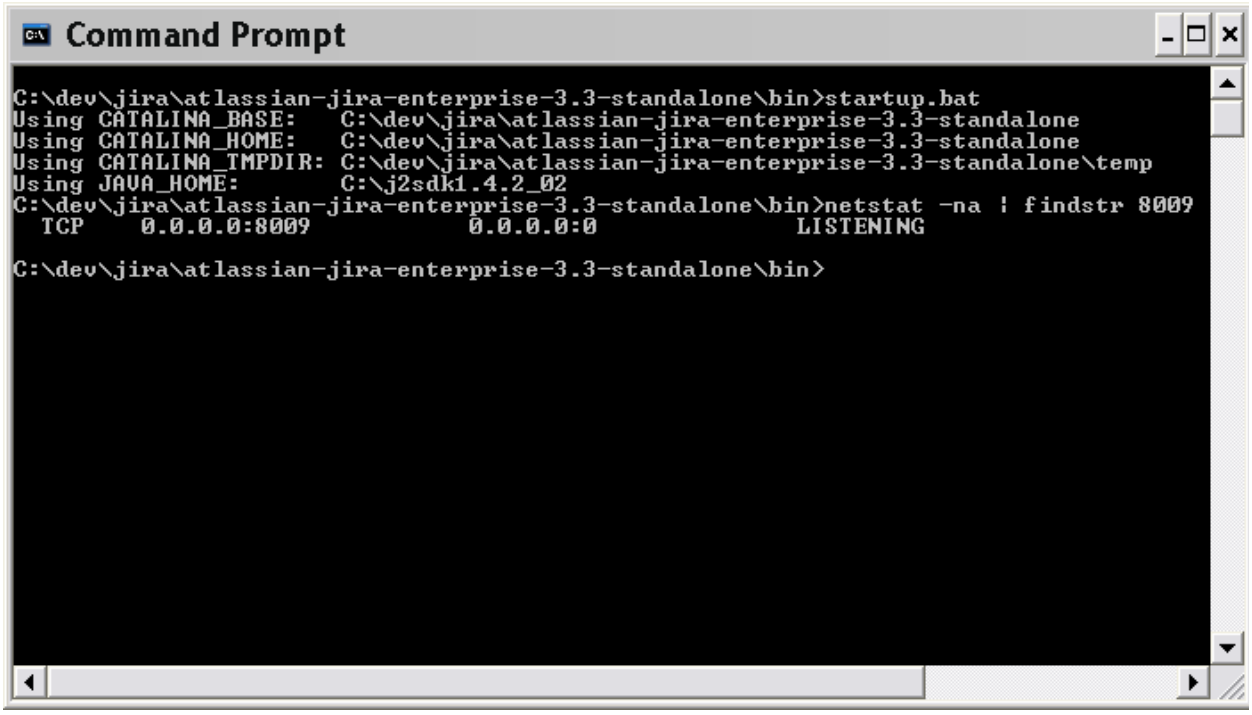
Please refer to the Integrating JIRA with Apache for reference.

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.

2. Restart Tomcat and ensure that no errors regarding used ports appear in the logs or in the Tomcat Console.
3. 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:



3. Configure IIS to forward requests to JIRA

On the machine where IIS is deployed:

1. 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).

2. 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.
3. 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.
4. 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.
5. 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:
   a. extension_uri — the path to the virtual directory that contains the **isapi_redirect.dll**
   b. log_file — the path to write the log file to
   c. 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.propertiesl` 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 `jira` .
    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`.

4. Configure IIS to forward requests to Confluence as well as JIRA

You can configure IIS so that it forwards requests to both JIRA and Confluence.

The following instructions describe how to forward from IIS to separate instances of JIRA and Confluence, running in separate Tomcat servers. The instructions assume that you have already set up IIS to forward to JIRA as described in section 3 above. The instructions also assume that you have already installed Confluence as per the Confluence Installation Guide.

The instructions describe how to make JIRA available under `www.example.com/jira` as described above, and Confluence available under `www.example.com/confluence`.

1. If JIRA and Confluence are running on the same machine, ensure that Confluence is listening on a different port to JIRA:
    a. Edit the `conf/server.xml` file.
    b. At the top of the file, change the `port` attribute of the `Server` element to a different port to the value for JIRA. For example, change it from `8005` to `8006`.
    c. Still in the `Server` element, Change the `port` attribute of the `Connector` sub-element to a different port to the value for JIRA. For example, change it from `8080` to `8090` .
2. Change the Confluence context path:
    a. Edit the `conf/server.xml` file `jira.xml` file.
    b. Change the `path` attribute of the `Context` element to "`/confluence`".
3. Restart Confluence after changing the ports and the context path, and test that Confluence works correctly by pointing your web browser at http://localhost:8090/confluence.
4. Configure Confluence to accept proxied requests: Remove the comments around the AJP/1.3 Connector section in the Confluence `conf/server.xml` or `jira.xml` file and change the `port` attribute to a value different to the value for JIRA. For example, change it from `8009` to `8010`.
5. Restart Confluence and ensure that no errors regarding used ports appear in the logs or in the Tomcat console.
6. Edit the `uriworkermap.properties` file and add the following mapping:

    ```
    /confluence/*=worker2
    ```

    The file should now contain the following mappings:

    ```
    /jira/*=worker1
    /confluence/*=worker2
    ```

7. Edit the `workers.properties.minimal` file:
    Change the line starting with `worker.list` to the following:

    ```
    worker.list=worker1,worker2
    ```

    Add the following lines to the end of the file (assuming the host is on the same machine as IIS and you changed the AJP/1.3 Connector port for Confluence to `8010`):

    ```
    worker.worker2.type=ajp13
    worker.worker2.host=localhost
    worker.worker2.port=8010
    ```

    The `workers.properties.minimal`file should now look like the following:

```
worker.list=worker1,worker2

#
# 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

worker.worker2.type=ajp13
worker.worker2.host=localhost
worker.worker2.port=8010
```

8. Create a **virtual directory** for Confluence in IIS. Set the `alias` to `confluence`. It can point to any directory.
9. Restart the IIS Service.
10. You are done! Confluence should now be available under `www.example.com/confluence`, and JIRA should still be available under `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

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, utilising 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 complication 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

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>
```

Ensure the `path` value is set with a prepending forward slash (`/`) . For example, `path="/jira"` r ather than `path="jira"`.

3. Edit Tomcat's `server.xml` to include a separate connector to proxy the requests. This requires the p `roxyName` & `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 working by accessing JIRA on the appropriate proxy connector, for example `http://jiraserver:8080/`. This should redirect to the proxy FQDN (in this example, http ://jira.atlassian.com), which will fail as the proxy is not yet configured. The test is to ensure Tomcat is set up to correctly redirect to the proxy.

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/
```

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
```

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
```

This is only required if a new instance has been created in favour 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/
```

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
```

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.
  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, utilising 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 favour 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

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 ModulesDebian/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 ModulesDebian/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/
```

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
```

This is only required if a new site has been created in favour 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/
```

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.
  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

- 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, utilising 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:

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>
```

Ensure the `path` value is set with a prepending forward slash (`/`) . For example, `path="/jira"` r ather than `path="jira"`.

3. Edit Tomcat's `server.xml` to include a separate connector to proxy the requests. This requires the `s cheme`, `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 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 working by accessing JIRA on the appropriate proxy connector, for example `http://jiraserver:8080/`. This should redirect to the proxy FQDN (in this example, https://jira.atlassian.com), which will fail as the proxy is not yet configured. The test is to ensure Tomcat is set up to correctly redirect to the proxy.

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 ModulesDebian/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.

2.2. Configure Apache to use those ModulesDebian/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-ssl`.
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>

SSLProxyEngine          On
ProxyRequests           Off
ProxyPreserveHost       On
ProxyPass               /       http://jiraserver:8080/
ProxyPassReverse        /       http://jiraserver:8080/
```

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
```

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. Enable the instance with the following:

```
# a2ensite default-ssl
Enabling site default-ssl.
To activate the new configuration, you need to run:
  service apache2 reload
```

6. Copy the certificate and private key to the appropriate directories.
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. *(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/
   ```

   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
   ```

   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 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>
```

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** function ality 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 AddSmtpMailServer>
  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 UpdateSmtpMailServer>
  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.

5. 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.

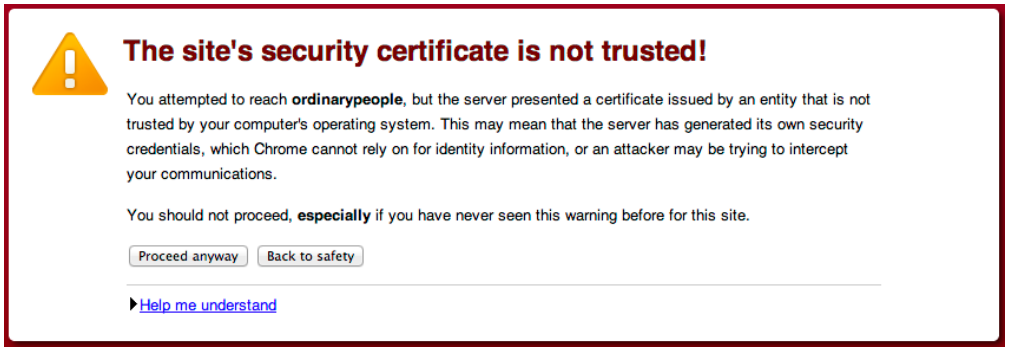
On this page:

Problem Symptoms

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.

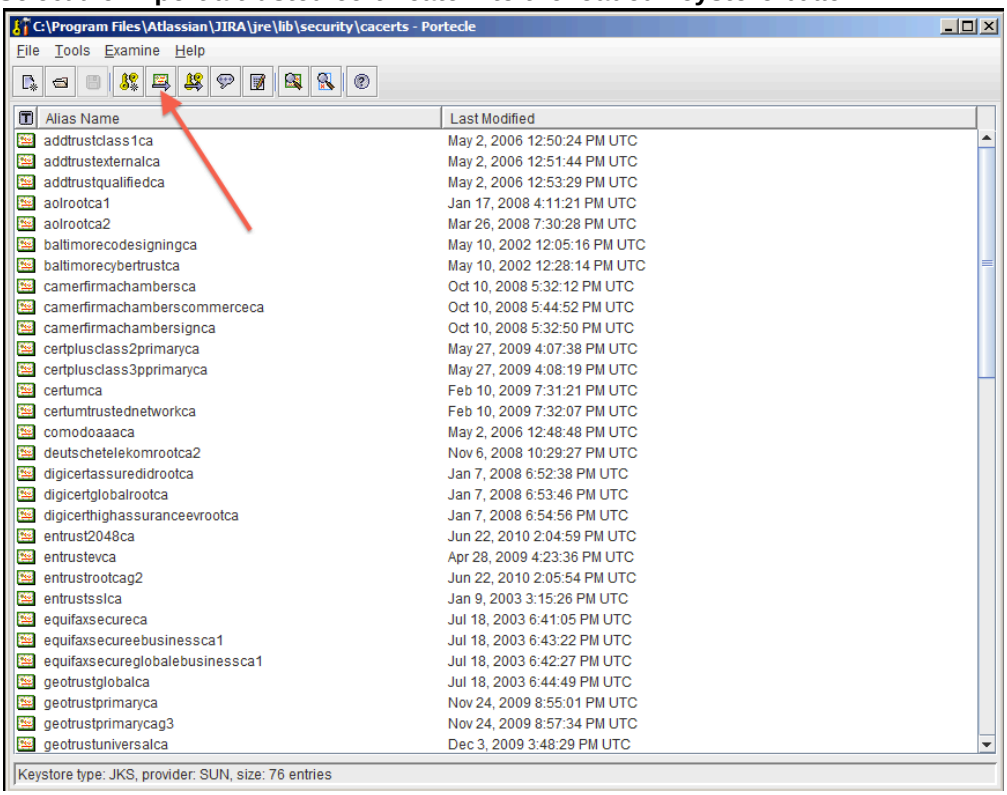
> If you're unable to install Portecle on the server or prefer the command line please see our Command Line Installation section below.

Obtain and Import the Server's Public Certificate

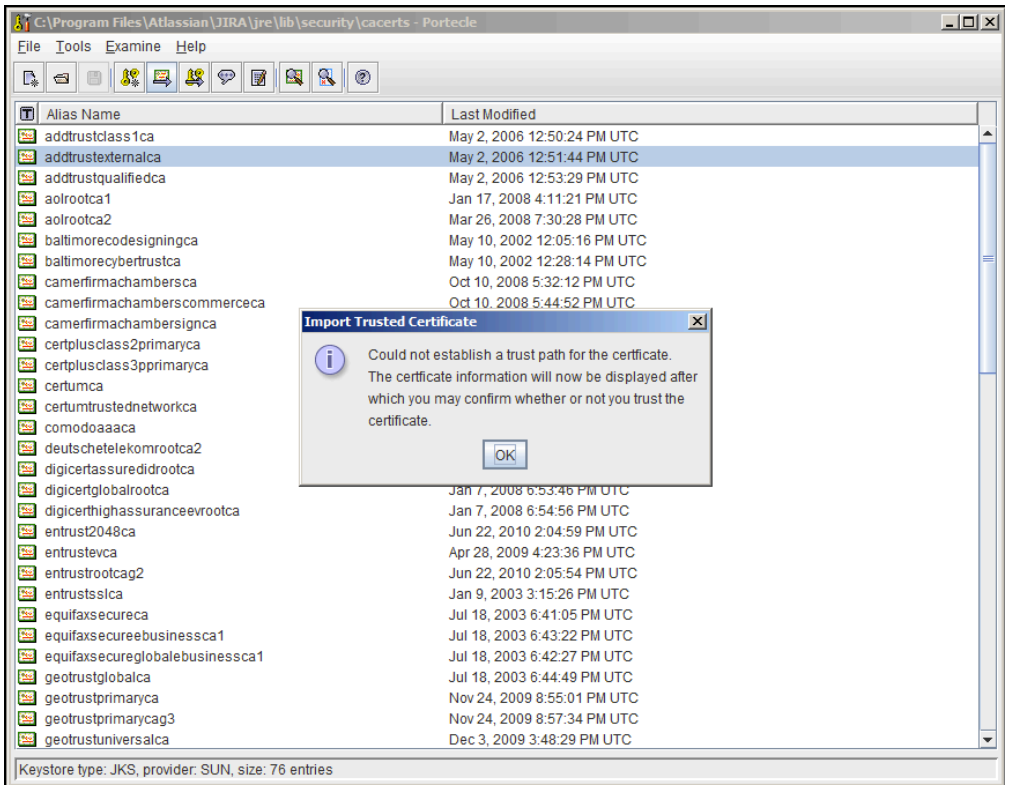
1. Download and install the Portecle app onto the server that runs JIRA.
   ⚠ This is a third-party application and not supported by Atlassian.
2. Ensure the <JAVA_HOME> variable is pointing to the same version of Java that JIRA uses. See our Setting JAVA_HOME docs for further information on this.
   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 `changeit`).
8. Select the **Import a trusted certificate into the loaded keystore** button:

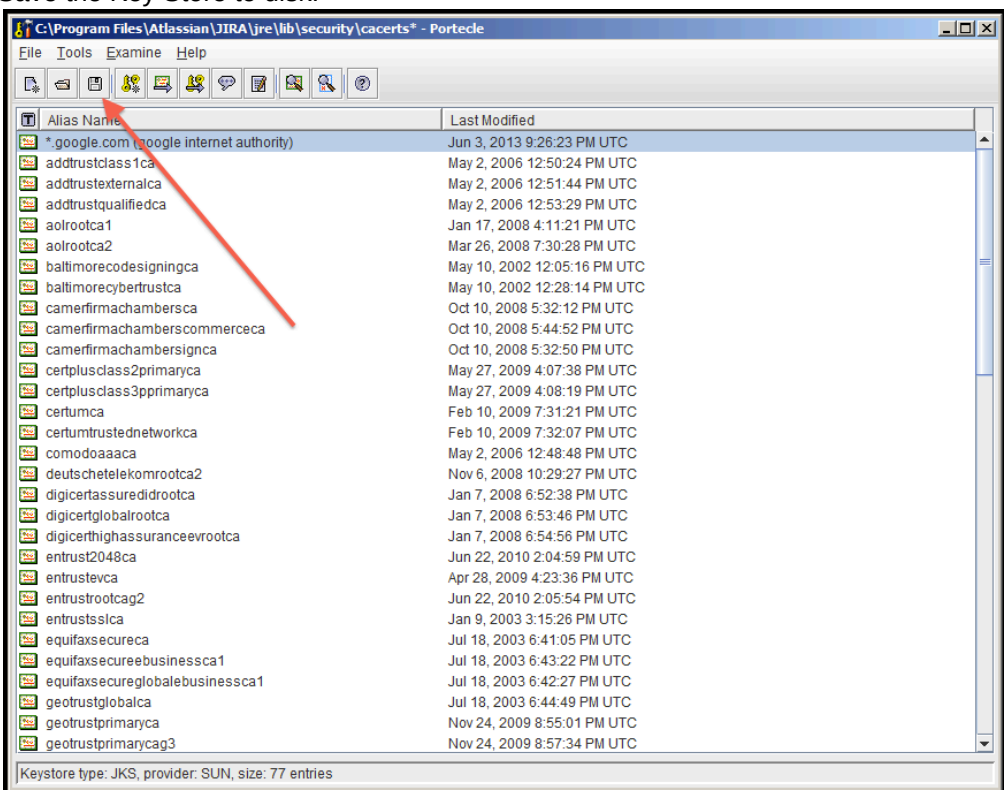


9. Select the certificate that was saved in step 6 and confirm that you trust it, giving it an appropriate alias (e.g.: confluence).
   a. You may hit this error:



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
   ```

   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.cert`. Edit the the `public.cert` file so it contains only what is between the `BEGIN CERTIFCATE` 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

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.

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 be applicable to your specific setup.

Why should you run JIRA over SSL or HTTPS? When web applications are being accessed across the internet, there is always the possibility of usernames and passwords being intercepted by intermediaries between your computer and the ISP/company. It is often 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.

On this page:

Related topics:

- Using the JIRA configuration tool
- Configuring JIRA options
- Integrating JIRA with Apache using SSL

Before you begin

Please note the following before you begin:

- 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.
- For JIRA installations installed using Windows Installer:
  - 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.
  - In this document, the term `<jira-install-dir>` refers to the JIRA application installation directory itself.

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)

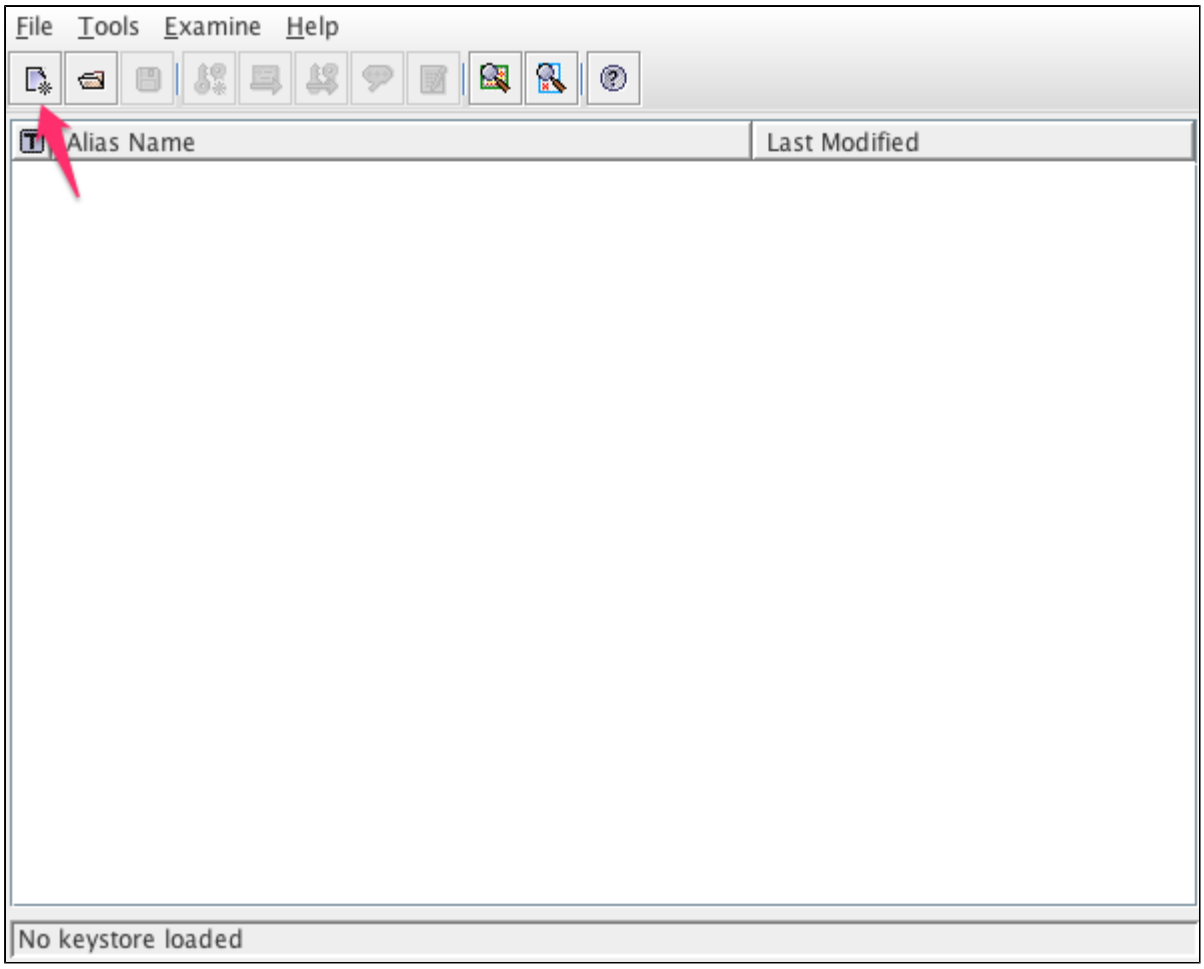
We recommend using a CA-signed certificate.

If you're unable to install Portecle on the server or prefer the command line, please see our Command Line Installation section below.

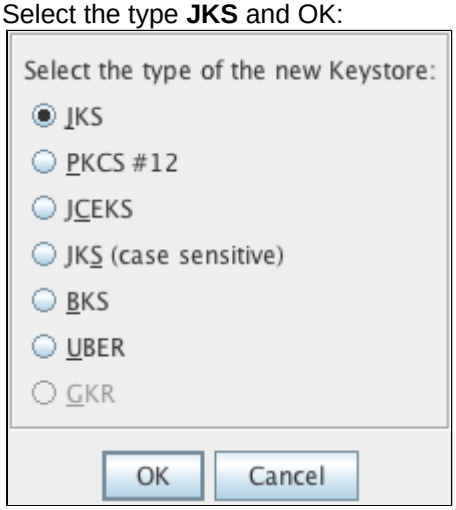
1. Download and install the Portecle app onto the server that runs JIRA.
   ⚠ This is a third-party application and is not supported by Atlassian.
2. Run the App as an Administrator, so it will have the appropriate permissions. Also, ensure the <JAVA_HOME> variable is pointing to the same version of 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
   ```

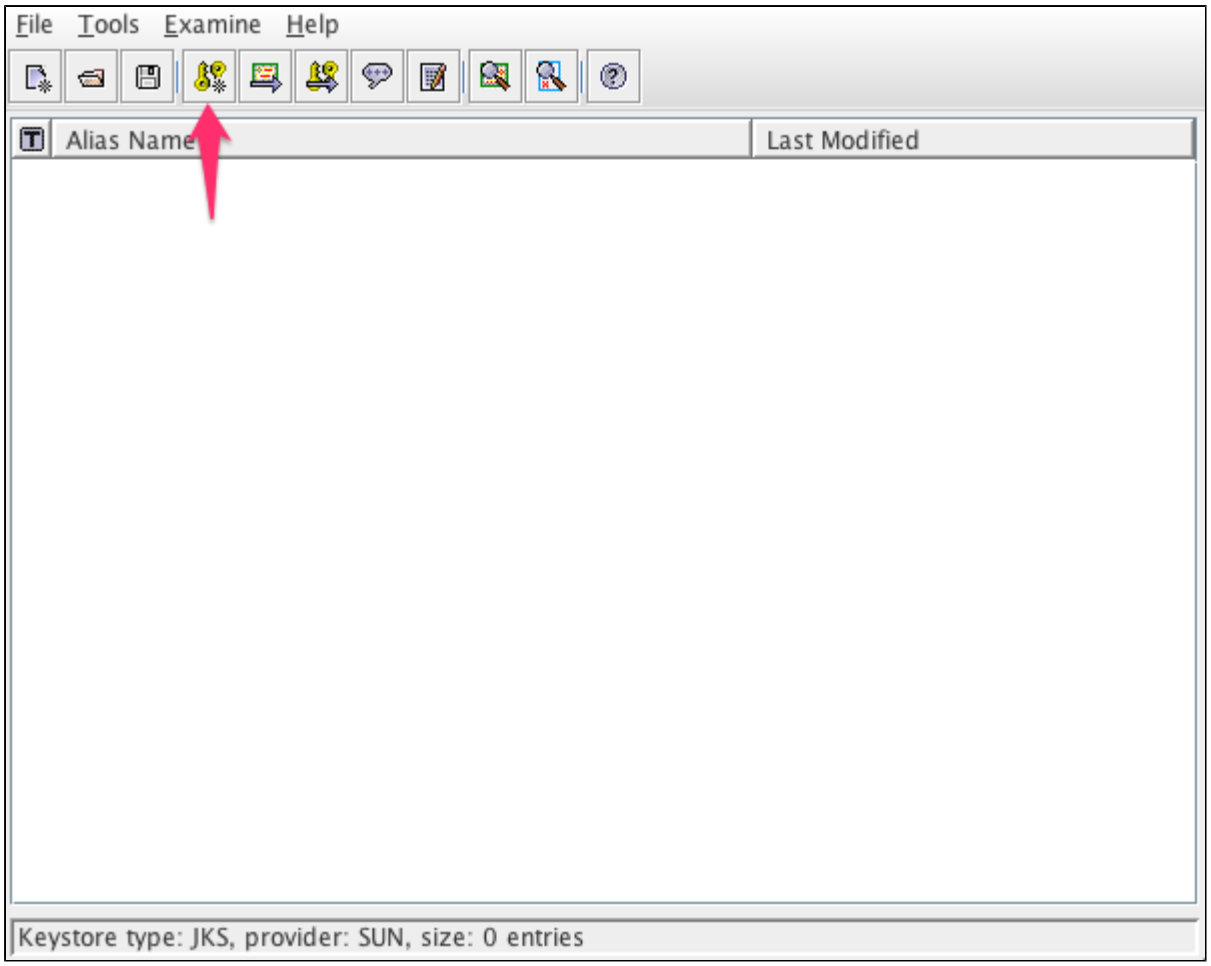
3. Select the **Create a new Keystore** option:



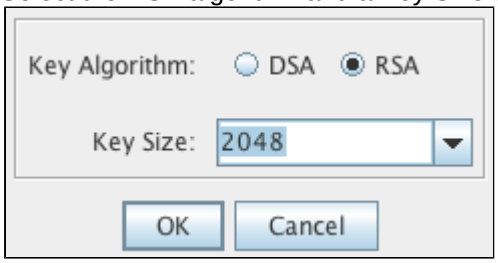
4. Select the type **JKS** and OK:



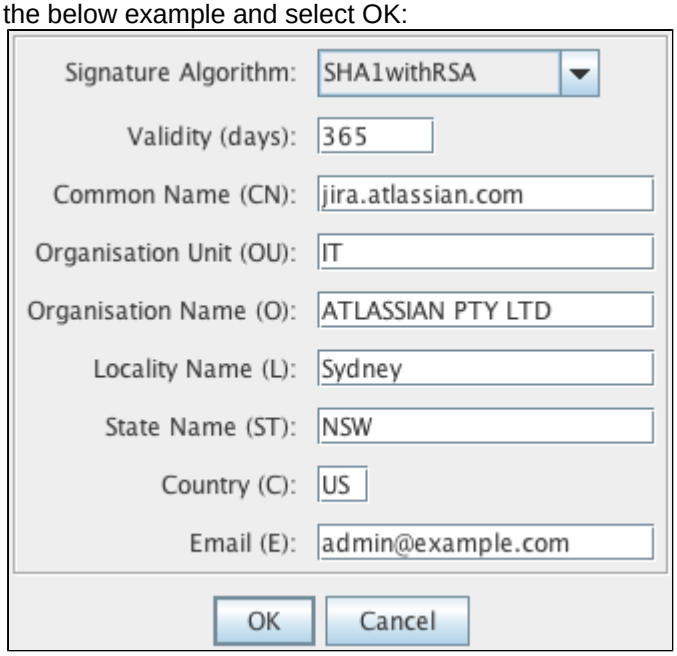
5. Select the **Generate Key Pair** button:



6. Select the RSA algorithm and a Key Size of 2048:

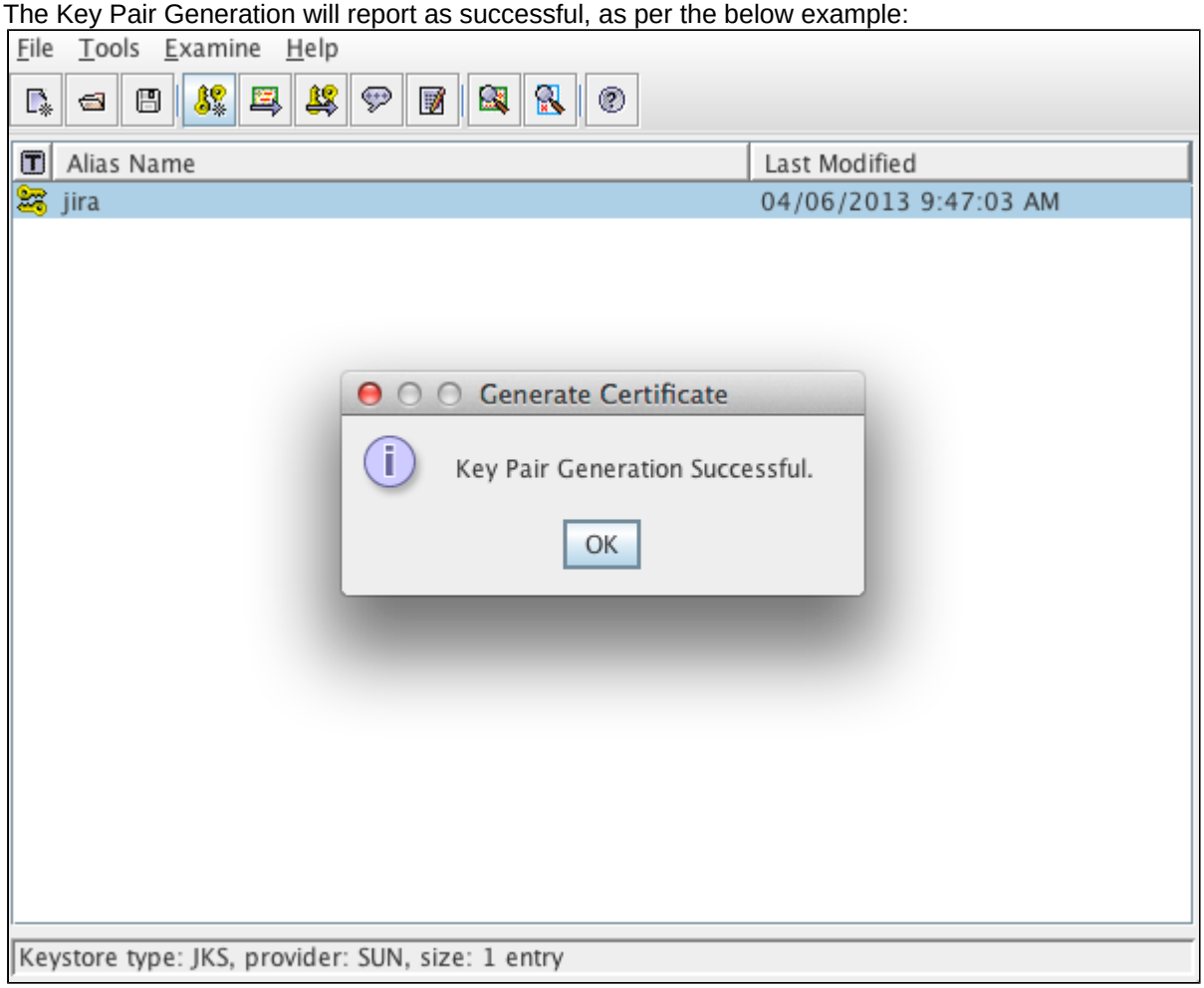


7. Make sure the **Signature Algorithm** is `"SHA1withRSA"` and then edit the certificate details, as per the below example and select OK:



⚠ 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.

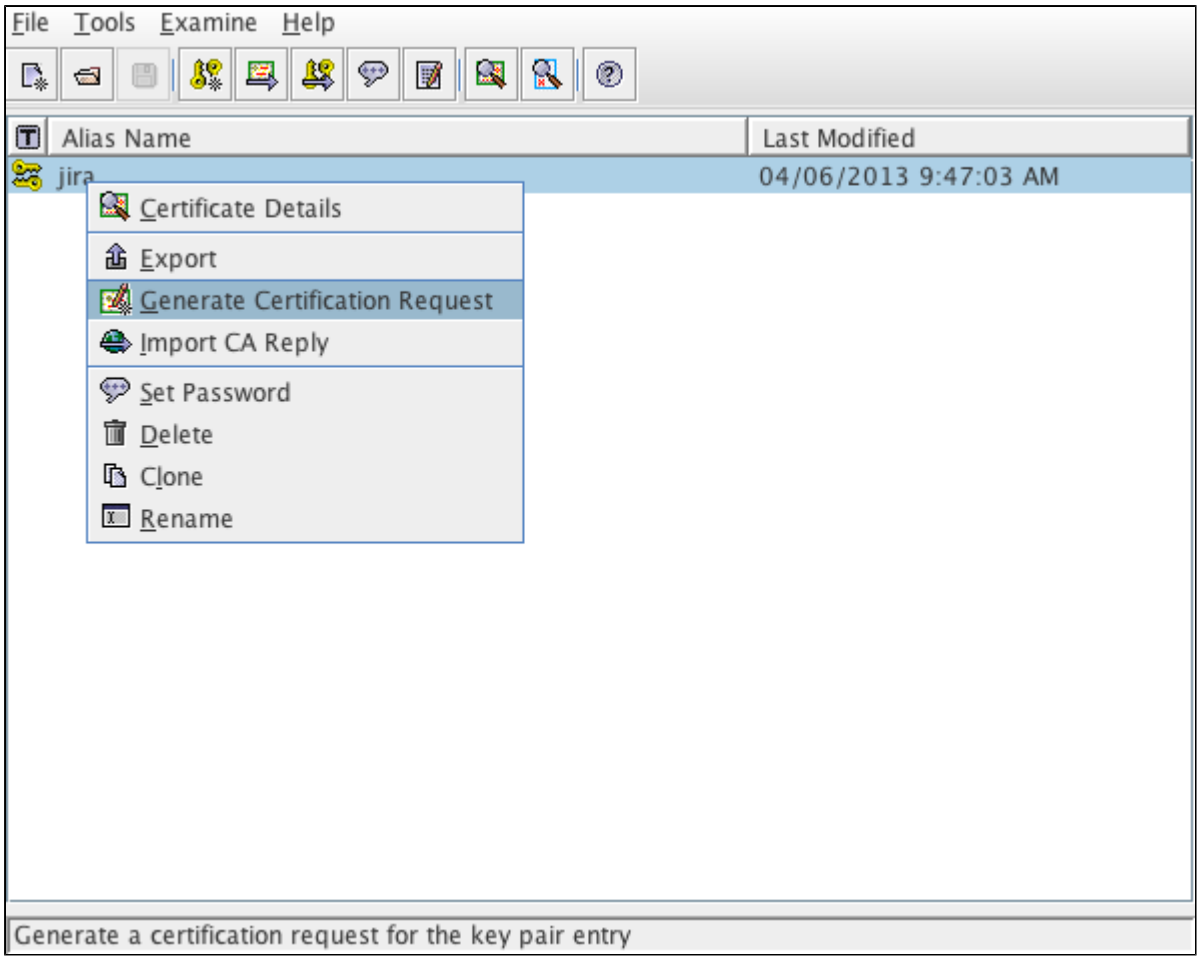
8. Choose an alias for the certificate - for example jira.
9. Enter a password for the KeyStore (the default password used is typically `changeit`).
10. The Key Pair Generation will report as successful, as per the below example:



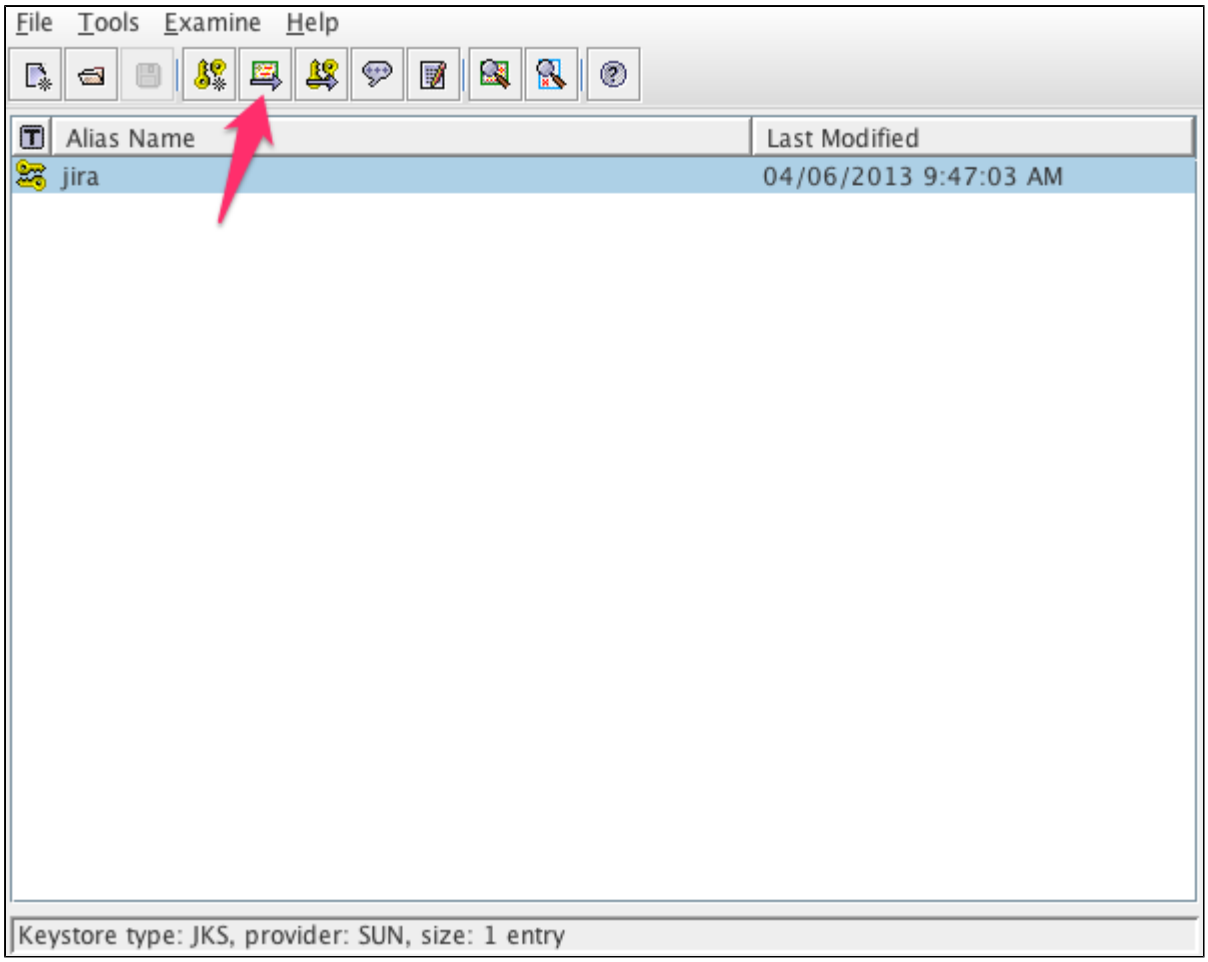
11. Save the KeyStore in `<JIRA_HOME>/jira.jks`, ensuring the use the same password in step 11. This can be done by **File > Save Keystore**.

> If using a self-signed certificate certificate, proceed to Configuring your web server using the JIRA configuration tool. Otherwise, continue on.

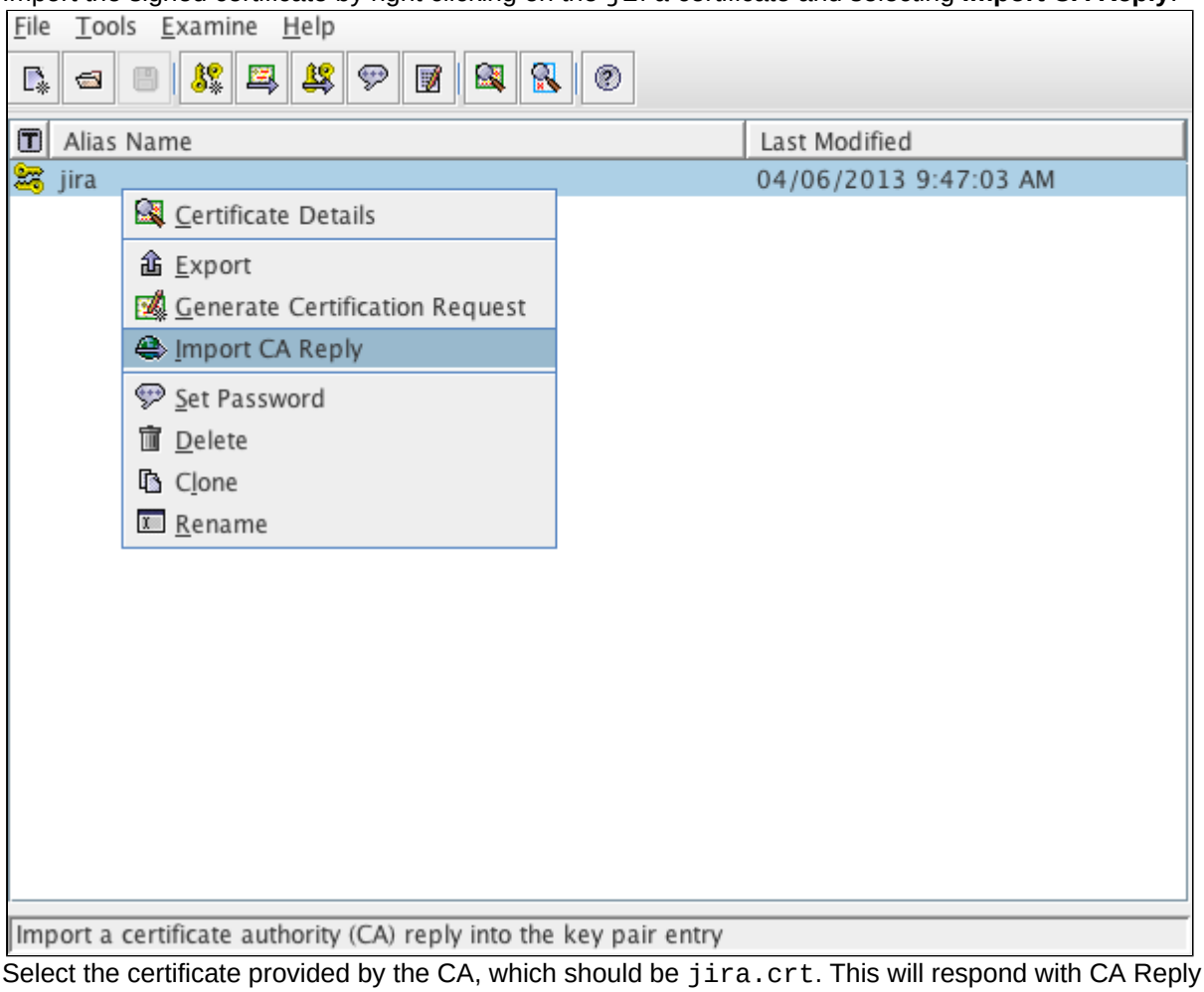
12. We need to generate a Certificate Signing Request for the CA to sign and confirm the identity of the certificate. To do so, right click on the certificate and choose **Generate CSR**. Save it in <JIRA_HOME >/jira.csr.



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

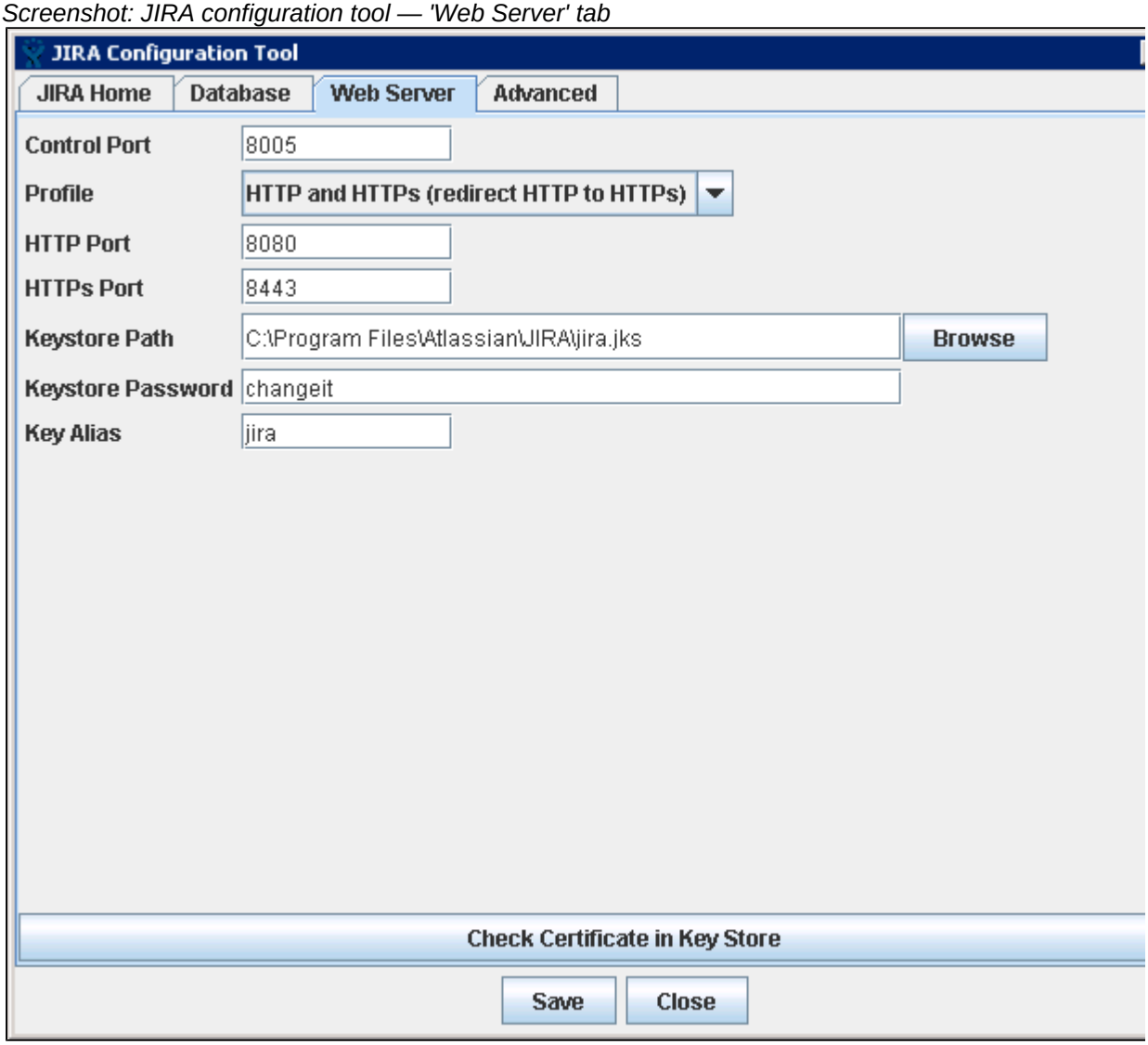
Import successful.

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



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	A profile is a preset web server configuration. You can pick from the four following values: • Disabled • HTTP only • HTTP & HTTPS (redirect HTTP to HTTPS) • HTTPS only 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.
HTTP port	Leave as default (8080). You can change the port number if you wish. See Changing JIRA's TCP ports. This will be disabled if you set the **Profile** to 'HTTPS only'.
HTTPS port	Leave as default (8443). You can change the port number if you wish. See Changing JIRA's TCP ports.
Keystore path	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 `<JIRA_HOME>/jira.jks`.
Keystore password	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.
Keystore alias	Each entry in the keystore is identified by an alias. We recommend using `jira` for this certificate as in step 10.

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 7 docs.

Command Line Installation

Create the Keystore

1. Generate the Java KeyStore (JKS):

```
<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 an appropriate password (e.g.: `changeit`).
3. Create the CSR for signing, using the password from step 2:

```
<JAVA_HOME>/keytool -certreq -keyalg RSA -alias jira -keystore
<JIRA_HOME>/jira.jks -file jira.csr
```

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 (this is provided by the CA):

```
<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
```

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 key store:

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11Protocol"
               maxHttpHeaderSize="8192" SSLEnabled="true"
               maxThreads="150" minSpareThreads="25"
               enableLookups="false" disableUploadTimeout="true"
               acceptCount="100" scheme="https" secure="true"
               clientAuth="false" sslProtocol="TLS"
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.
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"/>
```

   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>*.jspa</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. However, if you want to only redirect certain pages to HTTPS, you can do this manually. To do this, select the 'HTTPS only' profile in the JIRA configuration tool and save the configuration.

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 Http11Protocol to handle SSL connections — Edit the server.xml so that the SSL Connector tag you just uncommented specifies the Http11Protocol instead of the APR protocol

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11Protocol"
  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"/>
```

- **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 7 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

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:

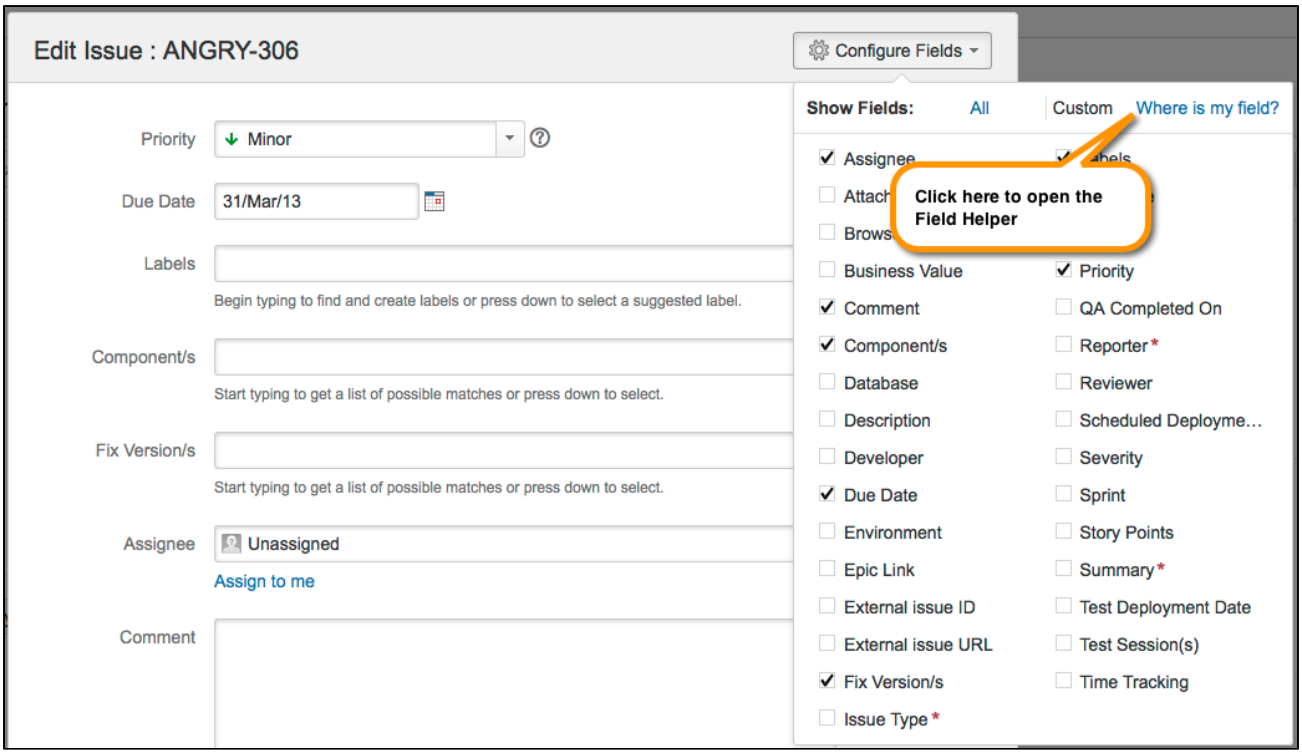
Field Helper

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:

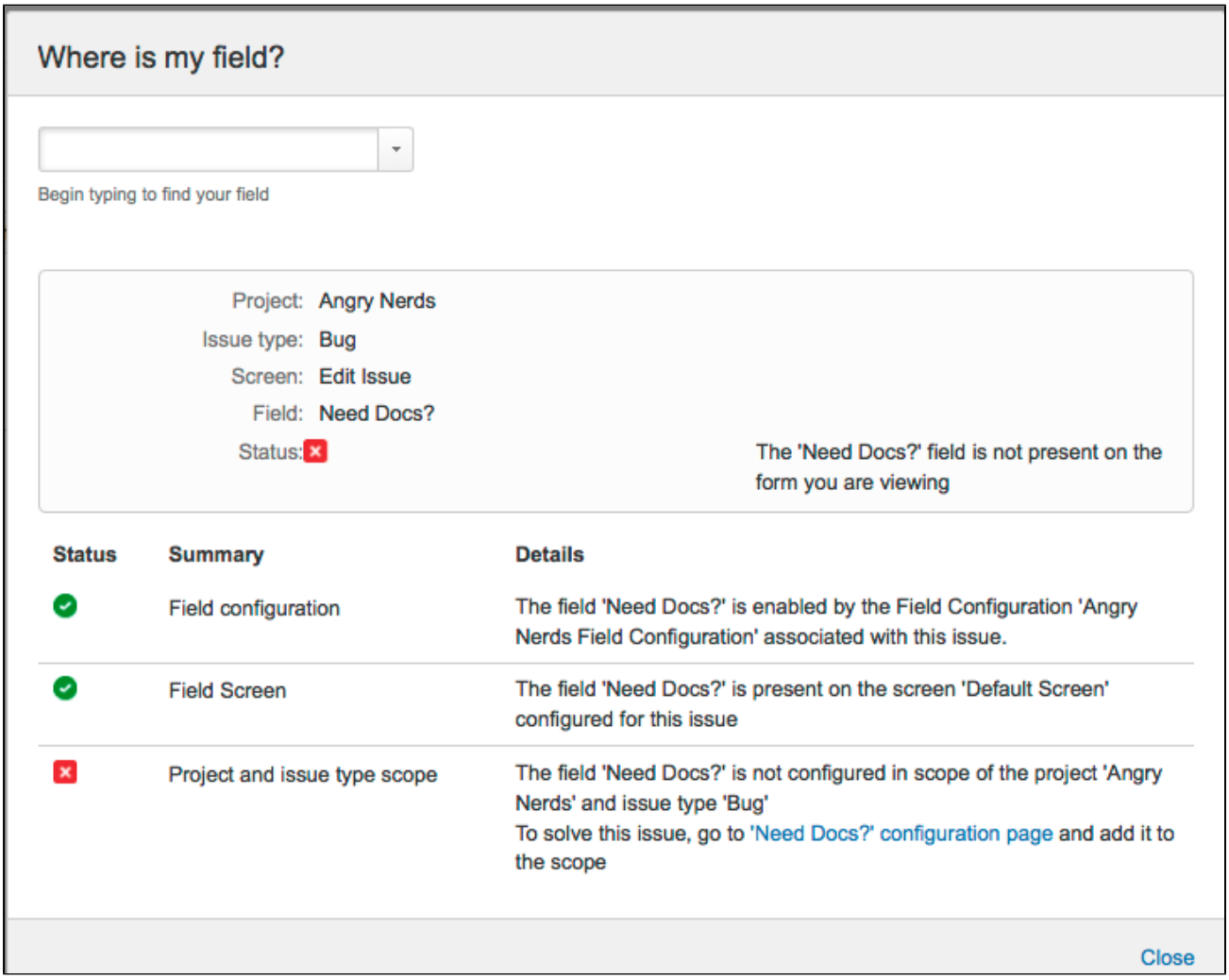
- Create Issue – in *Configure Fields* pop up
- Edit issue - in *Configure Fields* pop up
- View Issue- in *More Actions* menu
- Issue Navigator – in cog menu

Simply click on the link and then enter the field name in the search box!

Here's an example:



After you enter the name of the missing field, the Field Helper returns a form that explains why this field is not appearing:

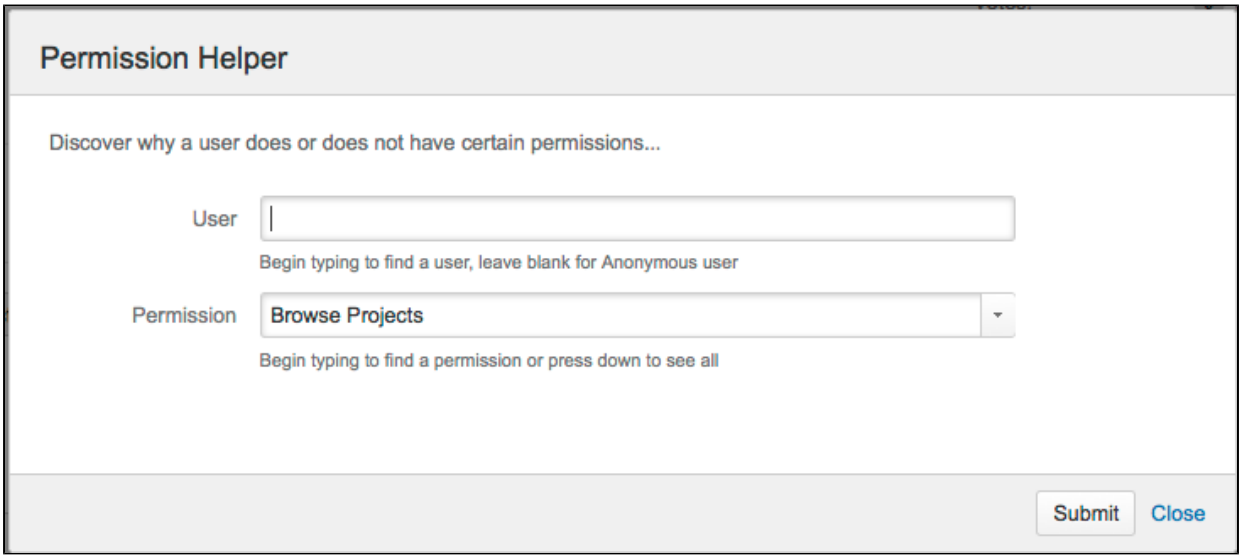


You can then use this information to fix your screen by adding this field to your project and issue type.

Permission Helper

The JIRA Admin Helper can help you diagnose why a user can or cannot see a certain issue.

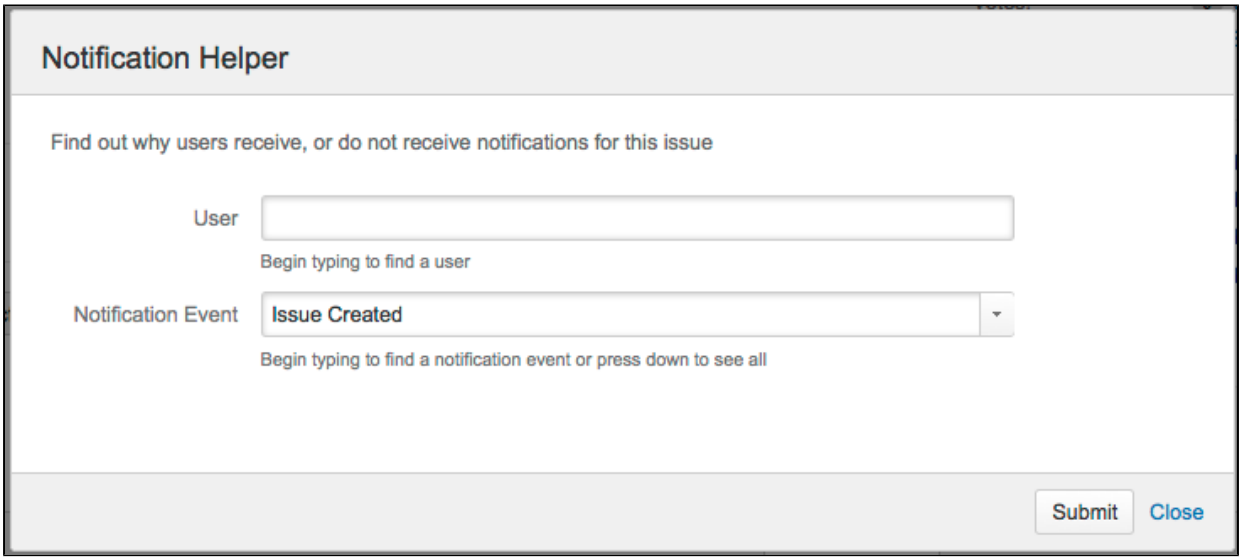
1. Choose
   > System.
2. Then choose **Admin Helper > Permission Helper**.
3. Enter the username of the user (leave blank for anonymous users), an issue key (for example, an issue that the user can/cannot see) and the permission to check.
4. Click **Submit**.



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 > Permission 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**.



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

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.
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.

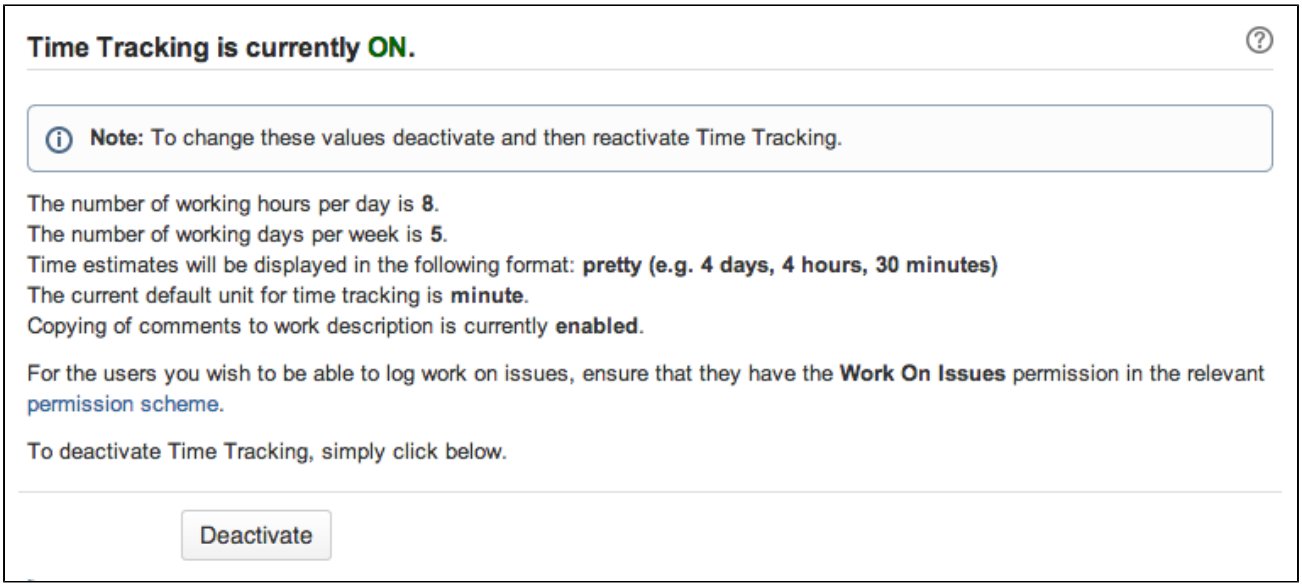
You will not lose any existing time tracking data by disabling/re-enabling time tracking.

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.

Screenshot 1: Time tracking is ON

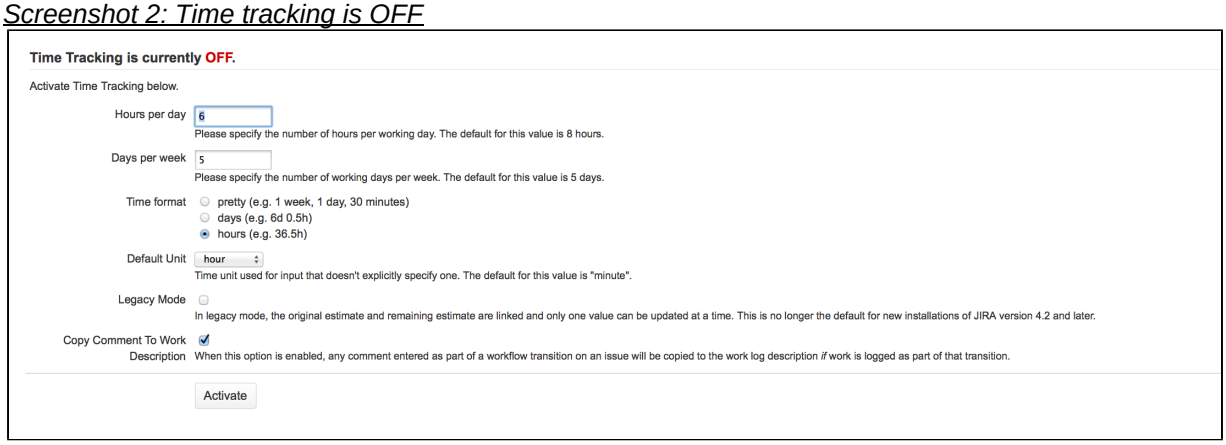


Configuring time tracking settings

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.

You will not lose any existing time tracking data by disabling/re-enabling time tracking.

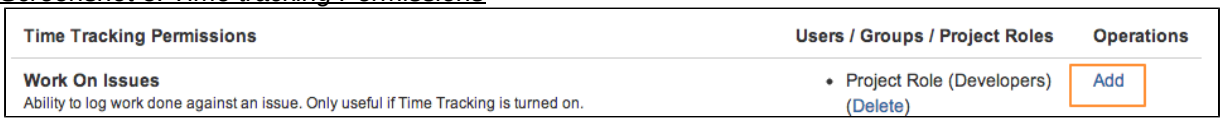
1. Choose  **> Issues.**
2. Select **Issue Features > Time Tracking** to open the Time Tracking page.
3. If Time Tracking is ON (refer to the indication at the top of the Time Tracking screen), click the **Deacti vate** button to turn time tracking OFF.
4. The time tracking settings will now be editable as shown in the following screenshot.
   Screenshot 2: Time tracking is OFF



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.
  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.
   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



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

JRA-25031 - Time Tracking Legacy Mode and Workflow Post Functions Error
VERIFIED

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

1. Choose  **> System**.
2. Select **General Configuration** to open the Administration page.
3. Scroll to the end of the page and click the **Edit Configuration** button to edit the three sections as described below:
   - Settings
   - Internationalization
   - Options

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: - **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. 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. *Default: **Public***
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). *Default: **3** (for new installations of JIRA)*
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. *Default: **ON***

Base URL*	The base URL of this JIRA installation. 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. 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. The address is determined by the mail server or individual project configuration.
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): • 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 *format*, 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

(If marked with an * the function is not available in the Cloud)

Setting	Description

Allow users to vote on issues	Controls whether voting is enabled in JIRA. Voting allows users to indicate a preference for issues they would like to be completed or resolved. See also the 'View Voters and Watchers' permission. *Default:* ***ON***
Allow users to watch issues	Controls whether watching is enabled in JIRA. Users can 'watch' issues which they are interested in. Users watching an issue will be notified of all changes to it. See also the 'View Voters and Watchers' and 'Manage Watcher List' permissions. *Default:* ***ON***
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. *Default:* ***80***
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. *Default:* ***10***
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. *Default:* ***ON***
External user management *	When turned **ON**, you will no longer be able to create, edit or delete users/groups from within JIRA (or via email or import); but you can still assign users/groups to project roles, and create/edit/delete user properties. Additionally, JIRA will not display options for users to change their password, or edit their profile.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) *Default:* ***OFF***
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** *Default:* ***NEVER***
Use gzip compression	Controls whether to compress the web pages that JIRA sends to the browser. It is recommended that this be turned ON, unless you are using mod_proxy. *Default:* ***OFF***
User email visibility	Controls how users' email addresses are displayed in the user profile page. - **PUBLIC** - email addresses are visible to all. - **HIDDEN** - email addresses are hidden from all users. - **MASKED** - the email address is masked (e.g. 'user@example.com' is displayed as 'user at example dot com'). - **LOGGED IN USERS ONLY** - only users logged in to JIRA can view the email addresses. *Default:* ***PUBLIC***
Comment visibility	Determines what will be contained in the list that is presented to users when specifying comment visibility and worklog visibility. - **Groups & Project Roles** - the list will contain groups and project roles. - **Project Roles only** - the list will only contain project roles. *Default:* ***Project Roles only***
Exclude email header 'Precedence: bulk'	Controls whether to prevent the **Precedence: Bulk** header on JIRA notification emails. This option should only be enabled when notifications go to a mailing list which rejects 'bulk' emails. In normal circumstances, this header prevents auto-replies (and hence potential mail loops). *Default:* ***OFF***

Issue Picker Auto-complete	Provides auto-completion of issue keys in the 'Issue Picker' popup screen. Turn OFF if your users' browsers are incompatible with AJAX. *Default:* ***ON***
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. *Default:* ***ON***
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. *Default:* ***Work around Internet Explorer security hole***
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). Applies only if outgoing email is enabled. Can be used with or without the custom 'Contact Administrators Message' below. Users with the **JIRA Administrators** global permission (not **JIRA System Administrators** - see JRA-27454 for details) will be notified as a result of this feature being used. *Default:* ***OFF***
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. *Default:* ***OFF***
Inline edit	Enables inline editing, i.e. click to edit a field on the screen. *Default:* ***ON***
Auto-update search results	Enables search results to be automatically updated when criteria are modified in a basic search. *Default:* ***ON***
Application recommendations	Enables you to turn on/off recommendations for other Atlassian applications on your JIRA instance. *Default:* ***ON*** Application recommendations are only displayed for JIRA Cloud evaluators (at this time).

Enable Atlassian analytics *	Enables you to turn on/off Atlassian analytics. *Default:* ***OFF***

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.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.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.

4. Click the **Update** button (which will appear in the **Operations** column on the right) to save the new value in the JIRA database.
   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:

1. Choose  **> System**.
2. Choose **General Configuration** in the left-hand panel.
3. Choose **Edit Settings**.
4. Enter the new URL in the **Base URL** text box.
5. 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

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.

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.

Add all parameters in a space-separated list, inside the quotations.

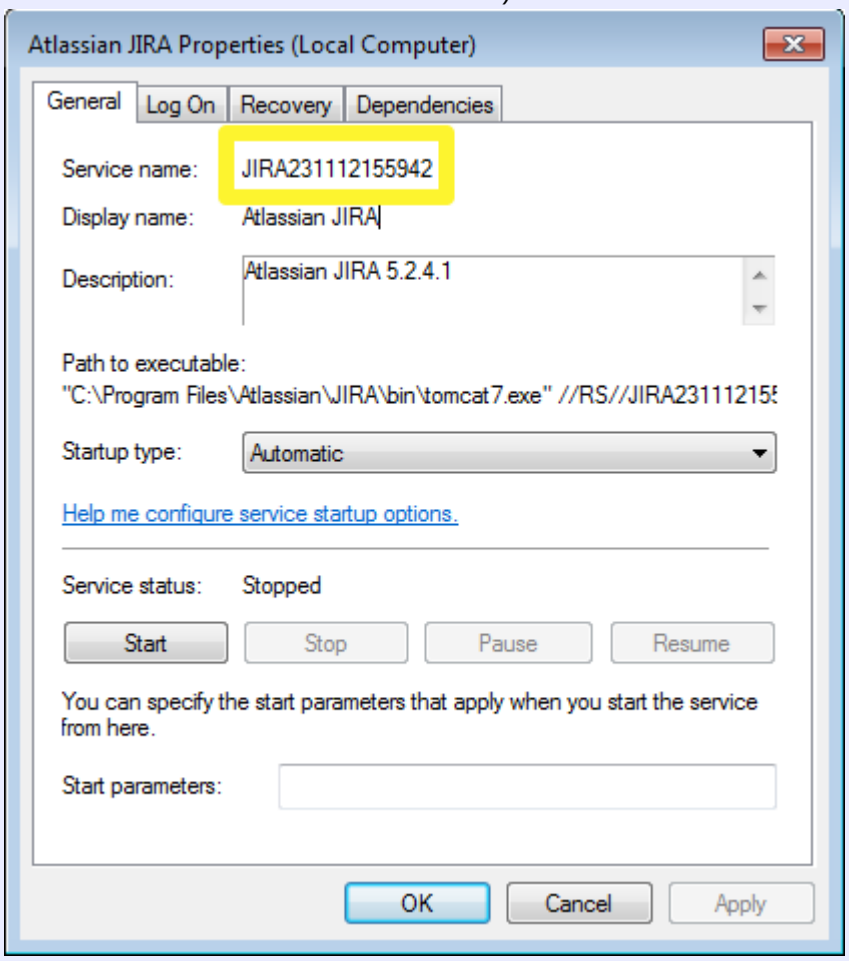
Windows service

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

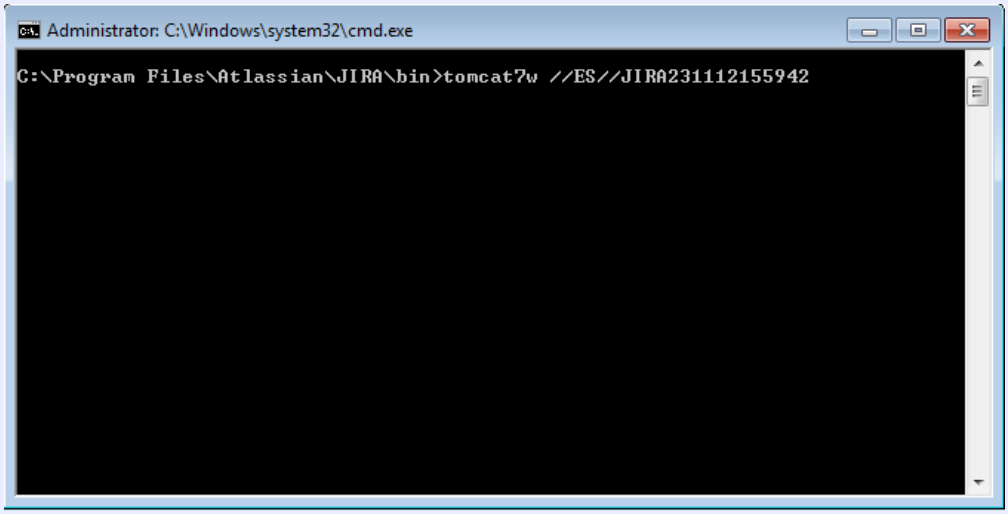
1. Identify the name of the service that JIRA is installed as in Windows (`Control Panel > Administrative Tools > Services`):



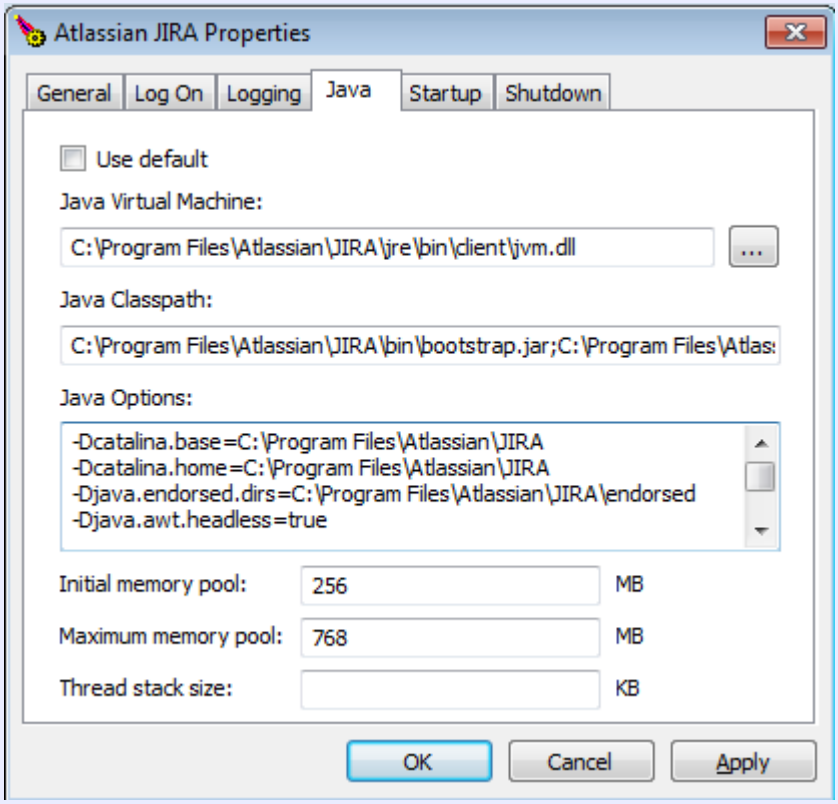
   In the above example, the **SERVICENAME** is: `JIRA231112155942`
2. Open the command window from `Start >> Run >> type in 'cmd' >> Enter`
3. `cd` to the `bin` directory of your JIRA application installation directory.
4. Run:

```
tomcat7w //ES//%SERVICENAME%
```

   In the above example, it would be `tomcat7w //ES//JIRA231112155942`



5. Click on the Java tab to see the list of current start-up options:

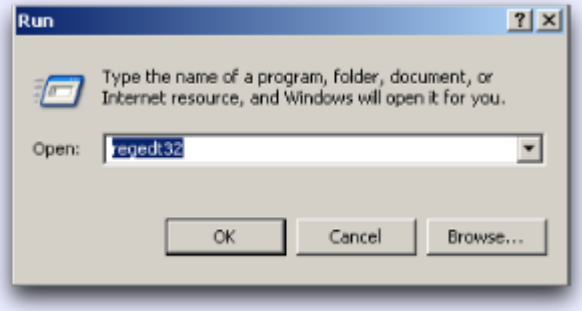


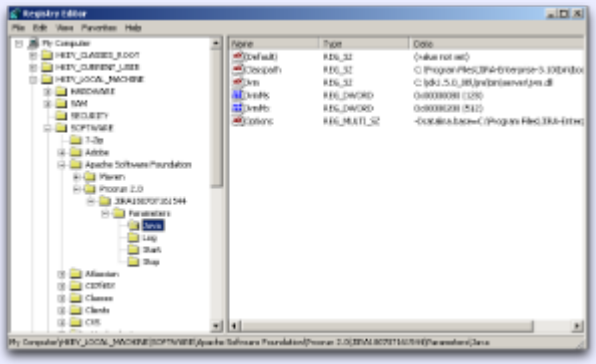
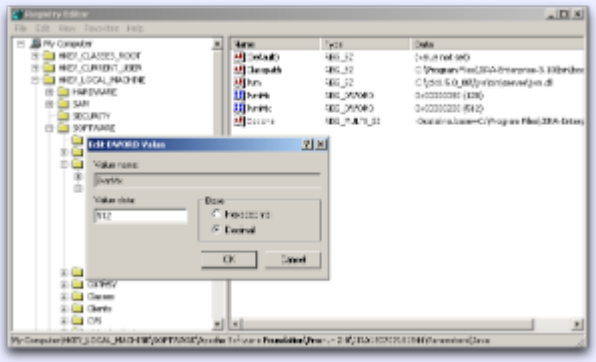
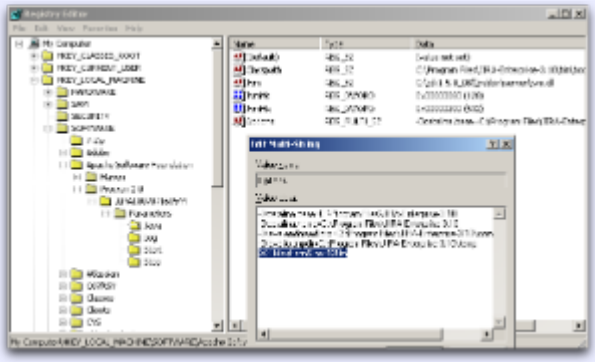
6. Append any new option on its own new line by adding to the end of the existing Java options. Refer to the list of parameters below.

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:+PrintGCTimeStamps -verbose:gc -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError	Set these for Garbage Collection tuning.	Usin to A Perf Usin Anal
-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 Fron
-Dmail.imap.auth.plain.disable -Dmail.imaps.auth.plain.disable		Auth conn
-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		Prob JIRA addr
Encoding property	**Notes**	**Rela**
-Dfile.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 actio Auth
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER	true	Out Men JRA
-ea/-da	Enable/Disable assertions	java Sen
-Djava.net.preferIPv4Stack		Sock 'Inva 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 HTT JIRA
-Dorg.apache.catalina.SESSION_COOKIE_NAME		Logg appl Con

-Datlassian.plugins.enable.wait	Time JIRA waits for plugins to load.	JIRA Whil Enal

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**	**Relat**
-Xmx -Xms XX:MaxPermSize	These properties are pre-existing. See related pages for instructions.	Increa
-XX:+PrintGCTimeStamps -verbose:gc -Xloggc:gc.log -XX:+HeapDumpOnOutOfMemoryError	Set these for Garbage Collection tuning.	Using to Ana Perfor Using Analy:
-agentlib:yjpagent=onexit=memory,dir=/path/to/write/snapshots		Profili Usage
Mail property	**Notes**	**Relat**
-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.	Migrat server Notific Incorr
-Dmail.debug	If set to "true", logs statements related to mail	Config server Creati from e
-Dmail.mime.decodetext.strict		Unabl or Bod From
-Dmail.imap.auth.plain.disable -Dmail.imaps.auth.plain.disable		Authe conne
-Dmail.imap.starttls.enable		'javax No log due to
-Dmail.mime.decodeparameters	Sets mail handler to work correctly with emails from RFC 2231-compliant mail clients.	
-Dmail.smtp.localhost		Proble JIRA - addre
Encoding property	**Notes**	**Relat**

-Dfile.encoding	Set to utf-8 for encoding consistency	Chara ASCII Quest Intern Encoc SQL E issues applic encod Intern Notific Are B Quest
Other Properties	**Notes**	**Relat**
-Duser.timezone		Incorr JIRA
-Dsvnkit.http.methods	Values include Basic,Digest,Negotiate,NTLM	JIRA 'java.l Unabl config Subve unkno action Authe
-Dorg.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER	true	OutOf Memo JRA-1
-ea/-da	Enable/Disable assertions	java.la Sendi
-Djava.net.preferIPv4Stack		Socke 'Invali Availa
-Djavax.net.ssl.trustStore		Unabl 'javax Due t
-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 t HTTP JIRA a
-Dorg.apache.catalina.SESSION_COOKIE_NAME		Loggi applic Conflu
-Datlassian.plugins.enable.wait	Time JIRA waits for plugins to load.	JIRA While Enabl

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
```

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 recognises 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

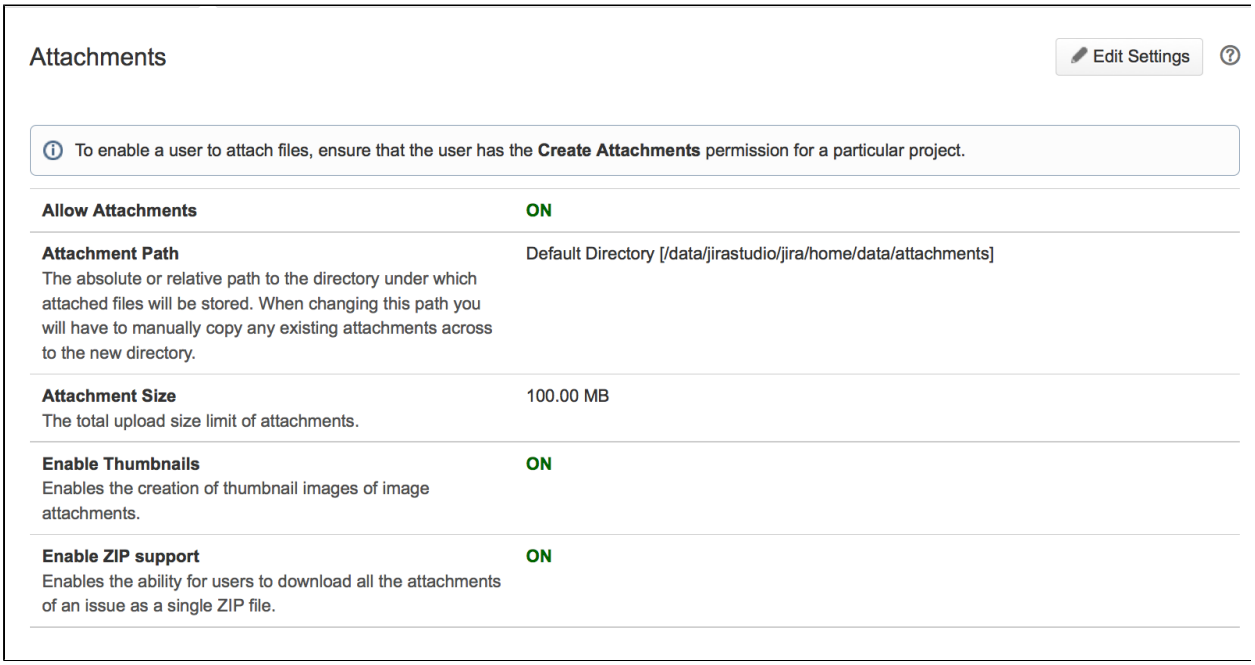
When file attachments are enabled, your users will be allowed to attach files and screenshots to JIRA issues. This requires space on the server for storing the attachments.

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:**

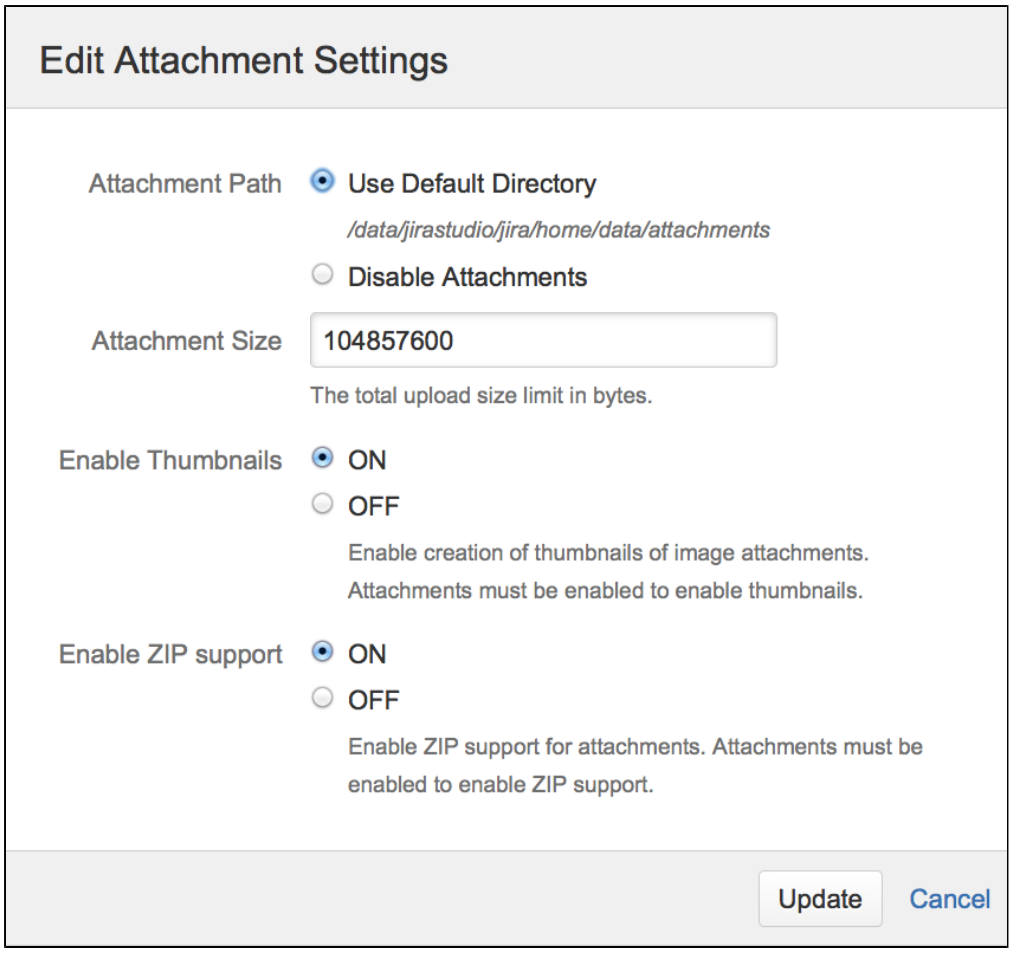
- Your users must also have the **Create Attachments** permissions to attach files to issues.
- To allow users to attach a file *when creating a new issue*, you need to ensure that the **Attachment** field is *not hidden* within the field configuration(s) associated with the specific issue type(s).
- 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.



3. Click the **Edit Settings** button, which opens the **Edit Attachment Settings** dialog box:



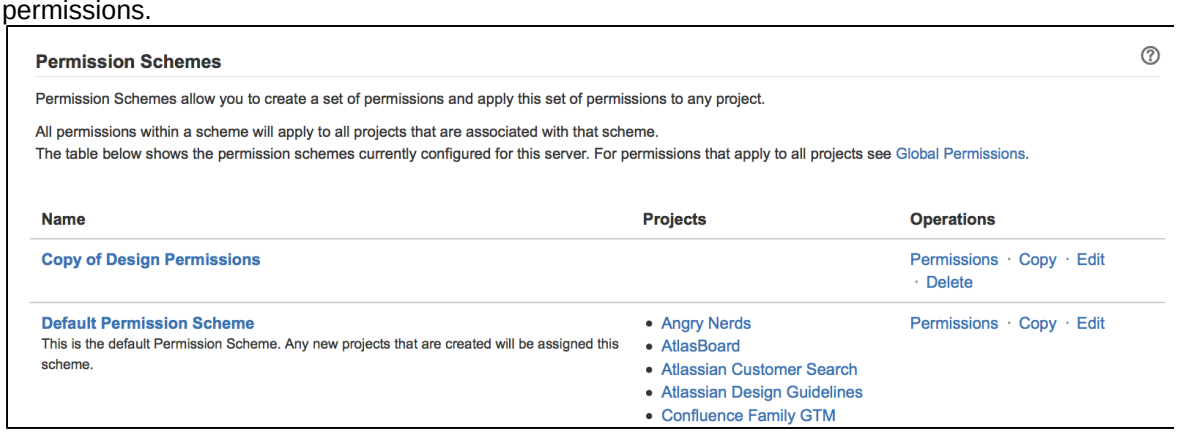
4. In the **Attachment Path** field, choose the **Use Default Directory** option. If you see more attachment path options than what is shown in the screenshot above, please refer to the note below.
   As mentioned above, if you have not logged in as a user with the **JIRA System Administrators** global permission, then this option will not be available to you.
5. In the **Attachment Size** field, specify the maximum attachment size. The default is 10485760 bytes (10 MB). The maximum attachment size 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

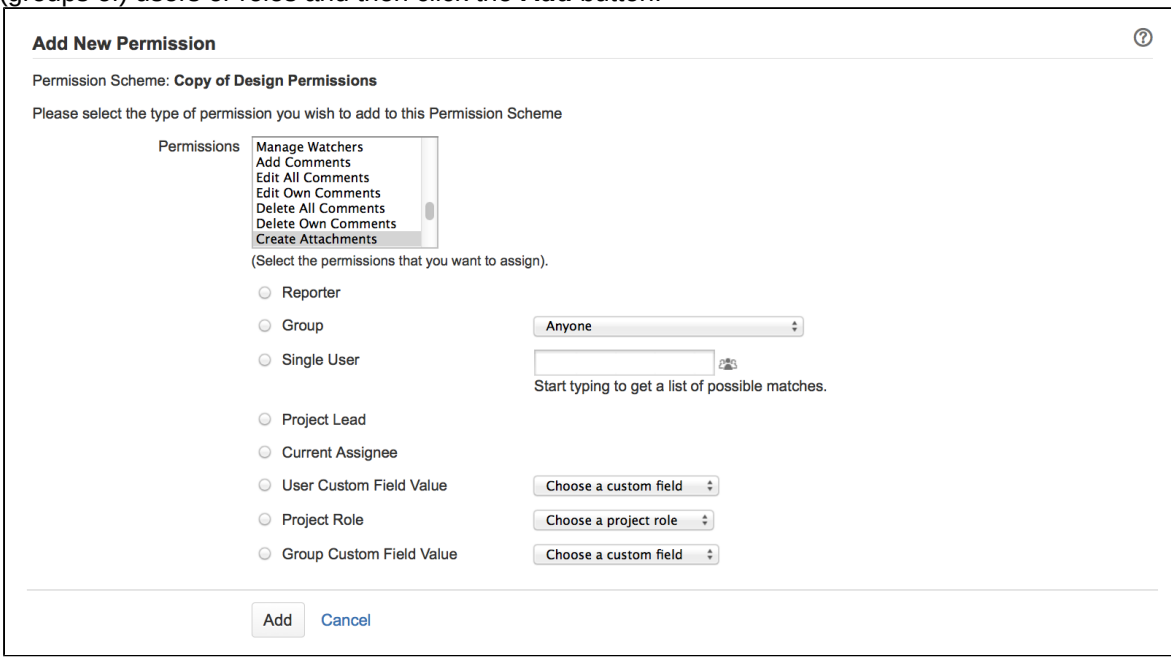
1. Choose
   

> 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.



    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.



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.

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.

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.

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.
   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

   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.
   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 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 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.

Note:

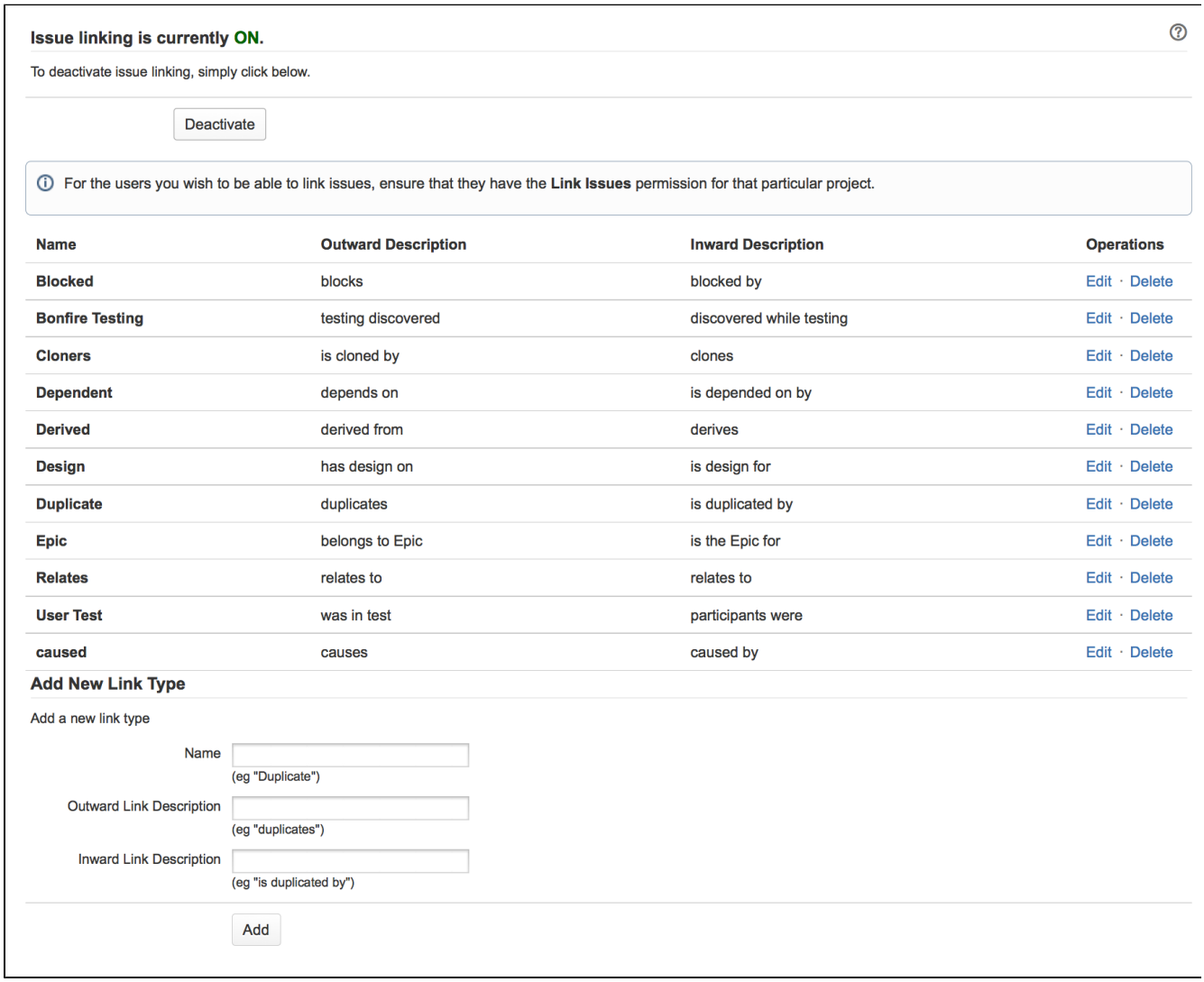
- Your users must have the Link Issues permission before they can link issues.
- Issue linking must be enabled in order for your users to be able to link issues. **Issue linking is enabled by default.** If your organization does not require the ability to link issues, you can disable it globally for 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



Editing or deleting a link type

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 t

o 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 instanc e, 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 o r 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 **Deactiva te** 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 administratorscan 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` pro perty on JIRA's Advanced settingspage.

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 the whitelist

JIRA administrators can choose to allow incoming and outgoing connections and content from specified sources by adding URLs to the whitelist.

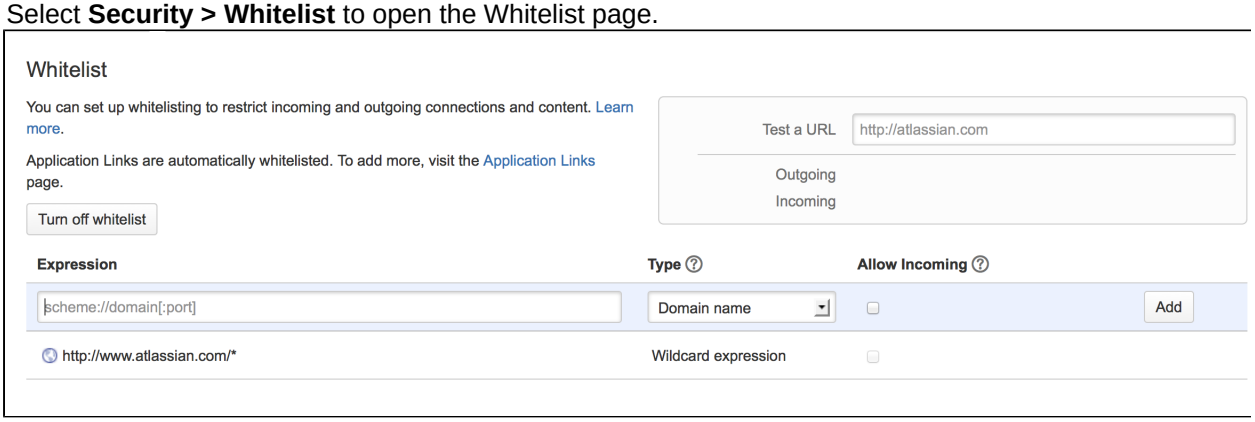
JIRA will display an error if content has been added that is not from an allowed source, and prompt the user to add the URL to the whitelist.

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.



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.	`http://www.example.com`
Exact match	Allows only the specified URL.	`http://www.example.com/thispage`
Wildcard Expression	Allows all matching URLs. Use the wildcard * character to replace one or more characters.	`http://*example.com`
Regular Expression	Allows all URLs matching the regular expression.	`http(s)?://www\.example\.com`

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

Sub-tasks are enabled by default. However, this feature can be disabled from the Sub-Tasks administration page.
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.

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.
A default sub-task issue type is automatically available for use. You can edit it by clicking its **Edit** li nk in the Operations column.

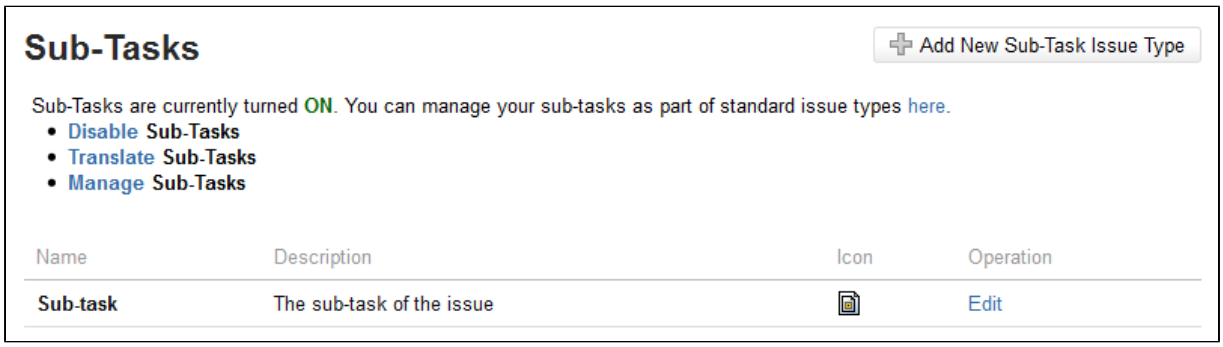
Defining sub-task issue types

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.



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 URLor 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 administratorscan 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).

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.

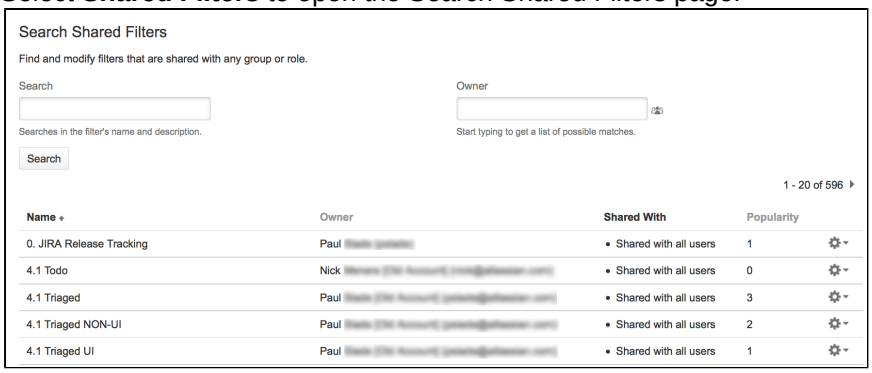
Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** and **JIRA Users** global permissions.

On this page:

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.



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.

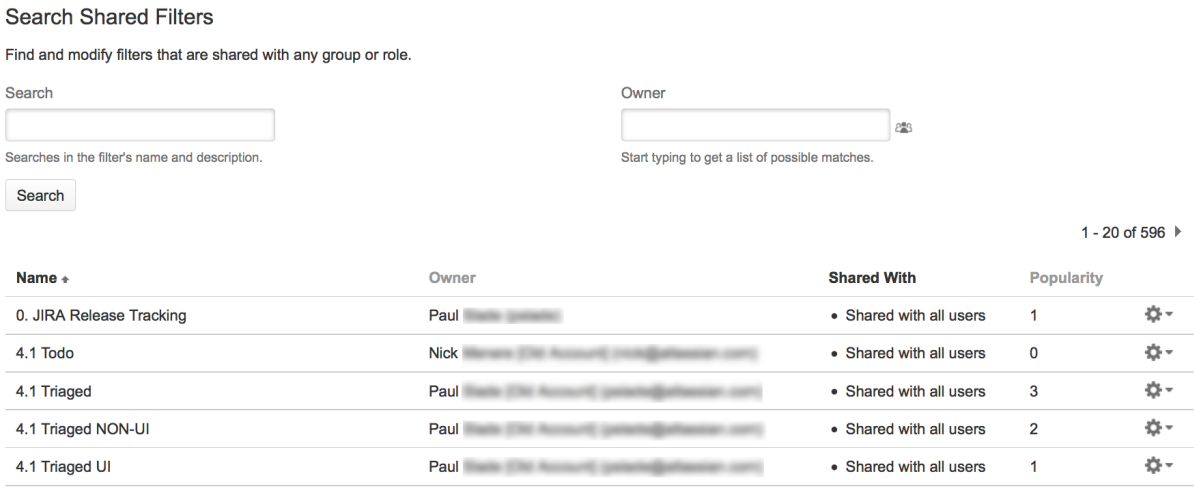
Please note:

- 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.

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.



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.

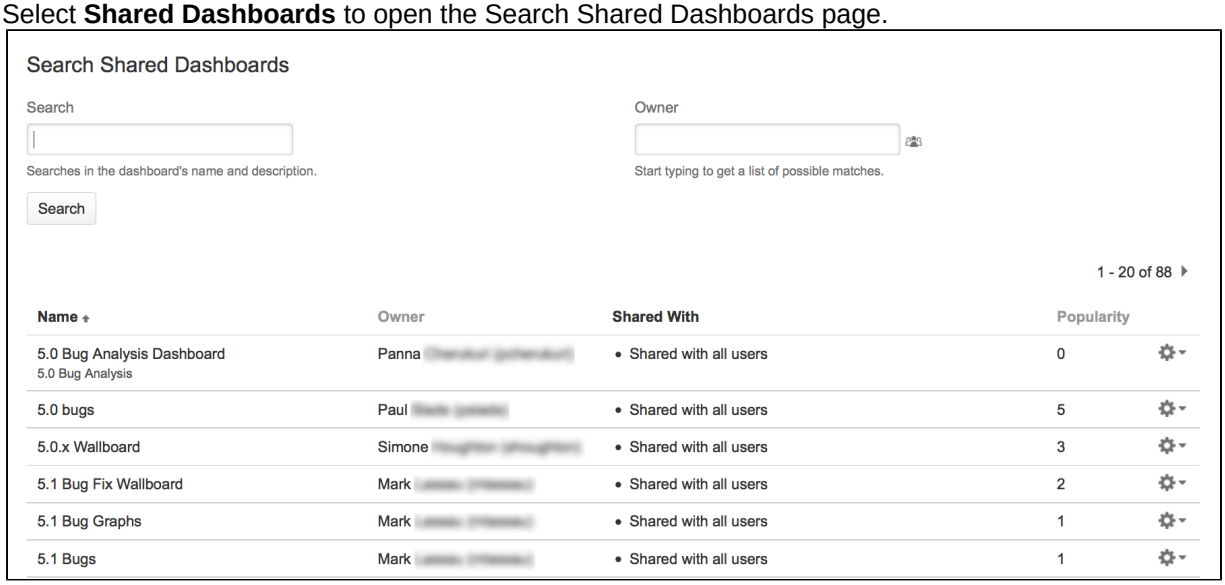
Note: For all of the following procedures, you must be logged in as a user with the **JIRA Administrators** global permission.

On this page:

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  **> System**.
2. Select **Shared Dashboards** to open the Search Shared Dashboards page.



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.

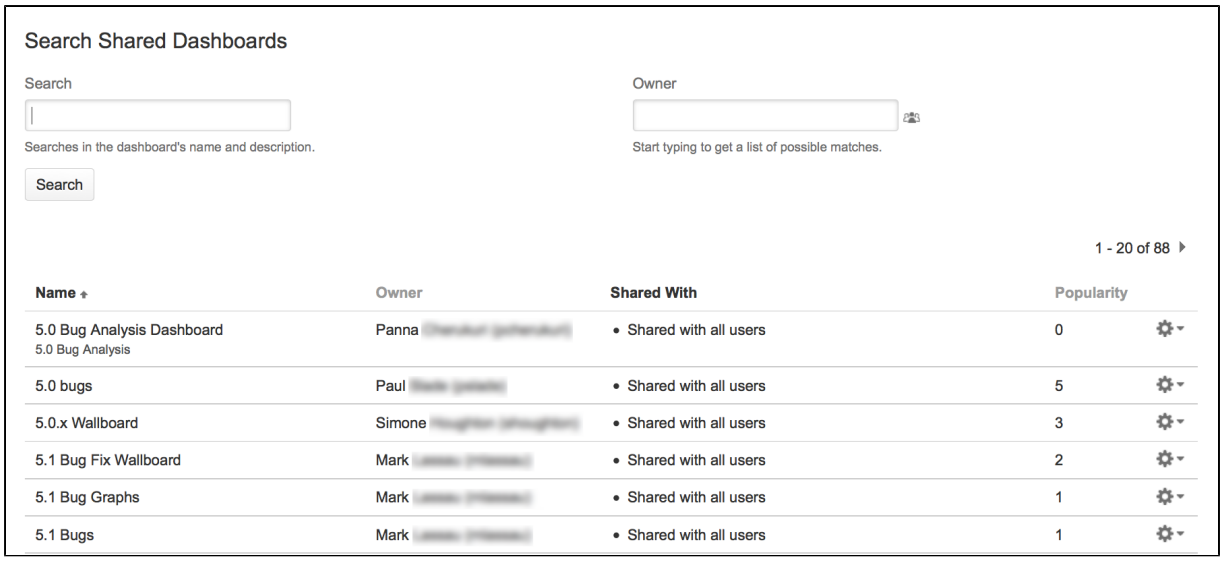
Please note:

- 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.



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 marked that shared dashboard as a 'favorite'.
4. 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.
5. 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:

1. Choose  **> System**.
2. Select **General configuration** to open the Administration page.
3. 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).

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
- Performance testing scripts
- JIRA application cookies
- Preventing security attacks
- Using the JIRA application configuration tool
- Running JIRA applications as a Windows service

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

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
```

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
```

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.

Performance testing scripts

> Please be aware that the content on this page is not actively maintained and Atlassian **can not guarantee providing any support for it**. Furthermore, the performance testing scripts which you can download from Atlassian's public Maven repository (via the link on this page) are no longer

supported.

This page is provided for your information only and using it is done so at your own risk. Instead of using these scripts, we would recommend our JIRA Performance Testing with Grinder page.

This page contains scripts and hints for testing usage load on your JIRA installation.

When setting up a new JIRA installation, it is useful to understand how it will perform under your anticipated load before users begin accessing it. Scripts that generate 'request' (or usage) load are provided in our public Maven repository (link below). Using these scripts, you can find out where you may need to consider improving your configuration to remove bottlenecks.

While this kind of testing is not an exact science, the tools and processes described here are intended to be straightforward and configurable, and provide you with an extensible way to assess load testing.

The performance tests described on this page utilise JMeter. While it is not necessary to know JMeter, briefly reading through the JMeter documentation is recommended as it may help you resolve any JMeter-specific issues.

It is rarely the case that these scripts will perform representative testing for you 'out of the box'. However, it should be possible to build an appropriate load test by configuring or extending these scripts.

On this page:

Load testing scripts should not be used on a production JIRA installation!
While we recommend using a copy of your production data for testing usage load, the load testing scripts below *will modify data within the targeted JIRA installation!* Hence, these scripts *should not be used on a production JIRA installation*. Instead, use a copy of your production JIRA data on a *test* JIRA installation.

If you do run these test scripts against your production JIRA installation, you will be responsible for any data loss and backup recovery!

Likewise, when making changes to your JIRA installation to remove performance bottlenecks, it is useful to assess the impact of these changes in a test JIRA installation before implementing them in production.

Prerequisites

You will need the following:

- A JIRA installation, set up and running with an administrator user. The scripts assume that the username/password combination of this user is 'admin'/'admin'.
- It is recommended that you test with a production quality database, such as one listed on the Supported platforms page. Do not use H2, the evaluation database which is built into your JIRA installation.
- Apache JMeter (currently version 2.3.4). If you intend to do high load testing, please use our modified version of JMeter instead (which requires Java 1.6).
- The load testing scripts and resources which are available in our public Maven repository — Please choose the version that most closely matches your JIRA version and download the ZIP or Gzip file in that directory. If in doubt, download the ZIP file archive.

Users have reported problems using the Windows built-in unzip utility to extract these archives. If you encounter such a problem, please use a third party file archiving and extraction program (for example, 7-Zip) to extract these performance tests.

Quick, just tell me how to run these tests!

If you do not want to read the rest of this document, here are the main points:

1. Create the **setup test** data:

```
<jmeter location>/bin/jmeter -n -t jmeter-test-setup.jmx
-Jadmin.user=<username> -Jadmin.pass=<password>
```

2. Run the **fixed load test**:

```
<jmeter location>/bin/jmeter -n -t jmeter-test-fixedload.jmx
```

The remainder of this document is just an elaboration of those two steps.

> For information on how to use JMeter, please refer to the JMeter documentation.

Performance tests

JIRA performance tests are made up of two parts:

- Setup test — runs first and prepares the JIRA installation for a subsequent **fixed load test**
- Fixed load test — simulates a number of users accessing the JIRA installation.

Setup test

The **setup test** is responsible for:

- Creating projects
- Creating users
- Creating and commenting on (and optionally resolving) issues.

Running the setup test:

After extracting the performance test zip file, change into the `performanceTest` directory. From this directory, run the performance setup test:

```
<jmeter location>/bin/jmeter -n -t jmeter-test-setup.jmx -Jadmin.user=<username>
-Jadmin.pass=<password>
```

where `<jmeter.location>` is the base directory of JMeter

If you omit the `-n` switch, JMeter will run as a GUI. You may then start the test from within the GUI.

As seen above with the `admin.user` and `admin.pass` parameters, JMeter supports `-Jparameter=value` command arguments in order to control execution. The following parameters control the setup test execution:

Configuration control

Parameter	Default	Explanation
jira.host	localhost	The hostname or address of the JIRA installation.
jira.port	8000	The network port that the JIRA installation is running on.
jira.context	/	JIRA webapp context.
admin.user	admin	Administrator username.
admin.pass	admin	Administrator password.

script.base	.	The location of the performance tests. This should only be set if you run the tests from outside the `scripts` directory.
remove.data	false	Running the script with this enabled will remove the users and projects created by the test.

User control

Parameter	Default	Explanation
create.users.enable	true	Create users in the target JIRA installation. Use **false** if you already have the users created elsewhere.
browseissue.max	250	The number of users to be created for browsing the JIRA installation (aka "browseissue" users).
createissue.max	250	The number of users to be created for creating issues (aka "createissue" users).
editissue.max	250	The number of users to be created for editing issues (aka "editissue" users).
search.max	250	The number of users to be created for searching issues (aka "search" users).
useraction.max	250	The number of users to be created for browsing user information (aka "useraction" users).
browseissue.groupname	none	The group to which "browseissue" users will be placed. Use **none** for no group.
createissue.groupname	jira-developers	The group to which "createissue" users will be placed. Use **none** for no group.
editissue.groupname	jira-developers	The group to which "editissue" users will be placed. Use **none** for no group.
search.groupname	none	The group to which "search" users will be placed. Use **none** for no group.
useraction.groupname	jira-developers	The group to which "useraction" users will be placed. Use **none** for no group.

Project control

Parameter	Default	Explanation
create.projects.enable	true	Create projects. Use **false** if you want to use existing projects (in existing data).
project.max	20	The number of projects to create in the system.

Issue control

Parameter	Default	Explanation
create.issues.enable	true	Creates issues in the target JIRA installation. Use **false** if you do not want the test to create sample issues.
issue.max	3000	The number of issues to be created.
issue.comment.enable	true	Controls whether or not comments are added to issues.

issue.comment.max	10	If `issue.comment.enable` is true, then the number of actual comments created on an issue is chosen randomly between 0 and this value.
issue.close	true	Controls whether or not issues will be closed automatically after being created.
issue.close.percentage	60	If `issue.close` is enabled, then this value defines the percentage of issues closed.
issue.setupload.threads	10	The number of threads used for creating the issues.
issue.setupload.pause	50	The amount of time (in milliseconds) for which a simulated user will 'sleep' between each request during issue creation.
resource.dir	resources	The directory which contains the CSV data resources.

Test output

Once you have chosen your target settings, run JMeter and you should get output similar to the following:

```
jmeter -n -t jmeter-test-setup.jmx
Created the tree successfully using jmeter-test-setup.jmx
Starting the test @ Mon Oct 26 23:53:28 CDT 2009 (1256619208435)
Generate Summary Results +   931 in  31.3s =   29.7/s Avg:    26 Min:
13 Max:  3256 Err:     0 (0.00%)
Generate Summary Results +  2948 in 180.0s =   16.4/s Avg:    31 Min:
8 Max:  1162 Err:     0 (0.00%)
Generate Summary Results =  3879 in 211.4s =   18.3/s Avg:    29 Min:
8 Max:  3256 Err:     0 (0.00%)
Generate Summary Results +  5048 in 179.9s =   28.1/s Avg:    44 Min:
7 Max:   936 Err:     0 (0.00%)
Generate Summary Results =  8927 in 391.4s =   22.8/s Avg:    37 Min:
7 Max:  3256 Err:     0 (0.00%)
Generate Summary Results +  3114 in 180.1s =   17.3/s Avg:    41 Min:
7 Max:   805 Err:     0 (0.00%)
Generate Summary Results = 12041 in 571.3s =   21.1/s Avg:    38 Min:
7 Max:  3256 Err:     0 (0.00%)
Generate Summary Results +  4956 in 179.8s =   27.6/s Avg:    45 Min:
7 Max:  1844 Err:     0 (0.00%)
Generate Summary Results = 16997 in 751.4s =   22.6/s Avg:    40 Min:
7 Max:  3256 Err:     0 (0.00%)
Generate Summary Results +   313 in  17.1s =   18.3/s Avg:    37 Min:
7 Max:   165 Err:     0 (0.00%)
Generate Summary Results = 17310 in 768.5s =   22.5/s Avg:    40 Min:
7 Max:  3256 Err:     0 (0.00%)
Tidying up ...    @ Tue Oct 27 00:06:17 CDT 2009 (1256619977181)
... end of run
```

This output will be updated every 3 minutes, showing the most recent activity as well as a summary for the whole test.

Result logs

In addition to this summary data, which is output on the command line, log files are created for both the successful (`jmeter-results-setup.jtl`) and unsuccessful (`jmeter-assertions-setup.jtl`)

results. These log files are saved in JTL format (which is based on XML). There are a number of parsers which will generate graphs from these log files. For more information, see the JMeter wiki page on Log Analysis.

Fixed load test

Once the setup test has successfully run, the **fixed load test** can be run. This test will simulate a number of users accessing the JIRA installation.

This test has a number of parameters for tweaking the behavior if the test. By default, the test has the following behavior and strategy:

- Several groups of users, all running concurrently for a fixed amount of time, each with a small delay between requests.
  - 'Edit Issue' (editissue) users browse a project and then attempt to find an issue. They will then comment, edit or change the workflow of that issue.
  - 'User Action' (useraction) users create filters, view watches and votes.
  - 'Browse Issue' (browseissue) users browse projects and issues.
  - 'RSS' users browse project and then periodically fetch the RSS feed for that project.
  - 'Create Issues' (createissue) users add new issues to the instance.
  - 'Search Issues' (search) users search for issues using the quick search textbox.

There is **no execution of JavaScript** by the JMeter client. JavaScript performance will depend on several factors such as your browser and workstation speed. JMeter does not measure this.

Running the fixed load test:

```
<jmeter location>/bin/jmeter -n -t jmeter-test-fixedload.jmx
```

As with the setup test (above), this command will run the **fixed load test** with the default values. Similarly, it is possible to control the execution of JMeter with `-J` parameters. The fixed load test has the following available parameters:

Configuration control

Parameter	Default	Explanation
jira.host	localhost	The hostname or address of the JIRA installation.
jira.port	8000	The network port that the JIRA installation is running on.
jira.context	/	JIRA webapp context.
admin.user	admin	Administrator username.
admin.pass	admin	Administrator password.
script.base	.	The location of the performance tests. This should only be set if you run the tests from outside the `scripts` directory.
script.runtime	1800	The amount of time to run for (in seconds).
resource.dir	resources	The subdirectory which contains the resource CSV files. Replace this if you wish to customize the backend data.

Edit issue

Parameter	Default	Explanation
editissue.threads	5	The number of simultaneous 'Edit Issue' users to simulate.
editissue.pause	15000	The pause between each 'Edit Issue' user request (in milliseconds).

workflow.matchname	(Close | Resolve)	A regular expression to match the workflow to action.
editworkflow.percentage	20	The percentage of 'Edit Issue' user requests that will attempt to change the issue workflow.
addcomment.percentage	60	The percentage of 'Edit Issue' user requests that will attempt to add a comment to an issue.
editissue.percentage	20	The percentage of 'Edit Issue' user requests that will attempt to edit an issue.
editissue.issuestoown	5	The number of issues the test attempts to assign to an 'Edit Issue' user.

User actions

Parameter	Default	Explanation
useraction.threads	1	The number of simultaneous 'User Action' users to simulate.
useraction.pause	40000	The pause between each 'User Action' user request (in milliseconds).
createfilter.percentage	10	The percentage of 'User Action' user requests that will attempt to create a filter.
viewwatches.percentage	10	The percentage of 'User Action' user requests that will attempt to 'view watches'.
viewvotes.percentage	10	The percentage of 'User Action' user requests that will attempt to view votes.

Browse issues and projects

Parameter	Default	Explanation
browseissue.threads	40	The number of simultaneous 'Browse Issue' users to simulate.
browseissue.pause	3000	The pause between each 'Browse Issue' user request (in milliseconds).
userprofile.percentage	10	The percentage of 'Browse Issue' user requests that will attempt to browse a user profile.
browsedashboard.percentage	20	The percentage of 'Browse Issue' user requests that will attempt to browse the dashboard.
dashboard.category	allprojects	The project category for project status gadget requests.

RSS

Parameter	Default	Explanation
browserss.threads	10	The number of simultaneous 'RSS' users to simulate.
browserss.pause	60000	The pause between each 'RSS' user request (in milliseconds).

Create issues

Parameter	Default	Explanation

issue.create.threads	3	The number of simultaneous 'Create Issue' users to simulate.
issue.create.pause	15000	The pause between each 'Create Issue' user request (in milliseconds).
issue.comment.max	2	The maximum number of comments on an issue. The actual number is chosen randomly between 0 and this value.

Search for issues

Parameter	Default	Explanation
search.threads	2	The number of simultaneous 'Search' users to simulate.
search.pause	15000	The pause between each 'Search' user request (in milliseconds).

Test output

Once you have chosen your target settings, run JMeter and you should get output similar to the following:

```
jmeter -n -t jmeter-test-fixedload.jmx
Created the tree successfully using jmeter-test-fixedload.jmx
Starting the test @ Wed Oct 28 01:13:22 CDT 2009 (1256710402435)
Waiting for possible shutdown message on port 4445
Generate Summary Results +   568 in  97.9s =     5.8/s Avg:    62
Min:     1 Max:  1534 Err:     0 (0.00%)
Generate Summary Results +  3861 in 179.4s =    21.5/s Avg:    39
Min:     0 Max:   494 Err:     0 (0.00%)
Generate Summary Results =  4429 in 277.4s =    16.0/s Avg:    42
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results +  7356 in 180.0s =    40.9/s Avg:    37
Min:     0 Max:   481 Err:     0 (0.00%)
Generate Summary Results = 11785 in 457.3s =    25.8/s Avg:    39
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 10841 in 180.1s =    60.2/s Avg:    38
Min:     0 Max:   995 Err:     0 (0.00%)
Generate Summary Results = 22626 in 637.4s =    35.5/s Avg:    39
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 11821 in 180.3s =    65.6/s Avg:    37
Min:     0 Max:   507 Err:     0 (0.00%)
Generate Summary Results = 34447 in 817.3s =    42.1/s Avg:    38
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 11904 in 180.9s =    65.8/s Avg:    38
Min:     0 Max:   658 Err:     0 (0.00%)
Generate Summary Results = 46351 in 997.4s =    46.5/s Avg:    38
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 11697 in 180.3s =    64.9/s Avg:    38
Min:     0 Max:   488 Err:     0 (0.00%)
Generate Summary Results = 58048 in 1177.4s=    49.3/s Avg:    38
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 11731 in 180.0s =    65.2/s Avg:    39
Min:     0 Max:   810 Err:     0 (0.00%)
Generate Summary Results = 69779 in 1357.4s=    51.4/s Avg:    38
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 11646 in 180.0s =    64.7/s Avg:    39
Min:     0 Max:   776 Err:     0 (0.00%)
Generate Summary Results = 81425 in 1537.4s=    53.0/s Avg:    38
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results + 11810 in 180.0s =    65.6/s Avg:    39
Min:     0 Max:   798 Err:     0 (0.00%)
Generate Summary Results = 93235 in 1717.3s=    54.3/s Avg:    38
Min:     0 Max:  1534 Err:     0 (0.00%)
Generate Summary Results +  5453 in 109.1s =    50.0/s Avg:    42
Min:     0 Max:   858 Err:     0 (0.00%)
Generate Summary Results = 98688 in 1826.4s=    54.0/s Avg:    39
Min:     0 Max:  1534 Err:     0 (0.00%)
Tidying up ...    @ Wed Oct 28 01:43:49 CDT 2009 (1256712229128)
... end of run
```

This output will be updated every 3 minutes, showing the most recent activity as well as a summary for the whole test.

Result logs

As above, there will be output on the command line and log files will be created for both the successful (`jmet`

er-results-setup.jtl) and unsuccessful (jmeter-assertions-setup.jtl) results. These log files are saved in the JTL format (based on XML). There are a number of parsers which will generate graphs from these logs files. For more information, see the JMeter wiki page on Log Analysis.

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

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.

On this page:

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.

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:

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 recognise 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.

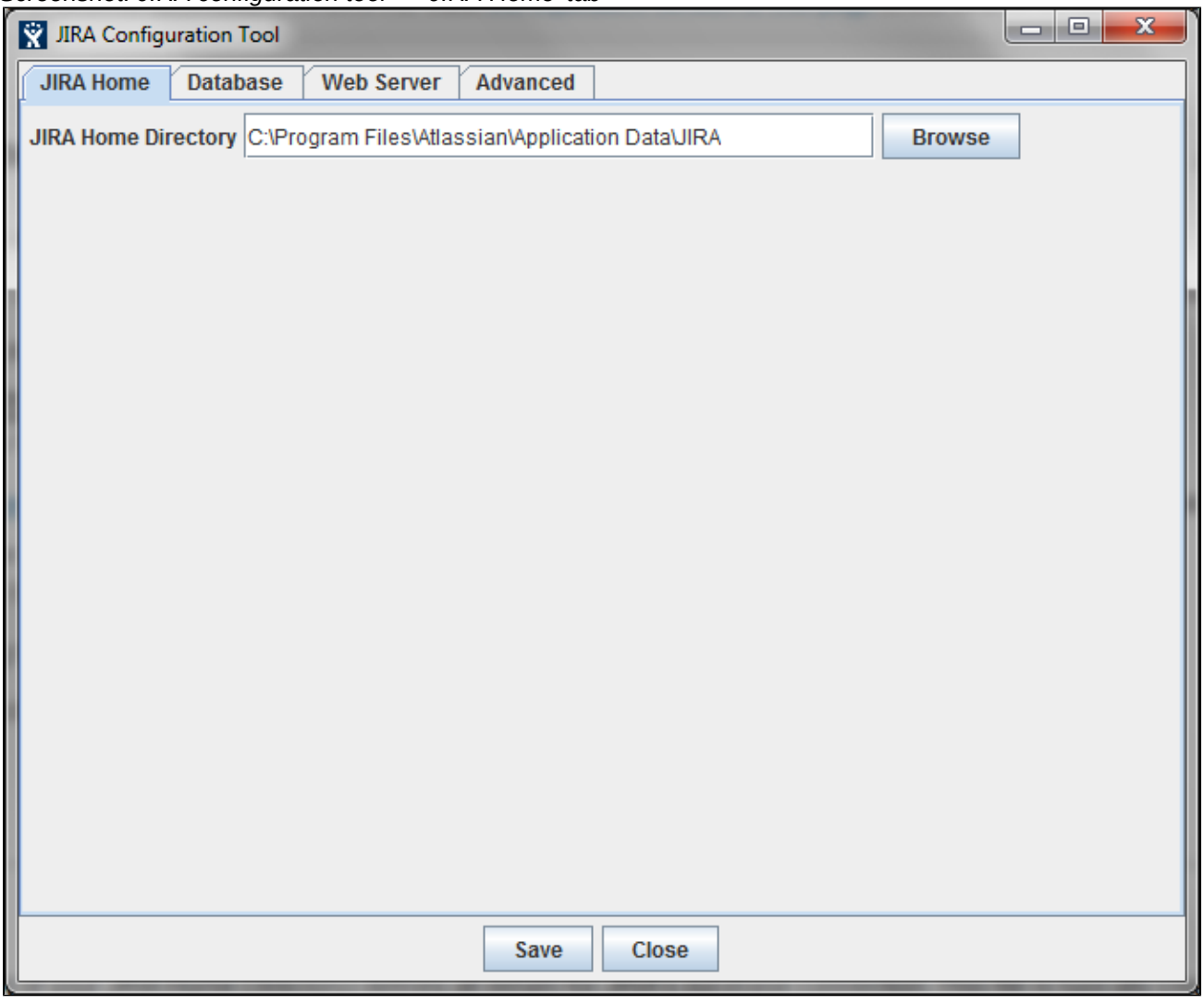
The JIRA configuration tool can be run with a graphical user interface or via a command-line interface using the `-c` or `--console` argument. The following sections show the graphical user interface, but the functionality is the same regardless of the interface.

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 installation directory. For more information, please see Setting your JIRA home directory.

Screenshot: JIRA configuration tool — 'JIRA Home' tab

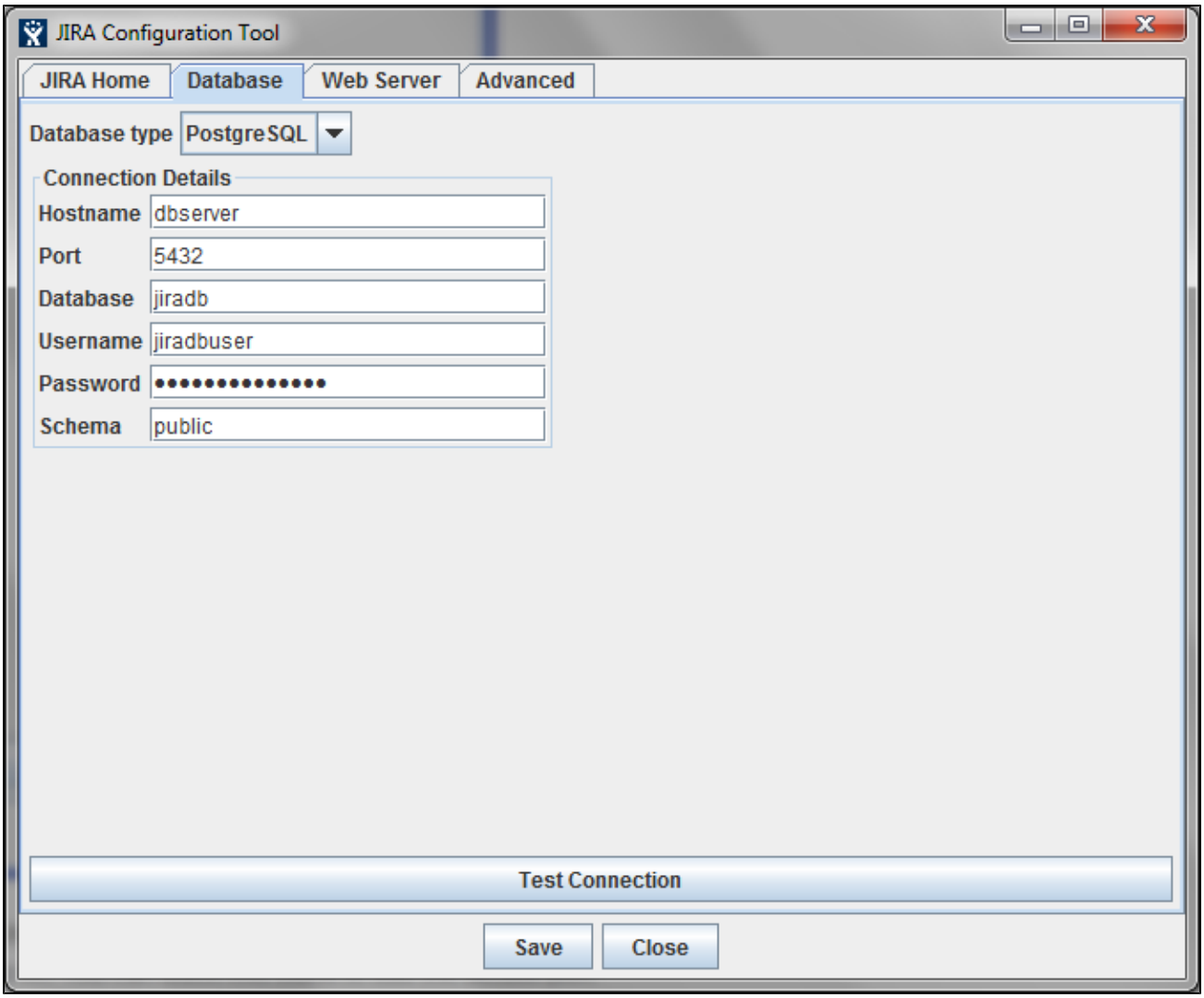


Configuring the database connection

To configure JIRA's database connection using the JIRA configuration tool, follow the appropriate procedure for your database type:

- Connecting JIRA to PostgreSQL
- Connecting JIRA to MySQL
- Connecting JIRA to Oracle
- Connecting JIRA to SQL Server 2008

Screenshot: JIRA configuration tool — 'Database' tab

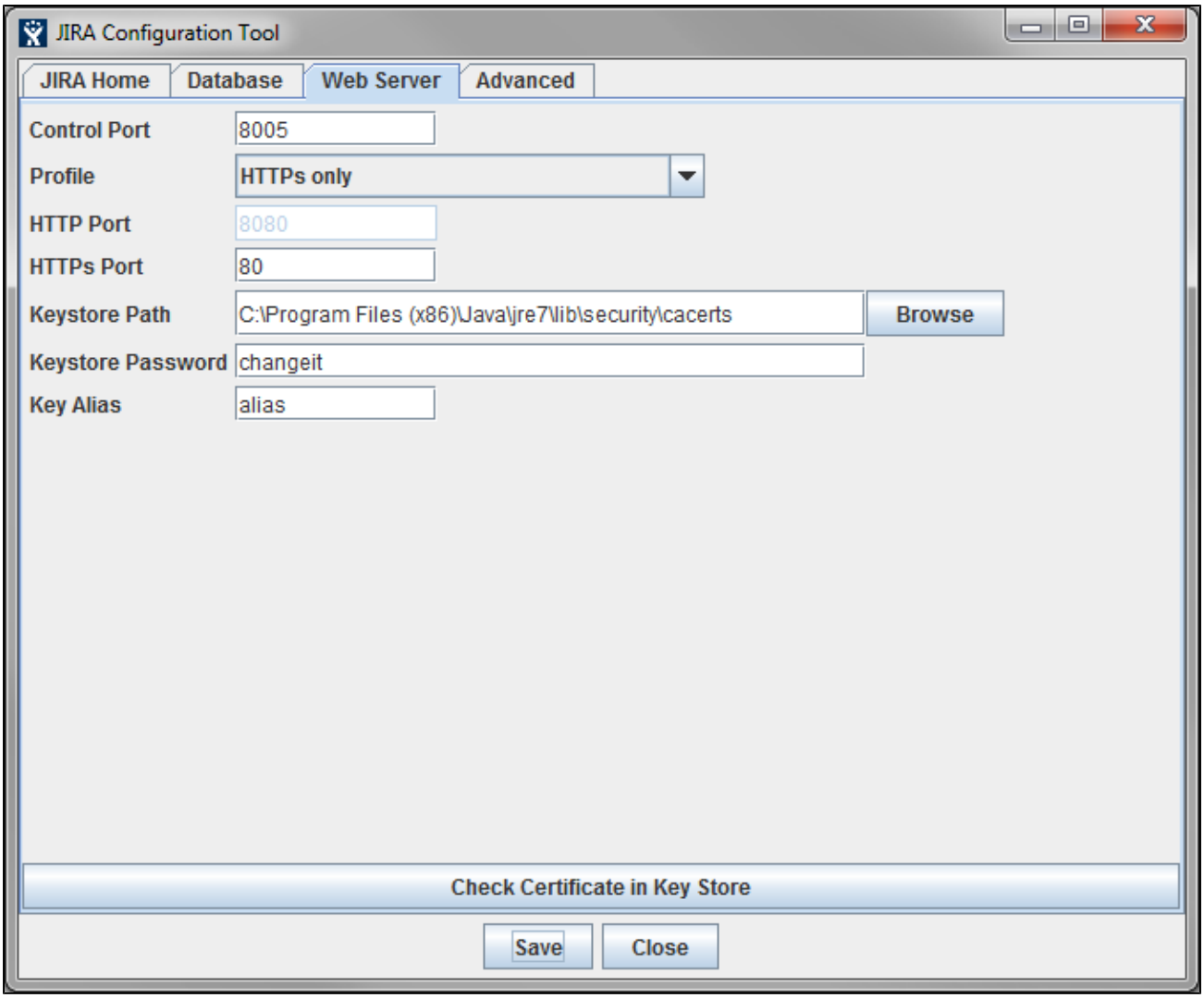


Configuring JIRA's web server

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 over SSL or HTTPS

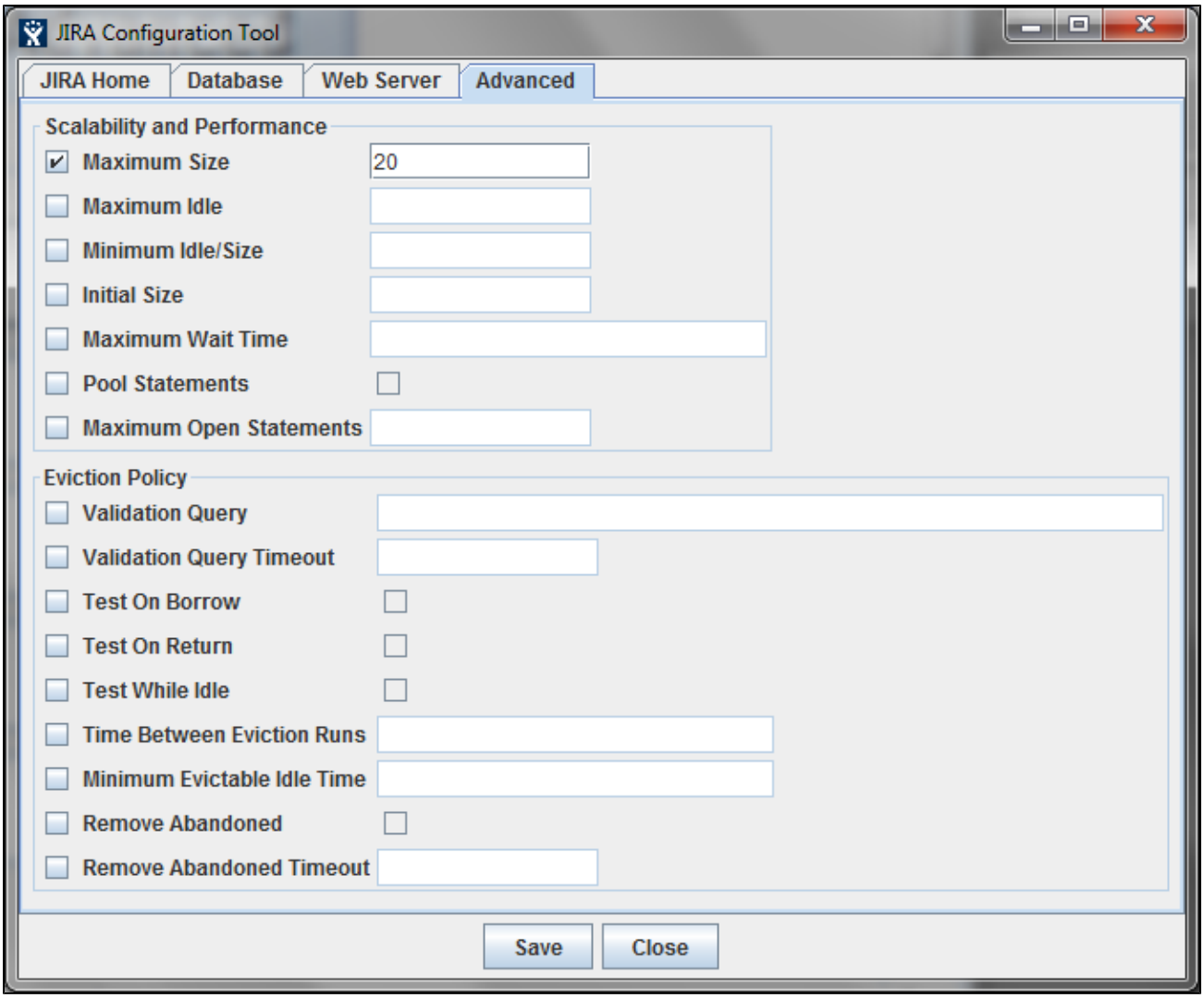
Screenshot: JIRA configuration tool — 'Web Server' tab



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.**

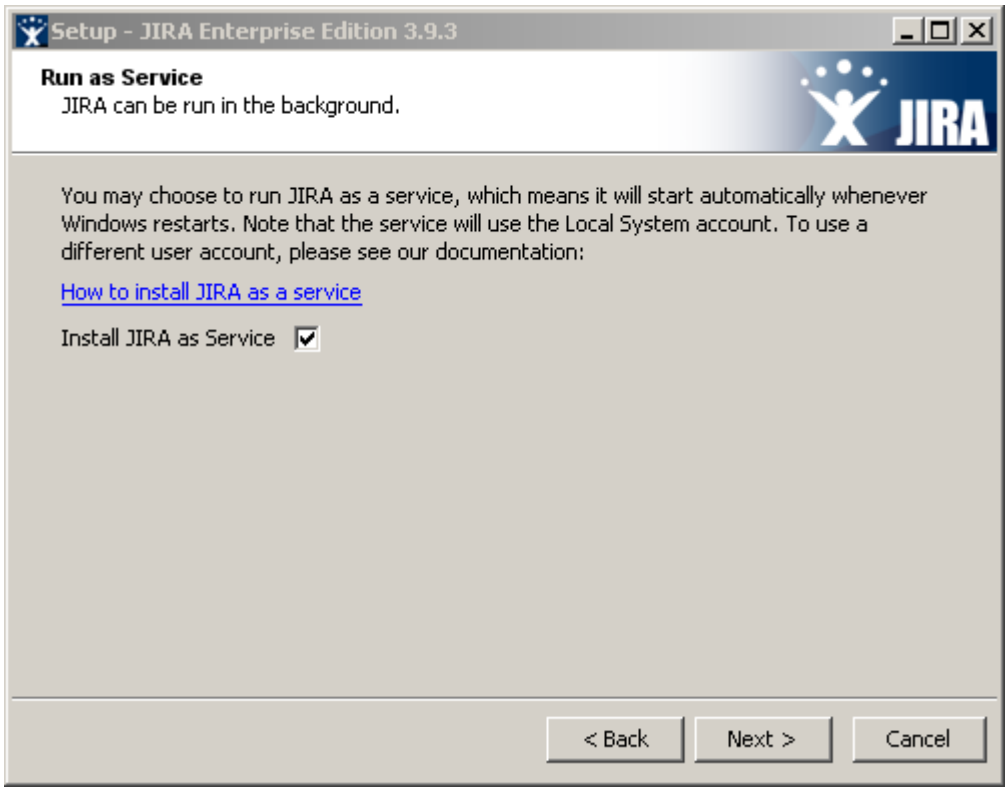
Running JIRA as a Windows service has other advantages. When started manually, a console window opens, and there is a risk of someone accidentally shutting down JIRA by closing this window. Also, the JIRA logs are properly managed by the Windows service (found in `logs\stdout*.log` in your JIRA home 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

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 zervice

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.

Please note:

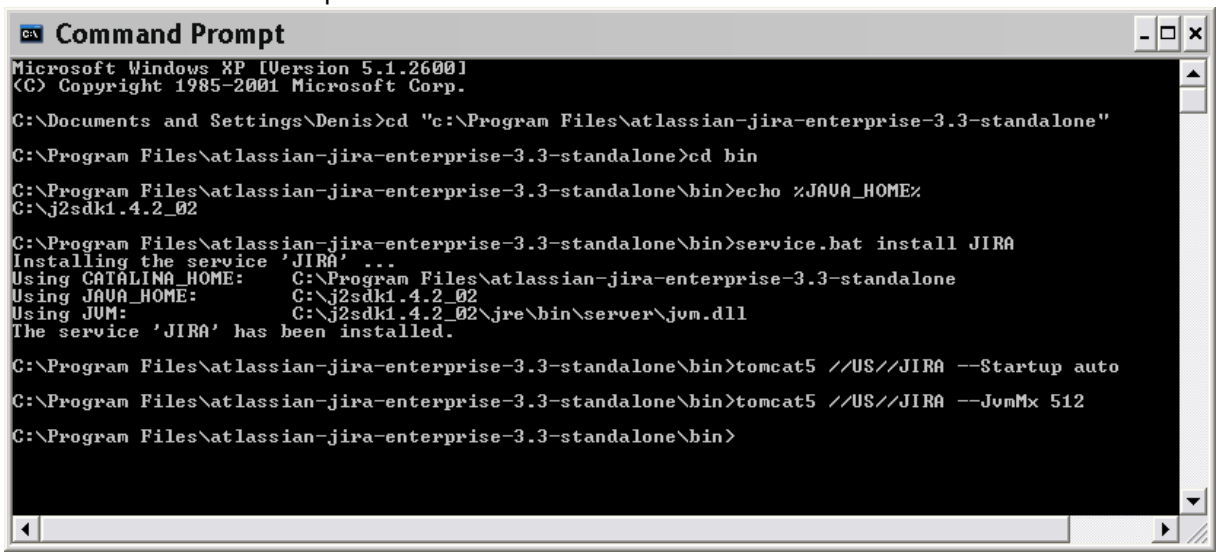
- On any Windows operating system with User Account Control (UAC), such as Windows Vista or Windows 7, you must either disable UAC or run 'cmd.exe' as an administrator (e.g. by right-clicking on 'cmd.exe' and selecting "Run as administrator") in order to execute the script in the procedure below. If 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 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.
   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:

   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:

   ```
   tomcat7 //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:

     ```
     tomcat7 //US//service_name --JvmMx 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:

     ```
     tomcat7 //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:

     ```
     tomcat7 //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:

     ```
     tomcat7 //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 `tomcat7 //DS//JIRA`.

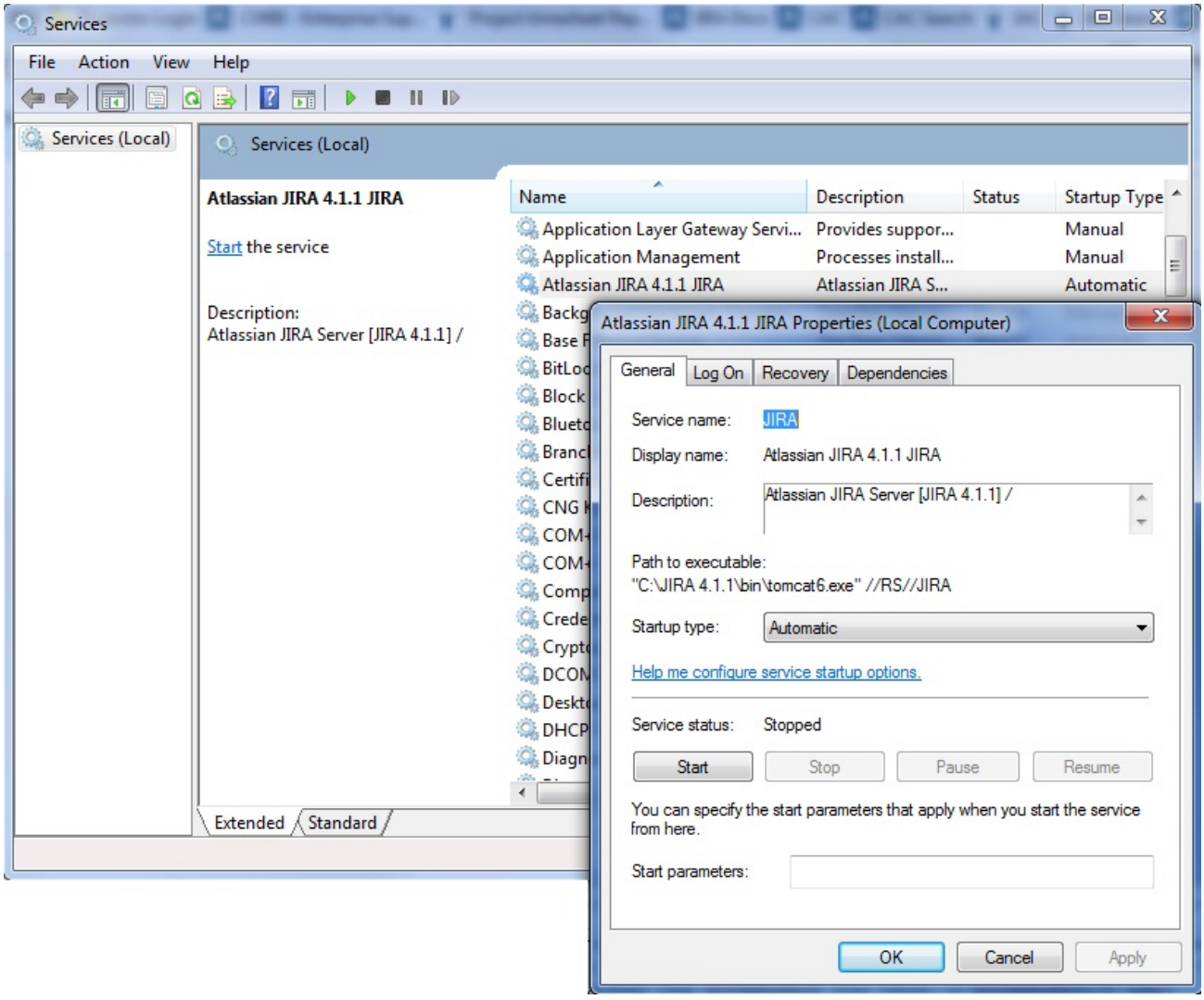
Changing the Windows user that the JIRA service uses

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.

Note that you must also specify these network drives by UNC and not letter mappings, e.g. **\\backupserver\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.

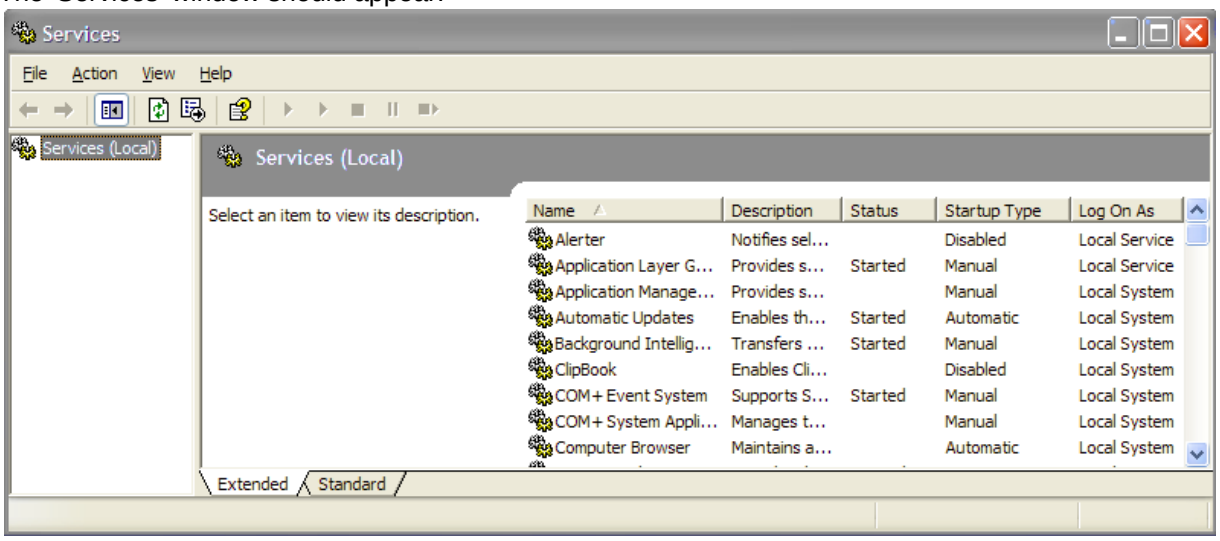
If you wish, you can also set up dependency rules by editing the system registry. Please see http://support.mi

crosoft.com/kb/193888 for details on how to do this.

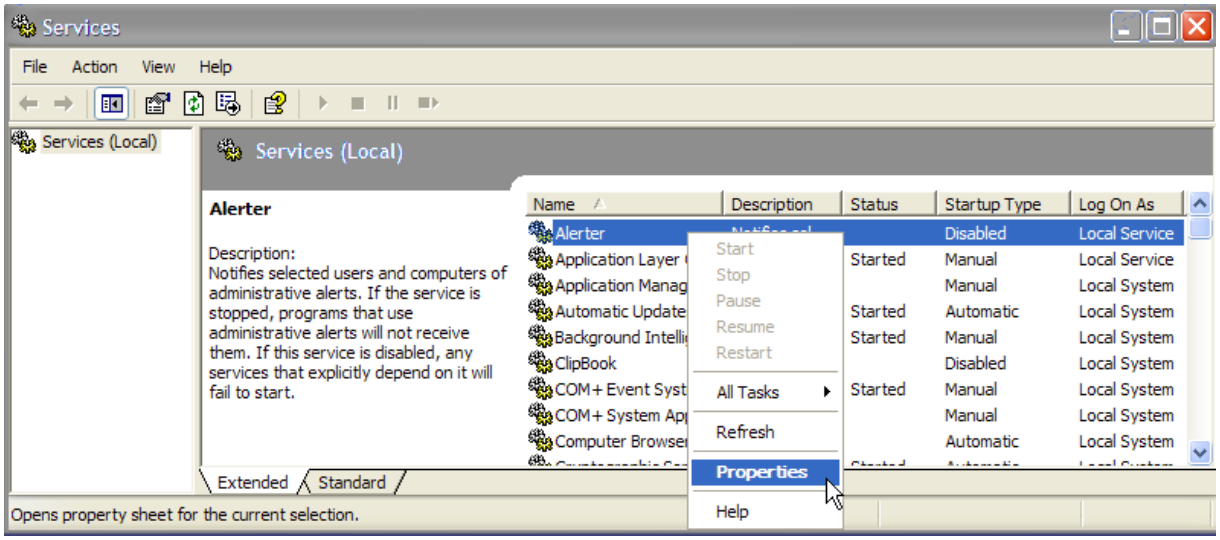
Locating the name of a service

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:

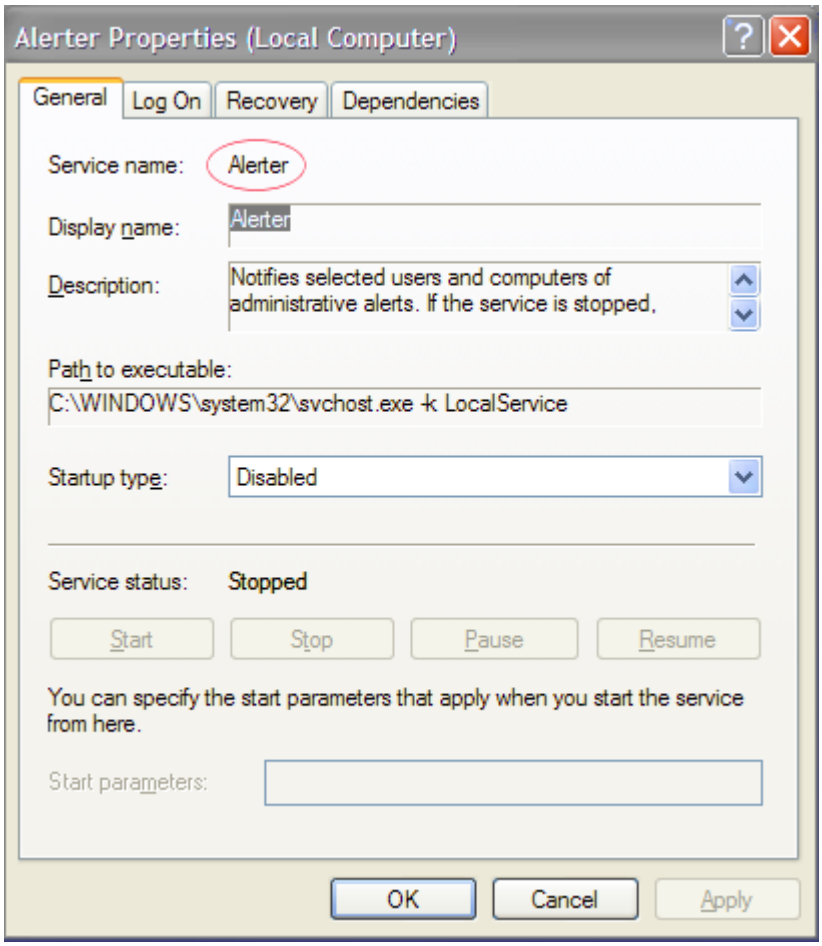
1. Navigate to **'Control Panel' > 'Administrative Tools' > 'Services'**.
2. The 'Services' window should appear:



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.