



Documentation for JIRA Software Server 7.2

Contents

Using JIRA on a mobile device	5
Getting started with JIRA Software	6
Getting started as a JIRA Software manager	7
Setting up your workspace	7
Customizing your project	10
Creating your backlog	11
Grooming your backlog	13
Planning your sprint	15
Tracking your progress	17
Wrapping up your work	19
Doing more with your agile projects	21
Getting started as a JIRA Software user	24
Learn to plan and estimate for scrum teams	27
Plan for the team	28
Customize the team board	30
Estimate in story points	31
Analyze team reports	33
Optimize future plans	36
Installing JIRA Software	38
Using JIRA applications with HipChat	38
Using JIRA applications with Portfolio for JIRA	41
Using JIRA applications with Confluence	44
JIRA Software overview	47
JIRA applications overview	47
Leading an agile project	50
Starting a new project	51
Configuring a project	52
Creating a board	53
What is a board?	56
Configuring a board	58
Configuring filters	60
Enabling ranking	62
Configuring columns	62
Configuring swimlanes	68
Configuring Quick Filters	72
Customizing cards	76
Configuring estimation and tracking	79
Configuring the issue detail view	81
Configuring working days	82
Workflows	83
Organizing work with components	84
Customizing the issues in a project	86
Configuring development tools	86
Configuring collaboration tools	88
Building a backlog	89
Using the backlog	90
Planning a version	92
Configuring versions in a Scrum project	93
Configuring versions in a Kanban project	94
Working with epics	95
Linking a Confluence page to an epic	97
Getting to work	98
Running sprints in a Scrum project	99
Planning sprints	99
Using Active sprints	102
Using Parallel Sprints	104

Monitoring the progress of a sprint	105
Completing a sprint	107
Reopening a sprint	109
Linking a Confluence page to a sprint	112
Checking the progress of a version	113
Using the release page to check the progress of a version	114
Monitoring work in a Kanban project	118
Releasing a version	120
Checking the release status of a version	121
Deploying a release	121
Running a Bamboo build when releasing a version	123
Reporting	124
Burndown Chart	128
Control Chart	130
Comparing different methods of calculating the rolling average on the Control Chart	136
Cumulative Flow Diagram	137
Epic Burndown	139
Epic Report	144
Release Burndown	145
Sprint Report	150
Velocity Chart	151
Version Report	153
Working as a team member in an agile project	155
Searching for issues	156
Basic searching	159
Quick searching	161
Advanced searching	164
Advanced searching - fields reference	170
Advanced searching - keywords reference	198
Advanced searching - operators reference	201
Advanced searching - functions reference	210
Search syntax for text fields	226
Saving your search as a filter	232
Working with search results	236
Constructing cron expressions for a filter subscription	246
Working with issues	247
Creating issues and sub-tasks	248
Creating issues using the CSV importer	251
Editing and collaborating on issues	256
Linking issues	260
Editing multiple issues at the same time	264
Scheduling an issue	269
Moving an issue	271
Approving a service desk request	271
Visual editing	272
Attaching files and screenshots to issues	273
Logging work on issues	275
Estimating an issue	279
Flagging an issue	282
Ranking an issue	283
Transitioning an issue	284
Printing issue cards	285
Viewing the development information for an issue	285
Referencing issues in your development work	289
Processing issues with Smart Commits	290
Configuring dashboards	294
Adding and customizing gadgets	296
Gadgets for JIRA applications	297
Managing your user profile	303
Allowing OAuth access	306
Requesting add-ons	308
Using keyboard shortcuts	309

Administering JIRA Software	310
Permissions overview	311
Managing project role memberships	316
Getting help	316

Using JIRA on a mobile device

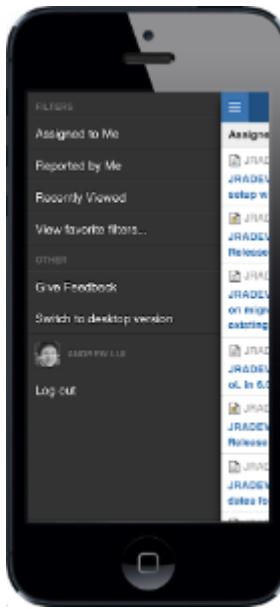
When you view a JIRA page on a mobile device, such as an iPhone or an Android phone, JIRA will display an optimized version of the page. JIRA chooses the mobile or desktop interface based on your device.

The JIRA mobile interface is designed for viewing and interacting with issues on the go. If you need full access to JIRA, you can always switch to the JIRA desktop interface via the mobile menu (shown in the screenshots below).

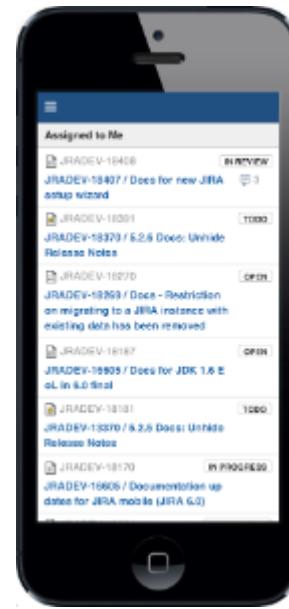
What does JIRA look like on a mobile device?



Viewing an issue on a mobile device



JIRA menu on a mobile device



Viewing issues ("Assigned to Me" filter) on a mobile device

What can you do in JIRA on a mobile device?

The JIRA mobile interface has been designed to give users quick access to their issues on the go. This includes:

- Viewing issues, comments, attachments, issue links and your favorite filters.
- Performing basic operations like adding comments, watching or voting on issues and assigning issues to users.

If you need to create or modify issues on the go, you can still do so by switching to the desktop interface via the mobile menu (shown in the screenshots above).

Frequently asked questions

- [What mobile devices are supported?](#)
- [Do I need to install an app to view JIRA on a mobile device?](#)
- [Can I access my JIRA Cloud site via a mobile device?](#)
- [Why can't I view my custom field in JIRA on my mobile?](#)

What mobile devices are supported?

See [Supported Platforms](#) for details of supported mobile devices.

Do I need to install an app to view JIRA on a mobile device?

No, JIRA is viewed on a mobile device via a web interface (optimized for mobile devices), not an app. Simply browse to your JIRA server's URL using your mobile browser to bring up the mobile interface for JIRA.

Can I access my JIRA Cloud site via a mobile device?

Yes, just enter the URL of your JIRA Cloud site in your mobile web browser.

Why can't I view my custom field in JIRA on my mobile?

The JIRA Mobile interface will show custom fields in the issue details screen. Custom fields that have their own custom field renderer will not display on the JIRA Mobile interface. You will need to switch to the desktop interface to view these fields.

Can I disable JIRA mobile for my Cloud site?

You can disable JIRA mobile for your Cloud site, so that users will only be able to access the desktop view of JIRA on their mobile device.

JIRA mobile is implemented as a add-on in JIRA, so you can disable it by disabling the add-on. For instructions on disabling add-ons, see [Managing Add-ons](#). Note, JIRA mobile is a system plugin.

Getting started with JIRA Software

JIRA Software unlocks the power of agile by giving your team the tools to easily create & estimate stories, build a sprint backlog, identify team commitments & velocity, visualize team activity, and report on your team's progress.

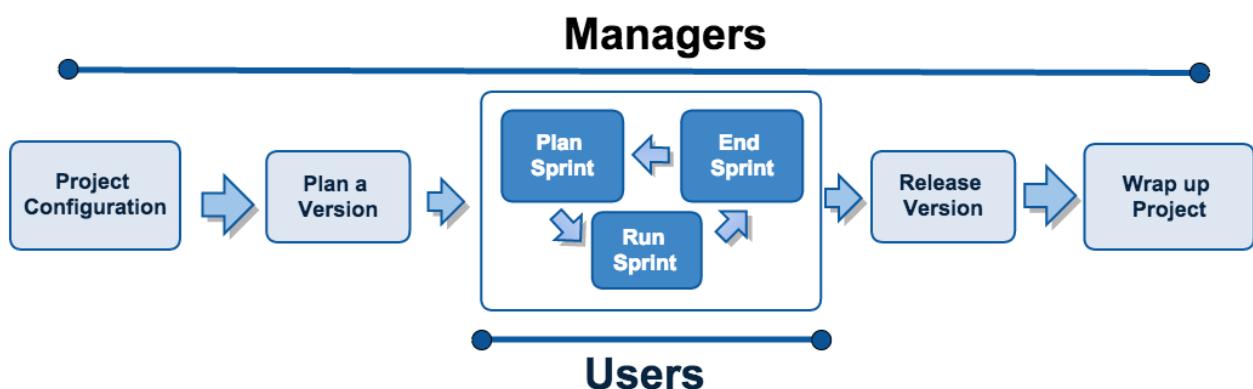
To give your team a tour through a complete project, the following guide contains two tutorials that show a simple agile workflow at a small software company. This guide touches on some of the most used features and follows the development team at Teams in Space as they work to improve their next generation space travel software.

In this simple workflow, a manager sets up and runs a project while individual users add content and work with issues within a sprint. The following pages are set up to follow this structure and are divided into two tutorials:

- Getting Started as a JIRA Software Manager: A guide for managers who set up and run the agile project
- Getting Started as a JIRA Software User: A guide for users who work on and resolve issues

JIRA Software workflow

The simplified workflow and setup in this tutorial is intended for teams using scrum, so some of the concepts presented may not apply to all teams. Here's how the manager and users work together to complete a sprint:



Roles

Typically, variations on the following roles can be found in an agile development environment. We

recommend that managers complete both tutorials while users complete the user tutorial.

JIRA Software Managers

User with administrative rights for your instance of JIRA Software. These roles are usually filled by scrum masters or development managers.

▼ Managers can...

- Access all features in JIRA Software
- Create and update projects
- Add and remove users
- Start and end sprints
- Perform other agile tasks

[Get started as a manager](#)

JIRA Software Users

User who works on and resolves issues. These roles are normally filled by software developers.

▼ Users can...

- Create and manage issues
- Check code review status
- Create branches (with integrated development tools)

[Get started as a user](#)

Getting started as a JIRA Software manager

Welcome to JIRA Software! This tutorial will help you set up your workspace, and create and edit a project for your team. To get your project up and running, we'll also show you how to create, set up, and organize your scrum board.

By the end of this tutorial, you will have:

- Created a new software project
- Added users
- Prepared your backlog
- Started and completed a sprint
- Evaluated the results

But first, let's set up your JIRA Software workspace.

Audience: Scrum masters and software development managers that want to streamline their agile workflow!

Time: 1h 30m

[Let's go!](#)

Setting up your workspace

1. Setting up
your
workspace
2. Customizing
your project
3. Creating
your
backlog
4. Grooming
your
backlog
5. Planning
your sprint
6. Tracking
your
progress
7. Wrapping
up your
work

8. Doing more
with your
agile
projects

When setting up your JIRA Software workspace, you'll need to do the following:

1. Sign up for a JIRA Software site.
2. Create your project.
3. View your Scrum board.

Sign up for a JIRA Software site

If you want to install JIRA Software Server instead of signing up for JIRA Software Cloud, see these instructions: [Installing JIRA applications](#), then skip to the 'Create a project' step below.

You will need your own JIRA Software site for this tutorial. Let's set you up with a JIRA Software Cloud site. **Cloud** is our hosted offering and will allow you to set up your own site without installing a thing! If you already have a site, you can skip this step.

Signing up for JIRA Software will provide you with a fully functional JIRA Software site for one month.

1. Open [this link](#) in a new tab to view the Atlassian Cloud signup page.
2. Follow the signup form steps to enter your site URL and administrator username.
3. Once you have completed the signup process, grab a quick coffee (tea works as well!) — it takes about 10 minutes to create your JIRA Software Cloud site. You'll receive an email when your Cloud site is ready.

Create your project

A JIRA Software project is a collection of issues and tools that allows your team to coordinate the development of a piece of software. Every project contains configurable boards and workflows that you can create and customize to fit your team's workflows.

1. Log in to your Cloud site using the link and credentials you set when you signed up. You'll see the System Dashboard, as shown below.

The screenshot shows the JIRA System Dashboard. At the top, there are navigation links for Dashboards, Projects, Boards, Create, Search, and Tools. The main content area includes sections such as 'Introduction' (Welcome to JIRA), 'Assigned to Me' (no issues assigned), 'Activity Stream' (Your Company JIRA, no activity found), 'Guide for JIRA Administrators' (Getting Started checklist: Create or import your first project, Create your first JIRA issue, Build your team, Add your own style), 'Do more with JIRA' (Install JIRA Agile for project management, Install JIRA Capture for exploratory testing), and 'Administrator documentation' (link to admin documentation). There are also 'I'm done, hide this list' buttons for the checklist and documentation sections.

2. Choose **Projects > Create project**, and then select your project type. Typically, you would choose 'Scrum software development' for iteration planning, or 'Kanban software development' for constraint-based task management.
For this tutorial, let's do Scrum since most software developers use scrum in agile projects.
3. Choose **Scrum software development > Next**.

The screenshot shows the 'Create project' dialog. It has a header 'Create project'. Below it, under 'SOFTWARE', there are three options: 'Scrum software development' (selected, Agile development with a board, sprints and stories. Connects with source and build tools.), 'Kanban software development' (Optimise development flow with a board. Connects with source and build tools.), and 'Basic software development' (Track development tasks and bugs. Connects with source and build tools.). Under 'BUSINESS', there are three options: 'Task management' (Quickly organize and assign simple tasks for you and your team.), 'Project management' (Plan, track and report on all of your work within a project.), and 'Process management' (Track all the work activity as it transitions through a streamlined process.). At the bottom, there are buttons for 'Import a project' and 'Create with shared configuration', and 'Next' and 'Cancel' buttons.

4. In the Name field, enter **Teams in Space**.
5. Choose **Submit** to create your new project.

View your Scrum board

A Scrum board is automatically created with your new project. Use your Scrum board to view and work on issues, such as new features or bugs. You can rank, view, edit, and track issues on your scrum board via the **Backlog**, **Active sprints**, and **Reports**. Don't worry, we'll discuss these three along the way.

This is what your Scrum board will look like:

The screenshot shows the JIRA Scrum board interface. On the left, there's a sidebar with project navigation (Teams in Space), a sidebar for 'Backlog' (with sections for 'Active sprints', 'Reports', 'Issues', 'Test Sessions', 'Timesheets', and 'Add-ons'), and a 'PROJECT SHORTCUTS' section for adding links. The main area is titled 'Backlog' with a sub-section 'VERSIONS'. It displays a placeholder message: 'FILL YOUR BACKLOG WITH ISSUES. This is your team backlog. Create and estimate new issues, and prioritize your backlog using drag and drop.' Below this, it says 'Backlog 0 issues' and has a '+ Create issue' button. A 'Create Sprint' button is also visible. The top right of the main area has 'Board' and 'X' buttons.

Congratulations, you now know the basics of the JIRA Software interface, and have just created your first project! Let's go ahead and customize it!

Next

Customizing your project

1. Setting up your workspace
2. Customizing your project
3. Creating your backlog
4. Grooming your backlog
5. Planning your sprint
6. Tracking your progress
7. Wrapping up your work
8. Doing more with your agile projects

JIRA Software makes it easy to customize your board to fit your workflow. In this step, let's set up your software team, which will be the Teams in Space team. Teams in Space is an imaginary new company pioneering space travel for innovative travel providers.

Add users

A software team without any members just won't cut it! Let's go ahead and configure Teams in Space's development team. The team consists of you (the manager) and two developers (Jennifer and Kevin).

1. Choose  > **User management**.
2. Select **Users**.
3. In the Create new users page, enter the following details for each user:

Full name	Email address
Jennifer Evans	jevans@veryrealemail.com
Kevin Campbell	kcampbell@veryrealemail.com

4. Click **Create users**.

Configure estimation and tracking

One of the first things you need to do for a new Scrum project is to configure estimation and tracking. This is essential for understanding how much work your team has and how much it can do, as you build a backlog, run sprints, and review reports.

Scrum teams use different methods to calculate the amount of work involved in completing an issue, and in turn, a sprint. Many teams separate **estimation** (used for measuring the amount of work in the backlog and calculating velocity) from **tracking** (used for measuring burndown of hours used during the sprint), using different units for each. For example, some teams *estimate* tasks in *story points*, then *track* tasks using *hours*.

In this tutorial, Teams in Space uses story points for both estimation and tracking, as per the instructions below.

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Estimation** tab.
3. In the **Estimation Statistic** field, select **Story Points**. Leave the **Time Tracking** field set to **None**.

Estimation

Issues can be estimated when in the Backlog to get an idea of how much work is being committed to in a sprint. [Read more about estimation and tracking](#).

Estimation Statistic **Story Points**

Estimate issues in the Backlog by entering values for Story Points. Your velocity from sprint to sprint will be measured against these estimates.

Time Tracking **None**
 Issues will burn down their Story Points value upon completion.

Remaining Estimate and Time Spent
 Track time against issues using JIRA's Remaining Estimate and Time Spent fields.

Great! You've successfully added users and configured your project to use story points for estimates. Now, let's work on your backlog!

[Previous](#)

[Next](#)

Creating your backlog

1. Setting up your workspace
2. Customizing your project
3. Creating your backlog
4. Grooming your

- backlog
- 5. Planning your sprint
- 6. Tracking your progress
- 7. Wrapping up your work
- 8. Doing more with your agile projects

Your backlog is a list of tasks that represents outstanding work in a project. Usually, a project would have issues in the backlog, and you can add these issues to a sprint so your team can work on them. Since Teams in Space is a new project, you won't have issues on your backlog. Let's create some work for your team.

How do I get to my board?

If you navigate away from your board, you can easily return to it by selecting Boards from the top navigation menu, and selecting your board "Teams in Space" from the list of recent boards.

Backlog

The Backlog gives you a place to organize your sprints. You can create new issues or sub-tasks, organize your backlog, create versions, organize via epics, and start sprints.

Issue ID	Description
TIS-1	Expand travel to destinations outside of The Solar System
TIS-2	Build out a local office on Mars
TIS-3	Add support for teams larger than 20 people
TIS-4	Next Generation version of SeeSpaceEZ travel platform
TIS-5	Plans for our Summer Saturn Sale
TIS-6	Make working with our space travel partners easier
TIS-7	500 Error when requesting a reservation
TIS-8	Requesting available flights is now taking > 5 seconds

Create issues

By default, any team member can create issues. In this tutorial, however, you will create all of the backlog issues.

1. On the Teams in Space board, choose **Create** at the top of the screen.

Create Issue

Project *

Issue Type *

Summary *

Priority

Component/s

Description

Create another

Tip! If you select the **Create another** checkbox, JIRA Software will create your issue and automatically pre-populate a new 'Create Issue' dialog box with your previous issue details.

Fill in the fields using the data shown below. Only the fields with * are mandatory.

- **Project:** Teams in Space
- **Issue Type:** Story
- **Summary:** Expand travel to destinations outside of The Solar System
- **Priority:** High
- Leave all other fields blank or at their default values.

2. Choose **Create** to make a new issue. An issue key (**TIS-1**) is created for this issue, which comes in handy when searching for issues later.

Add more issues

One issue isn't enough to get your team working! Let's add more issues so you can create and run a complete sprint. Create the following issues using the same steps as above.

Type	Summary	Priority
Story	Build out a local office on Mars	High
Story	Add support for teams larger than 20 people	High
Story	Next Generation version of SeeSpaceEZ travel platform	Low
Story	Plans for our Summer Saturn Sale	High
Story	Make working with our space travel partners easier	Low
Story	500 Error when requesting a reservation	Medium
Story	Requesting available flights is now taking > 5 seconds	Highest

Great! Your backlog's all set! Now, let's plan some work for your developers to do!

[Previous](#)

[Next](#)

Grooming your backlog

1. Setting up
your
workspace
2. Customizing
your project
3. Creating
your
backlog
4. Grooming
your
backlog
5. Planning
your sprint
6. Tracking
your
progress
7. Wrapping
up your
work
8. Doing more
with your
agile
projects

An essential part of agile is regularly "grooming" or reviewing the contents of your backlog, particularly before starting any new work.

Adjust your backlog

Before starting your sprint, you need to prepare your backlog. You can easily adjust your backlog by:

- Right-clicking on issues to view, estimate, or add details
- Ranking your issues by dragging and dropping
- Creating new issues
- Editing an issue using the issue detail view

Estimate issues

Now, let's add some estimates to the issues in our backlog. This way, you can easily determine what you can accomplish, and your team can also have a way of measuring the success of the sprint.

1. On the Teams in Space board, select **Backlog**.
2. Select each issue on the left-hand side of the screen to display the issue details on the right-hand side of the screen.
3. Click the **Estimate** field on the right-hand side of the screen for each of the issues, and enter the following information for each issue:

Issue	Estimate
Expand travel to destinations outside of The Solar System	6
Build out a local office on Mars	2
Add support for teams larger than 20 people	4
Next Generation version of SeeSpaceEZ travel platform	3
Plans for our Summer Saturn Sale	8
Make working with our space travel partners easier	16

500 Error when requesting a reservation	6
Requesting available flights is now taking > 5 seconds	4

Rank the backlog

By default, the issues in your backlog are ranked in the order in which you added them. You can change the rank of your issues according to their relative priority. This helps you organize the issues in your backlog more effectively.

Rankings let you determine whether an issue is more important or urgent than another issue. For example, you may have two separate issues that are both of 'High' priority. Using JIRA Software ranking, you can assign one of the issues a higher ranking than the other.

1. Find issue **TIS-8** in your backlog. This issue has the 'Highest' priority, and therefore should be at the top of your backlog.
2. Select **TIS-8** and drag it to the top.
3. Move issues **TIS-2** and **TIS-5** to positions two and three in the backlog. These issues have 'High' priority, but they're not as high a priority as TIS-8.

The screenshot shows the JIRA Software backlog interface. On the left, there are navigation tabs for 'VERSIONS' and 'EPICS'. The main area is titled 'Backlog 8 issues' and contains the following list of issues:

- TIS-8 Requesting available flights is now taking > 5 seconds (Priority: 4)
- TIS-2 Build out a local office on Mars (Priority: 2)
- TIS-5 Plans for our Summer Saturn Sale (Priority: 8)
- TIS-1 Expand travel to destinations outside of The Solar System (Priority: 6)
- TIS-3 Add support for teams larger than 20 people (Priority: 4)
- TIS-4 Next Generation version of SeeSpaceEZ travel platform (Priority: 3)
- TIS-6 Make working with our space travel partners easier (Priority: 16)
- TIS-7 500 Error when requesting a reservation (Priority: 6)

On the right side of the backlog list, there are several icons: a 'Create Sprint' button, a gear icon, a 'Create issue' button, and other management icons.

Great! You've just groomed your backlog. Now, let's plan out the details of your sprint!

[Previous](#)

[Next](#)

Planning your sprint

1. Setting up your workspace
2. Customizing your project
3. Creating your backlog
4. Grooming your backlog
5. Planning your sprint
6. Tracking your progress
7. Wrapping

- up your work
8. Doing more with your agile projects

A sprint is a short period (ideally two to four weeks) during which a development team implements and delivers a discrete product increment, e.g. a working milestone version. In this tutorial, your team will be working in two-week long sprints. Let's go ahead and create a sprint for your team.

Sprint planning

Before creating and starting a sprint, your Scrum team would typically hold a sprint planning meeting. In this meeting, your team should:

- Review the estimates for selected issues
- Break down the selected issues into an initial list of sprint tasks
- Consider upcoming employee time-off, holidays, and other issues that may impact the completion of these sprint tasks
- Gauge the team's capacity team to complete these sprint tasks

By the end of the meeting, your team should be confident enough to commit to completing the work in the sprint.

In this tutorial, we will assume that the Teams in Space team can handle 20 story points of work in a sprint, and that everyone is available for the full sprint. Typically, you would know how much work your team can complete in a sprint by reviewing information from past sprints, usually through velocity and sprint reports.

Need help scheduling?

By using JIRA Software in tandem with Confluence, you can embed your sprints using Team Calendars for Confluence. This helps you see the duration of your sprint and how your team's availability or other team events could impact the sprint.

Create a sprint

1. On the Teams in Space board, click **Backlog**.
2. Click **Create Sprint** at the top of the Backlog.
3. Your new upcoming sprint will be added to your board, below any other future sprints. Select the **Sprint 1** text and edit the name of the sprint to 'Spring Cleanup'.
4. The top 4 issues in the Backlog are equal to 20 story points. This is what the team estimated that they could accomplish in the upcoming sprint. Drag and drop the top four issues from the Backlog into your new sprint.

Start your sprint

Streamline your work

By connecting your JIRA Software instance to a Confluence instance, you can link Confluence pages to your sprints via JIRA Software to build stronger user stories and to better plan for sprints or releases. For example, you may want to write up the sprint meeting notes in Confluence and link them to the sprint.

Now that you have created a sprint, you can go ahead and start it.

1. Click **Start Sprint**.
2. Today's date and current time become the start date and time for the sprint. For the purpose of this tutorial, enter an end date of 5 minutes from the start date and time.
3. Select **Start** to start the sprint and move the issues into Active sprints.

Congratulations! You've successfully got Teams in Space up and running! Now, let's look at Active sprints to track your team's progress.

[Previous](#)

[Next](#)

Tracking your progress

1. Setting up your workspace
2. Customizing your project
3. Creating your backlog
4. Grooming your backlog
5. Planning your sprint
6. Tracking your progress
7. Wrapping up your work
8. Doing more with your agile projects

During your sprint, you and your team will need to monitor your progress to make sure that everyone is on the same page. There are several tools that you can use to do this, which are described below.

Active sprints

The **Active sprints** page is where you monitor the progress of your team's work during a sprint. Here, your team can transition issues through a series of columns (statuses), allowing everyone to quickly visualize the progress being made in the sprint. You can also edit issues by adding information, such as descriptions, attachments, and comments.

Active sprints: Spring Cleanup

QUICK FILTERS: Only My Issues Recently Updated

0 days remaining Complete Sprint Board A

Columns	A column displays the current status of your issues in your workflow
To Do	TIS-8 Requesting available flights is now taking > 5 seconds TIS-2 Build out a local office on Mars TIS-1 Expand travel to destinations outside of The Solar System
In Progress	TIS-5 Plans for our Summer Saturn Sale
Done	

Swimlanes You can configure swimlanes using filters, such as stories, epics, assignees, etc

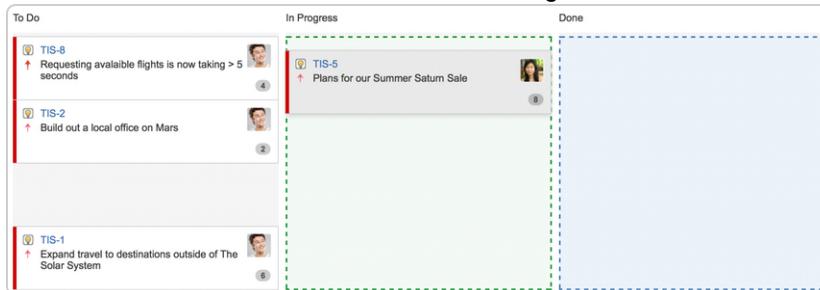
More about Active sprints

Issues that you ranked higher in your backlog appear at the top of the column, to make it easier for users to select the most important issue to work on.

Transition issues

During the sprint, let's say your team was only able to finish three issues, TIS-1, TIS-2, and TIS-5. Issue TIS-8 remains in the 'To Do' column since no team member was able to work on the issue. Let's show on the board what happened during the sprint by transitioning issues from one column to another.

1. On the Teams in Space board, click **Active sprints**.
2. Select **TIS-5** and move the issue to the 'In Progress' column.



Tip!

Use <Ctrl> or <Shift> to select multiple issues, and then drag and drop the issues to a column.

3. Select **TIS-5**, **TIS-2**, and **TIS-1**, and then move the issues to the 'Done' column.

View the Burndown Chart

You've just seen how your team is progressing, from the Active sprints of your board. The Burndown Chart is another useful tracking tool, which can help you visualize your team's progress, as well as determine whether your team is on target to achieve the sprint goal.

1. On the Teams in Space board, click **Reports**.
2. Select **Burndown Chart**.



The grey line in your Burndown Chart is a guide showing the rate of work required to complete the sprint. The

red line, on the other hand, shows the actual work completed by your team.

If your Burndown Chart shows the red line above the grey line, your team may not achieve the sprint goal. You may want to consider removing some issues from the sprint. Any changes to scope (e.g. issues added to sprint, issues removed from sprint) are shown in the table below the graph.

Well done! You now know how to track the progress of your team's sprints! Next, let's wrap up your work!

Previous

Next

Wrapping up your work

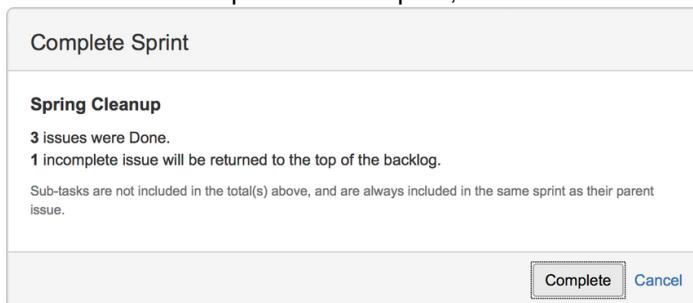
1. Setting up
your
workspace
2. Customizing
your project
3. Creating
your
backlog
4. Grooming
your
backlog
5. Planning
your sprint
6. Tracking
your
progress
7. Wrapping
up your
work
8. Doing more
with your
agile
projects

After your team finishes the work in a sprint, you and your team can then perform a retrospective of the sprint. Sprint retrospectives help determine where the team succeeded, and where improvements can be made. The more sprints completed by your team, the more data you can use to find significant areas for improvement.

End the sprint

Once your team reaches the end date of the sprint, you need to end the sprint — regardless if this means some issues in the sprint are not yet completed.

1. On the Teams in Space board, click **Active sprints**.
2. Select **Spring Cleanup** from the Active sprints drop-down.
3. Click **Complete Sprint**. The Complete Sprint dialog box will be displayed, showing the number of issues that are completed in the sprint, and the number of issues that were not completed.



In this case, the incomplete issues are moved to the backlog. However, if you had more sprints planned, then they would be moved to the next planned sprint. Note, you can always add issues that are returned to the backlog to another sprint, if you wish.

View the Sprint Report

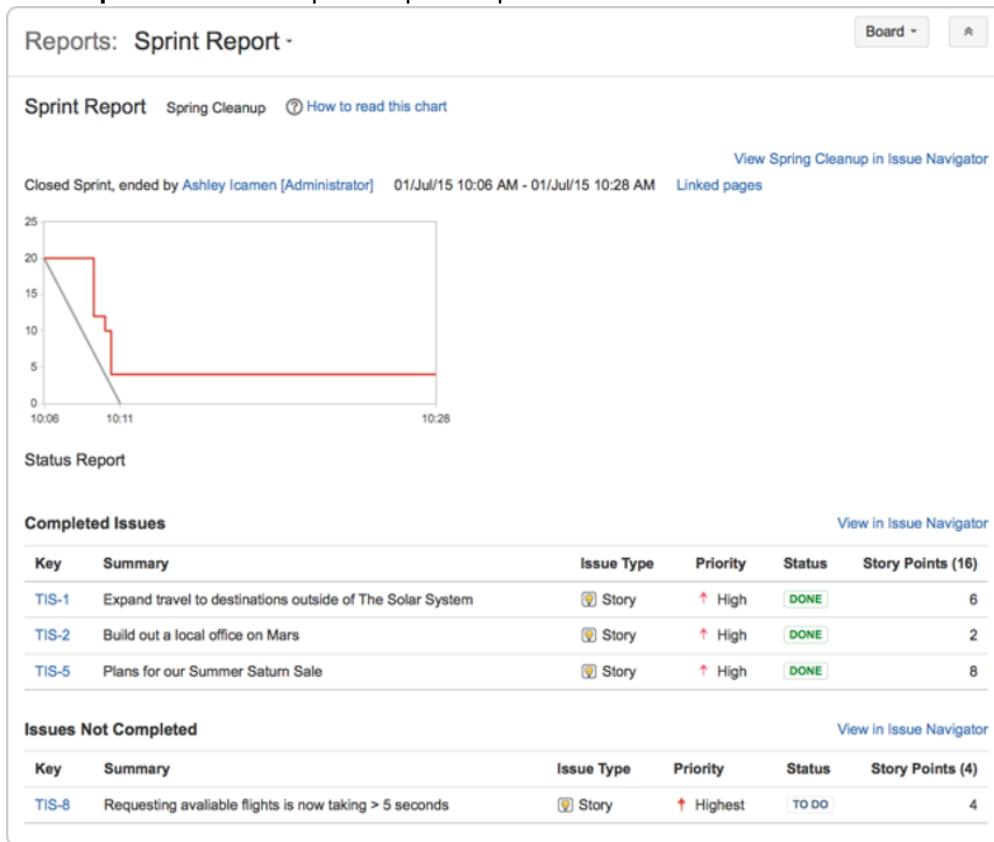
After a sprint, your team can hold a sprint retrospective meeting, to determine your wins for the sprint, as well as point out the potential areas of improvement that your team can tackle in future sprints. You can use the Sprint Report during sprint retrospective meetings to do this.

The Sprint Report shows a status list of issues in each sprint. It also provides a breakdown of the progress, status, and estimation information for each issue. You can also use the Sprint Report to perform progress checks in the middle of a sprint.

Looking for more reports?

There are many other reports available for your team to use in the Reports page. You can also create your own reports, or use reports in the Atlassian Marketplace.

1. On the Teams in Space board, click **Reports**.
2. Select **Sprint Report** from the Reports drop-down.
3. Select **Sprint 1** from the Sprint Report drop-down.



Other tools you can use

Other useful tools can give your team a better visual of the improvement areas – and more importantly, these tools can help your team figure out action plans for these areas. For this tutorial, some of the tools you may want to use are:

- Velocity Chart
- Release Hub

Velocity Chart

You can use the Velocity Chart to track the amount of work your team completes from sprint to sprint. Using the Velocity Chart lets you predict a more realistic amount of work that your team can commit in future sprints.

1. On the Teams in Space board, click **Reports**.
2. Select **Velocity Chart** from the Reports drop-down.



Release Hub

Aside from using reports (like the Sprint Report and Velocity Chart), you can also [monitor the progress of a version](#) after you complete a sprint. Monitoring a version's progress helps you see problems early, as well as determine the likelihood of releasing a version on time.

One more step to go! Let's look at some tips and tricks to help you get the most out of JIRA Software.

[Previous](#)

[Next](#)

Doing more with your agile projects

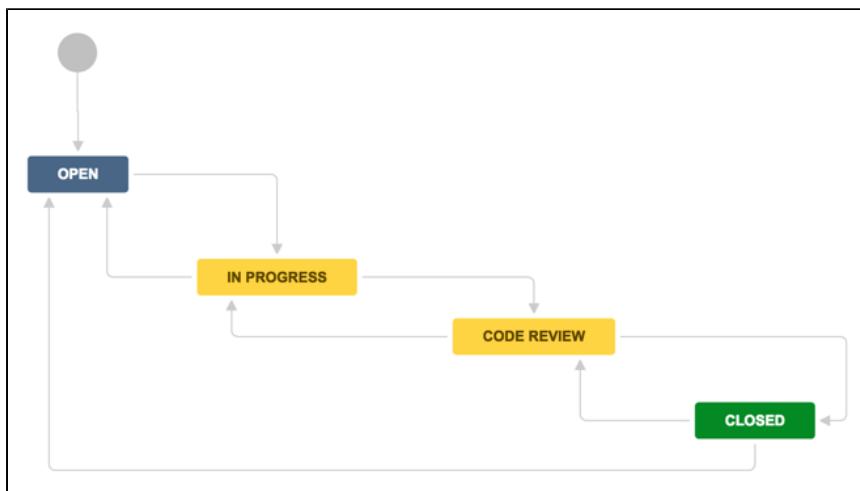
1. Setting up
your
workspace
2. Customizing
your project
3. Creating
your
backlog
4. Grooming
your
backlog
5. Planning
your sprint
6. Tracking

- your
progress
- 7. Wrapping
up your
work
- 8. Doing more
with your
agile
projects

Want to become a JIRA Software ninja? Take a look at these advanced topics!

Workflows

A workflow is a set of *statuses* and *transitions* that an issue goes through during its lifecycle. JIRA Software makes it easy to create and edit workflows that fit your team's needs. Click [here](#) for more information on creating and editing workflows.



Development panel

JIRA Software can be connected to a range of development tools, such as repository managers, code review applications, and build and deployment managers. These tools can help you keep your project tracking in sync with your development work. Click [here](#) for more information on integrating development tools.

UI should allow users to book on large or personal space craft

Details

Type:	Story	Status:	IN PROGRESS (View Workflow)
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	1.5	Fix Version/s:	3.0
Component/s:	Web Site		
Labels:	None		

Description

Currently LocalTransportController makes an assumption that all the participants in the group are on the same itinerary. Many of our local travel providers limit reservations to 4 people.

The UI should walk the user through booking a large travel vendor for the group and allow certain users to opt out for a personal space craft.

Activity

All Comments Work Log History Activity

There are no comments yet on this issue.

People

- Assignee: Ke [Assign](#)
- Reporter: Jen [Report](#)
- Votes: 0
- Watchers: 0 [Start watching](#)

Dates

Created: 25/May Updated: Yesterday

Time Tracking

Estimated: Remaining: Logged:

Development

4 commits 1 pull request [MERGED](#) 1 build

[Deployed to QA](#)

Notifications

Email other JIRA users a link to an issue either by sharing the issue with them, or by mentioning them in an issue's **Description** or **Comment** field. See [Editing and collaborating on issues](#) for more information on notifications in JIRA Software.

People

Assignee: [Link to Issue](#) <http://localhost:8080/browse/TIS-47>

Reporter: [User name or email](#) jennifer

Votes: [Showing 1 of 1 matching users](#)

Watchers: [Jennifer Evans - jevans@veryr...](#)

Dates

Created: Updated:

Time Tracking

Estimated: Remaining: 0m Logged: 3d 4h

Custom cards

Customize issue cards for your boards to bring the right information to your team's attention at a glance. You can change the card colors to help people quickly identify cards on your board as being of a particular issue type, priority, assignee, or any JQL that you want. Click [here](#) for more information on custom cards.

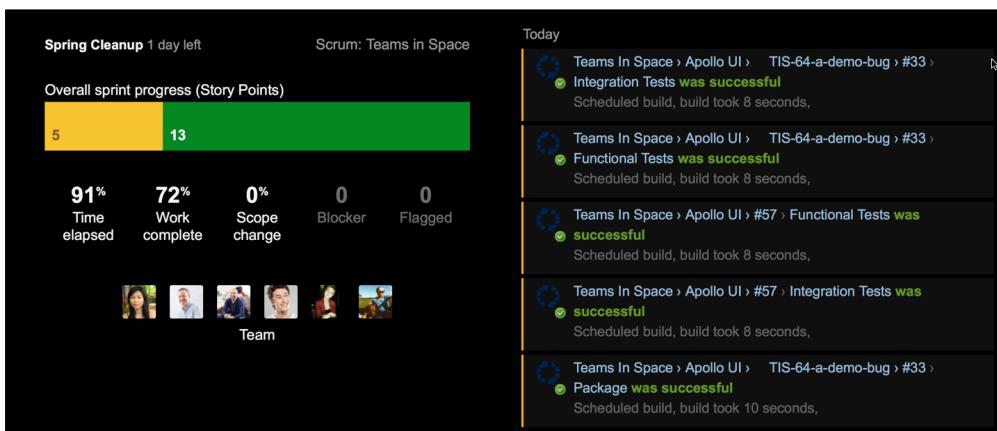


Confluence integration

You can link your JIRA Software instance to a Confluence instance to create and link Confluence pages to your [sprints](#) and [epics](#). For example, you can link your sprint meeting notes in Confluence directly to the relevant sprint. This makes it easy for your team to quickly share information about the sprint. Click on [sprints](#) or [epics](#) for more information on linking your project to Confluence.

Wallboards

A wallboard runs on a wall display and is used to monitor vital data about a project's progress. You can use wallboards to share information with your team during stand-ups. You can also add gadgets to a wallboard – make sure to add gadgets that are useful to your team, such as a gadget that displays the status of a sprint, including issue status, time remaining, and build status. Click [here](#) for more information on creating wallboards and setting up other gadgets for JIRA Software.



What next?

So that's it — we hope this guide has helped you get a feel for JIRA Software. You can continue your training by completing the [Getting started as a JIRA Software user](#) tutorial, checking out [our documentation](#), or visiting our community on [Atlassian Answers](#) for more information.

[Previous](#)

Getting started as a JIRA Software user

Welcome to your JIRA Software project! In this tutorial, we'll introduce you to your project and walk you through a simple workflow – from finding an issue assigned to you, to completing an issue. By the end of this

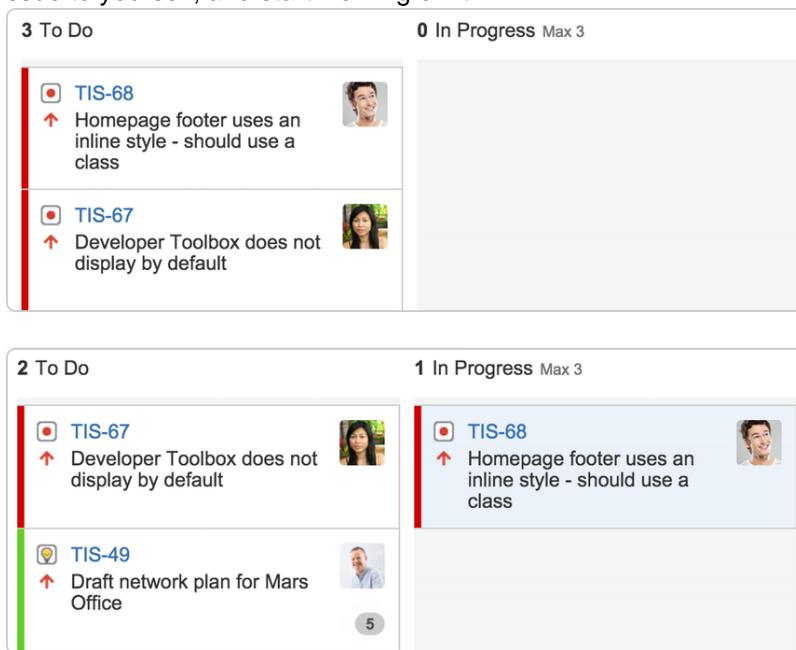
tutorial, you will have completed the following steps as a JIRA Software user going through a simple workflow:

[Find your issue](#) | [Work on an issue](#) | [Collaborate with your team](#) | [Create a branch](#) | [Start coding](#) | [Send your work for review](#) | [Finish your work](#)

Find your issue

Typically, your Development Manager will have already created a Scrum or Kanban project, and populated the backlog with issues. There are a number of ways to navigate around your project, but the easiest way to view information is by using boards. A board displays issues from one or more projects – it gives you a flexible way of viewing, managing, and reviewing your work. In this step, we describe how you would typically find an issue to work on, in the active sprint of the board of a Scrum project.

1. Select **Boards** from the top menu, and then select a board from the drop-down list.
2. Select **Active sprints** to see the issues in the sprint of the selected board.
3. Issues are represented by issue cards in the Active sprints of your board. The cards show quick reference information, like issue keys, assignees, and descriptions. Locate the top card in the **To Do** column and drag it to the **In Progress** column. By doing this, you are actually moving to assign the issue to yourself, and start working on it.



Work on an issue

By moving an issue to In Progress, you are indicating that work has started on it. At this stage, you will need to assign it to yourself, so that everyone knows who is working on it. You may also want to add some technical details about the issue – it's helpful for your teammates to know as much information as possible about the issues you're working on.

1. In the **Active sprints** of your board, select the issue in the **In Progress** column. The issue is displayed in the Issue Detail view.
2. In the Issue Detail view, select **Edit** from the 'cog' drop-down. The 'Edit Issue' dialog will be displayed.
3. If the issue is not assigned to you yet, type your name in the **Assignee** field.
4. Add some technical details about the issue as necessary.
Note, you may want to specify a component or a version (if components and versions have been configured for your project), or add attachments as needed, etc.
5. When you're done adding technical details, click the **Update** button.

Collaborate with your team

While you're working on an issue, you will probably need to share some information, clarify some requirements, or discuss some details about the issue with your team. This is easy to do with issue comments.

1. Select the issue to add a comment.
2. In the Issue Detail view, locate and click the **Comment** button.
3. In the **Comment** text box, type your comment.
4. To email other users about your comment, simply mention these users in the **Comment** text box (by typing @User's Name). An email will be sent to the users' email addresses that are registered with their JIRA accounts.
5. Click the **Add** button to save the comment.

Create a branch

After assigning an issue to yourself and entering technical details about the issue, you're ready to do some coding. It is recommended that you create a branch when you start working on an issue. This way, you'll have your own stream of work that won't interfere with the rest of the team's work. It also ensures that your changes get reviewed before being merged back into the master branch.

JIRA Software can be linked to a code hosting or repository management source, like [Bitbucket Cloud](#) or [Bitbucket Server](#). This lets you open, collaborate, and manage source code directly from within JIRA. Integrating an application lets you and your team create a branch directly from an issue, giving you a faster workflow from selecting an issue to coding.

Before creating a branch, you should already know how to use branches in the source repository that your team is using ([Bitbucket Cloud instructions](#), [Bitbucket Server instructions](#)).

1. Select the issue for which you want to create a branch.
2. In the Issue Detail view, locate the **Development** panel, and click **Create branch**. The Create branch dialog will be displayed.
3. Include the issue key in the branch name. If you have workflow triggers configured, the status of the issue may automatically transition to 'In Progress'. See [Configuring workflow triggers](#) for more information.
4. Enter other details for the branch as necessary.
5. Click the **Create branch** button. Your branch will be created in your source repository.

Start coding

Now that you have created a new branch, you can start coding without worrying about your changes affecting the master branch. Ensure that you add the issue key that references the issue that you're working on. When you [reference your issue key\(s\) in your development work](#), your connected development tools will also have links back to the relevant issues.

1. Go to your source repository where your new branch is created, and access your new branch.
2. In your new branch, implement your code or fixes as necessary.

Send your work for review

Use pull requests to tell your teammates about changes you've pushed to the repository. Once a pull request is sent, your team can review, discuss changes, or push follow-up commits.

1. Click **branches** in the Development panel to open a dialog in JIRA Software that shows linked branches in the Software Configuration Manager (SCM). If JIRA Software has been linked to more than one SCM, a tab will show for each SCM application (e.g. Bitbucket Server). The branches will be grouped under each SCM in these tabs.

TIS-45: 2 branches			
Repository	Branch	Pull request	Action
 Apollo UI	feature/TIS-45-email-non-registered-users-to-sign	MERGED	Create pull request

2. Click a repository or branch to open the linked SCM at the relevant repository or branch.
3. Hover over a **Pull request** status (e.g. **MERGED**) to show a popup displaying a link to the pull request.
4. Click **Create pull request** to create a pull request for the branch (to merge it back to master).
5. Include the issue key in the title of the pull request. If you have workflow triggers configured, the status of the issue may automatically transition to 'In Review'. See [Configuring workflow triggers](#) for more

information.

Finish your work

After you work has been reviewed and merged with the master branch, the last step is to close or resolve the issue.

1. Locate your card in the **Active sprints** of your board.
2. Drag and drop the card to the **Done** column. If you have workflow triggers configured, the status of the issue may automatically transition to 'Closed'. You can skip the next steps. See [Configuring workflow triggers](#) for more information.
3. Select the **Resolution** for the issue from the drop-down menu.

4. Provide details as necessary.
5. Select **Close Issue**.

Great job! You have successfully gone through a simple workflow using JIRA Software!

To do more awesome stuff, like managing your issues and customizing JIRA Software, click the 'Advanced info' button below.

[Advanced info](#)

Learn to plan and estimate for scrum teams

Estimating and planning for agile projects can be really challenging regardless of how experienced you are. There's always competing priorities, resourcing issues, business demands and time constraints.

This topic can't make any of those things go away, but if you're a newish to agile scrum or just new to JIRA Software, it will give you an overview of the JIRA Software features to help you plan and estimate.

Learning objectives

The aim is to give you a good introduction to:

- Planning concepts and functions
- Estimating methodologies
- Analysis tools and techniques
- Optimization approaches

We've also asked some of our [Atlassian teams](#) to provide some tips and advice along the way.

Learning activities

There's optional activities outlined at the bottom of each topic. You don't

Learn more about agile

It helps if you're familiar with the agile language and concepts we'll use. So if this is a bit new to you or you feel a bit rusty, check out the [Agile Coach](#), which covers lots of the basics.

have to do these, but it does help to put the theory into practice. You might want to create a dummy project to do this.

Pre-requisites

You need:

- Access to JIRA Software are. If you don't have a working copy, trial it [here](#).
 - The ability to create projects and configure boards.
- If you are working with an existing instance of JIRA software, you need this access to do some activities. If you [create a dummy project](#), you will get this access by default.

[Let's go](#)

Plan for the team

1. Plan for the team
2. Customize the team board
3. Estimate in story points
4. Analyze team reports
5. Optimize future plans

They say the best laid plans always go awry, and sometimes they do, but you still have to make them. At least in JIRA Software, the plans you make are highly flexible, customizable, and configurable.

Start with a backlog

All agile projects start with a [backlog](#) of issues that represent the work that needs to be done: user stories, tasks, bug fixes, etc. Anything that's been determined is a discrete piece of work.

- Backlogs are all about priorities and doing the most important things first
- Use quick filters to just see the issues you want
- Create new issues and start sprints in the backlog

To get to your backlog, open your board and click **Backlog**. Control board details shown using the options in the Board menu.

You can add issues quickly by clicking **+ Create issue** below the backlog list.

The screenshot shows the JIRA Software Backlog view. At the top, there are quick filters for "Only My Issues" and "Recently Updated". On the left, there are navigation links for "VERSIONS", "EPICS", and "SPRINTS". The main area displays a list of 11 issues under the heading "Backlog 11 issues". Each issue is represented by a row with a small icon, the issue key (e.g., LRC-4, LRC-12, LRC-2, LRC-1, LRC-7), the title, the priority (V1.0), the status (Safety testing, Structural engineering, Cosmetics, Structural engineering, Glass and locks), and the assignee (2, 8, 5, 6, 7). A "Create Sprint" button is located at the top right of the backlog list.

"You're just setting yourself up for failure if you try to plan sprints too far ahead. You'll waste energy making decisions about things that will almost certainly change scope or priority in a few weeks. The backlog is always waiting to be re-prioritized"

~ Atlassian Team Lead

Use epics for context

Epic are designed to tell the bigger story. They can be used to capture broader work themes, such as 'performance' or 'user interface', or to track larger items of work. There are advantages to using [epics](#).

- Epics can span multiple projects, which is especially useful if your board is set up to show issues from multiple projects.
- You can link an epic to a Confluence page to provide more information, such as design specs, business case, requirements, etc.

To create an epic

1. In your board backlog, click **EPICS** on the left side of the board, then click **Create epic**.
2. Complete the details of the epic in the window that appears and click **Create**.
3. Drag and drop backlog issues across to the epic they belong to.

See [Working with epics](#) for more information.

"Versions work best for our long term planning because they align to our service-level agreements. Our customers expect weekly releases."

~ Atlassian Release Manager

Versions keep the release in order

Lots of teams release **versions** of their product as updates or feature sets for customers. Allocating issues to versions can help the team plan sprint work in the lead up to a release.

To create a version

1. Click **VERSIONS** on the left side of the board, then click **Create version**.
2. Complete the details and click **Create**.

You can then drag and drop backlog issues across to the version they belong to.

See [Configuring versions in a Scrum project](#) for more information.

Rank and prioritize

When your backlog issues are listed in order of priority, it's far easier to create sprints that are based on the work that needs to be done next.

Ranking is easily done by dragging and dropping issues in the backlog.

"Ranking helps us knock over sprint planning really quickly because we can all see what has to get done next."

~ Atlassian Dev Lead

The screenshot shows a JIRA backlog board titled "Backlog" with 11 issues. The issues are categorized into three columns: "Safety testing" (2 issues), "Structural engineering" (8 issues), and "Cosmetics" (5 issues). Each issue card includes a priority icon (e.g., orange triangle for LRC-4), a title like "Test drive", and a due date like "V1.0". A sidebar on the left shows navigation links for "VERSIONS", "EPICS", and "Backlog". A "Create Sprint" button is located at the top right.

Rinse and repeat

The best thing about scrum projects is that as you progress through sprints, the more clearly you can see what your team is capable of.

There's a heap of reports and charts that you can consult to help with understanding your team's productivity rates. We'll talk about how some of these can help you get better at planning a bit further along.

LEARNING ACTIVITY

Create a dummy project and choose **Scrum software development** as the project type.

1. Create a backlog of issues. You just need a title to add new ones.
2. Re-prioritize the backlog rank by dragging and dropping
3. Create an epic and add a few issues to it
4. Create a version and add a few issues to it

Next

Customize the team board

1. Plan for the team
2. Customize the team board
3. Estimate in story points
4. Analyze team reports
5. Optimize future plans

The **board** is where all the action happens so you want to make sure that it's set up the way your team needs. You can change a heap of settings in the **board configuration** screens. Here's just a few.

⚠ You need to have special Board Admin rights to change board configurations. If you created the board, you already have them.

Board customization

Control everything from the board configuration page. In your board, go to **Board > Configure**.

- **Columns**
Add statuses and change the workflow to suit your team structure. For instance if you want to show testing progress separate to dev progress, add a column. You can also switch on an indicator to help identify slow moving issues.
- **Card layout**
Show up to three extra fields on the card layout. Say, if you want to show labels or components.

The screenshot shows the 'Configure' section of a JIRA board. On the left, a sidebar lists various configuration options: General, Columns, Swimlanes, Quick Filters, Card colors (which is currently selected), Card layout, Estimation, Working days, and Issue Detail View. The main content area is titled 'Card colors' and contains the text: 'If your team is 'visual' you can color code cards according to issue type, assignee, priorities, or queries'. Below this is a link to 'See Configuring a board to learn about the other things you might need.'

Quick filters

Set up quick filters so you can just click once to show specific issues. You can set up a filter for issue types, assignees, unassigned, etc. Pretty much anything you need.

Go to **Board > Configure > Quick Filters**.

Filters work based on JQL (JIRA Query language) which are really simple search statements. The result is something like:

TIP: as you start typing in the JQL field, the syntax helper pops up to assist

The screenshot shows the 'Quick Filters' configuration screen. It has a table with columns: Name, JQL, and Description. There is one entry: 'Engineering' in the Name column, 'component = Engineering' in the JQL column, and 'All issues related to vehicle engineering' in the Description column. An 'Add' button is visible at the bottom right of the table.

▼ TIP: Create filters from saved searches

If you have a giant backlog, you probably use the [issue search](#) pretty regularly to find the issues you want. And if you're repeating searches using similar criteria, then other people probably are too.

Next time you are searching for a set of issues you know you will need again, save your search as a filter. Once you save a filter, you can share it, favorite it, get emails of search results, use it with a dashboard, and heaps more.

1. Go to **Issues > Search for issues**.
2. Enter your search filters and click **Save As**.

LEARNING ACTIVITY

1. Create a new quick filter and use it on your board.
2. Search for an issue and then save the search as a favorite.

"We just couldn't work as a global team without quick filters. The guys in San Fran and Gdansk need to be able to see exactly the same set of issues as us when we're discussing work online."

~ Atlassian Dev Manager

[Previous](#)

[Next](#)

Estimate in story points

1. Plan for the team
2. Customize the team board
3. Estimate in story

- points
- 4. Analyze team reports
- 5. Optimize future plans

There's a huge variety of ways to estimate stories, but the objective of every approach is to be able to get better at predicting how much work the team can do each [sprint](#). In agile scrum, this translates to knowing the team's [velocity](#).

Velocity measures the number of 'estimation units' that a team usually completes from sprint to sprint. It is effectively a productivity rate based on an estimation of volume of work, and it is best worked out in a measure other than 'time'.

▼ What are story points?

Story points are a commonly used measure for estimating the size of an issue in scrum teams. During a typical planning session, a trivial bug fix might be estimated as a 1 or 2, and a larger feature might be anything up to a 12. Note that the scale of points does vary between scrum teams. Some use 1-12, others 1-5. The key is to use the same scale so that your velocity is consistently calculated.

If issues are estimated larger than this, they might need to be broken into smaller stories or subtasks.

The focus is on story points here because it's the more common scrum estimation method, and we use it at Atlassian. Your company may use hours or ideal days. If you do use time for estimation, you might want to have a look at [Configuring estimation and tracking](#).

"Even seasoned devs can't estimate accurately in hours/days. We just need to predict what can realistically get done with a reasonable degree of certainty. Story points all the way."

~ Atlassian Developer

▼ Q: Why use story points instead of hours?

A: It is much more useful in the long term to roughly predict how much work your team can do each sprint, than it is to try to estimate how long each story will take in time.

Story points enable the team to estimate stories in comparison to other stories, instead of forcing them to determine the time it will take to complete each story. Velocity is then worked out based on how many points the team can complete in each sprint. After a few sprints, team velocity stabilizes and estimation accuracy increases (or rather the same degree of inaccuracy is applied).

The problem with using hours to estimate is that the team is likely to estimate using the 'working hours' of the sprint (with some buffer time) and this is almost never an accurate reflection of a story's complexity or size. Accurate velocity is difficult to establish when stories are estimated in isolation, instead of by comparison.

Estimation, time and velocity are really critical things to understand. Read more in the [Estimating an issue](#) topic.

Estimate in points, track in time

Even when you estimate in story points, you can still track in time if you want. Knowing the team velocity (regardless of the measure used) enables you to roughly guess how long estimated backlog items will take to complete.

But JIRA Software does also have a couple of dedicated fields (**Remaining Estimate** and **Time Spent**) to track time while using story points.

⚠ NOTE: You need to be a board admin to make any configuration changes to a board. If you created the board, then you're already the admin.

Go to **Board > Configure > Estimation**.

Estimation

Issues can be estimated when in the Backlog to get an idea of how much work is being committed to in a sprint. [Read more about estimation and tracking.](#)

Estimation Statistic 

Estimate issues in the Backlog by entering values for **Story Points**. Your velocity from sprint to sprint will be measured against these estimates.

Time Tracking **None**
Issues will burn down their **Story Points** value upon completion.
 Remaining Estimate and Time Spent
Track time against issues using JIRA's **Remaining Estimate** and **Time Spent** fields.

- Select the **Estimation Statistic** (unit of estimation) - choose from story points, original time estimate, and issue count.
- Switch on the **Remaining Estimate** and **Time Spent** option to get a more accurate picture of how things are tracking in time units.
- If you leave **Time Tracking** as **None**, you can still refer to reports such as the Burndown chart to monitor progress.

Find out more about how time tracking work in projects in the [configuring estimation and tracking](#) topic.

▼ **Fun fact: Inaccuracy is good**

The goal of velocity is to be able to look at a backlog of not particularly well-understood stories, and guess how many sprints it will take to complete. It requires a similar level of uncertainty for all of the estimates in the backlog. Determining accurate velocity relies on the equality of the inaccuracy, which is why you should NEVER re-estimate issues after a sprint has started.

LEARNING ACTIVITY

In your dummy project, go to **Board > Configure > Estimate** page and change the **Estimation statistic** to one of the following:

- Story points
- Original time
- Issue count

Note the differences when you create and estimate an issue.

[Previous](#)

[Next](#)

Analyze team reports

1. Plan for the team
2. Customize the team board
3. Estimate in story points
4. Analyze team reports
5. Optimize future plans

JIRA Software comes with a whole lot of [reports](#) that will help everyone in the team understand where things are at. Here are just a few.

Access them all from the left navigation pane.

Velocity for the team

Teams want to know what their velocity is. They want to know how accurately they are estimating and how much they are getting done each sprint, in order to estimate better in future.

Velocity Chart

- See if the team is over-committing or under-committing stories in sprint planning
- Check for outliers and drill into a sprint to learn from past mistakes (e.g. a grossly underestimated story)
- Monitor how velocity alters during periods of team change or growth

>Show Velocity Chart

Reports: Velocity Chart -

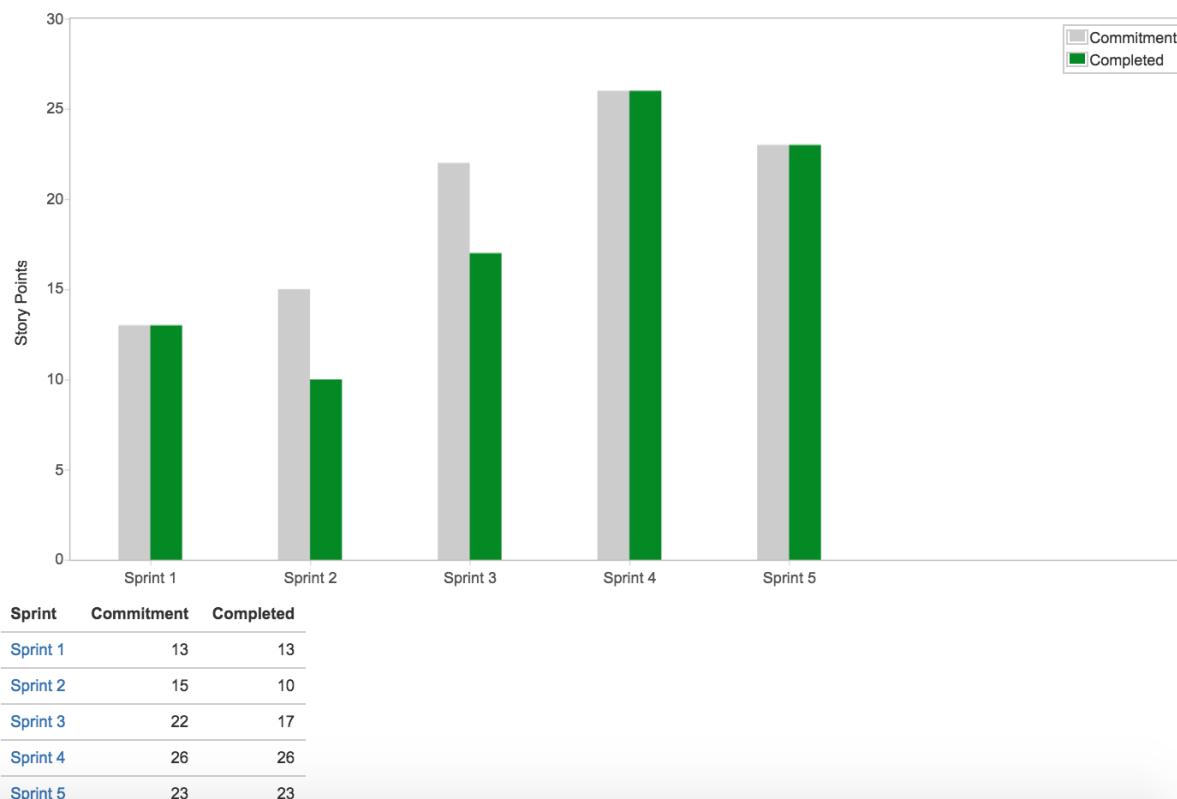
[Board](#) ▾


② How to read this chart

Track the amount of work completed from sprint to sprint. This helps you determine your team's velocity and estimate the work your team can realistically achieve in future sprints.

[Hide this information](#)

Velocity Chart



Burndown for the leads

Team leads (Dev leads, Tech leads, etc.) usually spend half their time thinking ahead to future work, and the other half firmly involved in the current sprint. They need to know how things are going now to help steer the team to a successful sprint completion.

Burndown Chart

- If the **Remaining Value** line it is headed straight across and not down, you need to find out why
- A successful sprint is where the **Remaining Values** line hits the

"Burndown is the fastest way for me to see what's going on and help out the team if things are looking a bit shaky."

~ Atlassian team lead

- bottom on or before the **Guideline**
- If the **Remaining Value** line ends before the guideline, you may have a bunch of overachievers or under-committers on your hands!

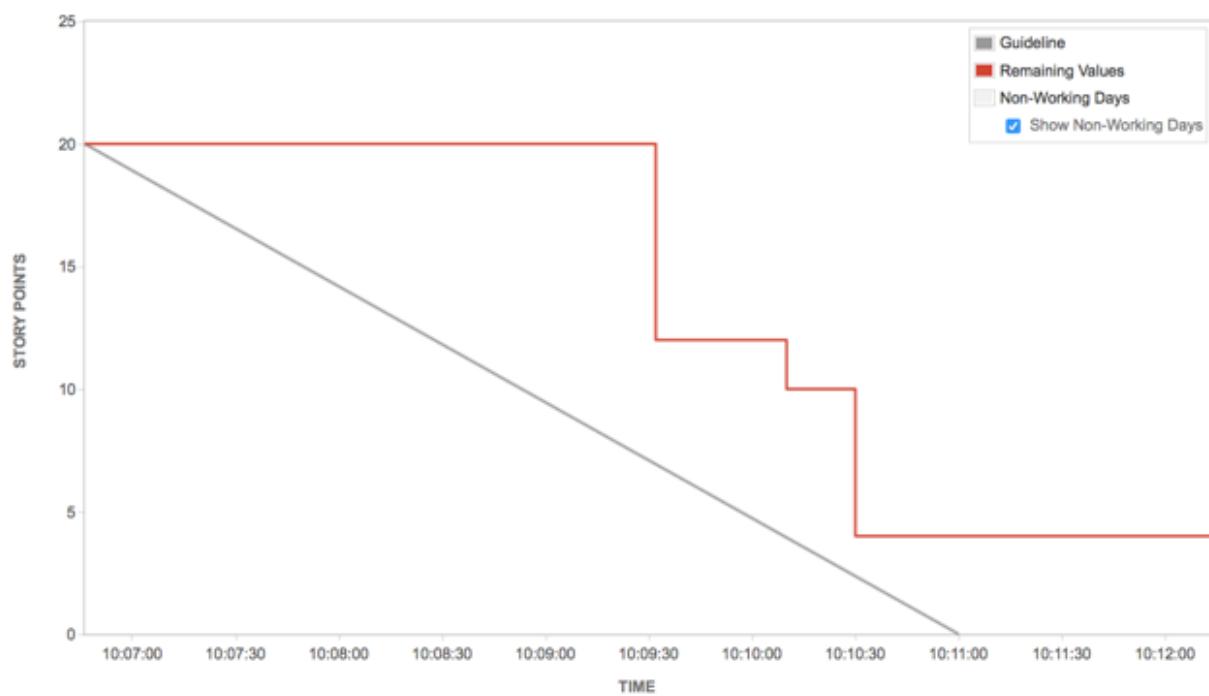
▼ Show Burndown Chart

Reports: Burndown Chart

Board ▾



Burndown Chart Spring Cleanup Story Points ⓘ How to read this chart



Story Points						
Date	Issue	Event Type	Event Detail	Inc.	Dec.	Remaining
01/Jul/15 10:06 AM	TIS-1 TIS-2 TIS-5 TIS-8	Sprint start		6 2 8 4		20
01/Jul/15 10:09 AM	TIS-5	Burndown	Issue completed		8	12
01/Jul/15 10:10 AM	TIS-2	Burndown	Issue completed		2	10
	TIS-1	Burndown	Issue completed		6	4

Epics for product managers

Product managers (or project managers, depending on your structure) like to keep an eye on the big picture stuff, so they can check if project milestones are being hit, where dev effort is being spent, and if the backlog is in need of a re-prioritization. The Epic Report is a good read for satisfying some of these demands.

Epic Report

- Easily see the how near or far an epic is from completion. Check with the team if things look unbalanced.
- Monitor unestimated issues. Chat to the dev lead to see if anything needs further clarification or breakdown.

>Show Epic Report

Epic Report -

Board ▾



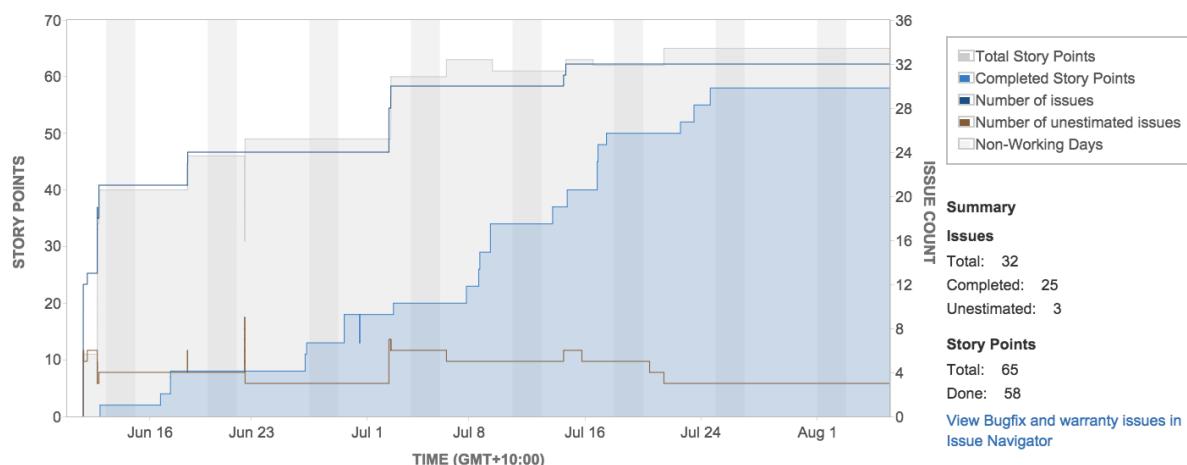
[How to read this chart](#)

Understand the progress towards completing an epic over time. This helps you manage your team's progress by tracking the remaining incomplete/unestimated work.

[Hide this information](#)

Bugfix and warranty issues ▾

TIS-100 Bugfix and warranty issues [Linked pages](#)



Status Report

Completed Issues

[View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (58)
TIS-81	Afterburner initial design	Bug	↑ Critical	RESOLVED	-
TIS-82	Afterburner initial prototype	Bug	↓ Minor	RESOLVED	-

LEARNING ACTIVITY

You need to have historical sprint data in your project to view the above reports. If you don't have access to an existing project, just browse the report help topics:

- Sprint Report
- Burndown Chart
- Epic Report

Previous

Next

Optimize future plans

1. Plan for the team
2. Customize the team board
3. Estimate in story points

4. Analyze team reports
5. Optimize future plans

When things are running well, velocity is stable, the team is feeling challenged and even the product manager is happy, it is not time to stop planning.

Plan ahead

Everything learned from burndowns, reports, story estimation, sprints and standups is ready and waiting to help make planning better for future sprints and issues. Even if the team changes, the budget is cut, or the company you work for insists on a whole new focus, the data collected is still really valuable.

Here's the top things you might change next time, based on what you learn this time:

▼ [Add more story points for high performing teams](#)

A high performing team that always completes a sprint with no work to spare may need to be challenged with more points in the next sprint. But be careful of tipping the scale too far in the other direction. Teams that rarely complete a sprint without leftover points may feel like they are underperforming, even if they aren't.

▼ [Change things up to get different results](#)

When things are going well, you don't want to mess with the formula. But when things aren't going as well, it can help to change things around a bit (if you have the luxury to do so). This could take the form of swapping responsibilities for a sprint, or introducing a different kind of sprint, such as an innovation sprint or bugfix only sprint. Change can be re-invigorating, and it can still contain work!

▼ [If overcommitment persists, please see your PM](#)

Failure to meet committed story points can be disappointing for a team, especially if they don't know why. Check in with the PM or person who reported the issue to clarify details during estimation, and raise concerns during retros. If estimates are way off due to lack of basic information, for example, this needs to be addressed.

"One of the best things about estimating as a team is that it sparks some really important discussions about how we build our product and how we can get better at what we do."

~ Atlassian Product Manager

Don't forget the invisible things

Scrum is all about team and a lot of things happen during sprints that need to be captured and fed into future plans. Here's a few things that new starters should know and experienced operators sometimes forget.

- **Always hold a retro. Always.**
Retrospectives can give you insights that reports can't. Make sure you discuss any issues that come up, which might result in adjustments to planning. Record insights (preferably in Confluence) and brag loudly about the good stuff.
- **Never assume the team knows how they are performing**
Just because velocity reports are available and are viewed in monthly team meetings, doesn't mean the team knows how they are going. Scrum leads and team leads need to use words too. Who doesn't like being told in person they are going well.
- **Say it with meaning even if you've said it before**
Standups and planning can become a bit routine over time, especially with experienced teams who have worked together for ages. Try not to let the rituals become lip service, make sure people are saying meaningful things and are listening to each other actively.

FURTHER LEARNING

If you want to learn more about estimating and planning, check out some of these options:

- Browse the JIRA Software documentation
- Take a few free classes at the [Atlassian University](#)
- Search and ask the Atlassian user community for [answers](#)
- See if [The Agile Coach](#) can help

- Search all Atlassian documentation

[Previous](#)

Installing JIRA Software

JIRA Software can be customized to suit your needs. You can also extend it and link it with other applications to provide you with the perfect solution to track all your work in one place.

Note, if you're upgrading from an existing JIRA instance, check out our [Migration Hub](#) first.

You can install JIRA Software Server anytime.

Before you start, read the [JIRA Software release notes](#) for the version that you are installing or upgrading to, then follow the instructions below.

1. View the Server products for JIRA [here](#).
2. Select the JIRA Software package and choose the number of users.
3. Download the installer.
4. Once the installer has downloaded, run it and follow the [Installing JIRA applications](#) steps.

If you experience any problems with your installation or you have any questions, contact [Support](#).

Get more out of your new JIRA Software instance

Connecting JIRA Software to other Atlassian products enables a host of new integration features. Read the pages below to learn more:

- [Using JIRA applications with Confluence](#) — Confluence is a content creation and collaboration platform that connects teams with the content, knowledge, and coworkers they need to get work done, faster.
- [Using JIRA applications with HipChat](#) — HipChat is hosted group chat and video chat for companies and teams.
- [Using JIRA applications with Portfolio for JIRA](#) — Portfolio for JIRA provides a single, accurate view for planning and managing initiatives across multiple teams and projects with ease.

Using JIRA applications with HipChat

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 issues and service desk requests directly in HipChat when someone on your team mentions them.

On this page:

- Connecting projects to rooms
- Invite users
- Discuss issues in rooms
- Issue preview
- Remove OAuth Permissions

Connecting projects to rooms

You can link JIRA projects with one or more HipChat rooms so that when issues are updated or created, messages are sent to the HipChat rooms that you specify.

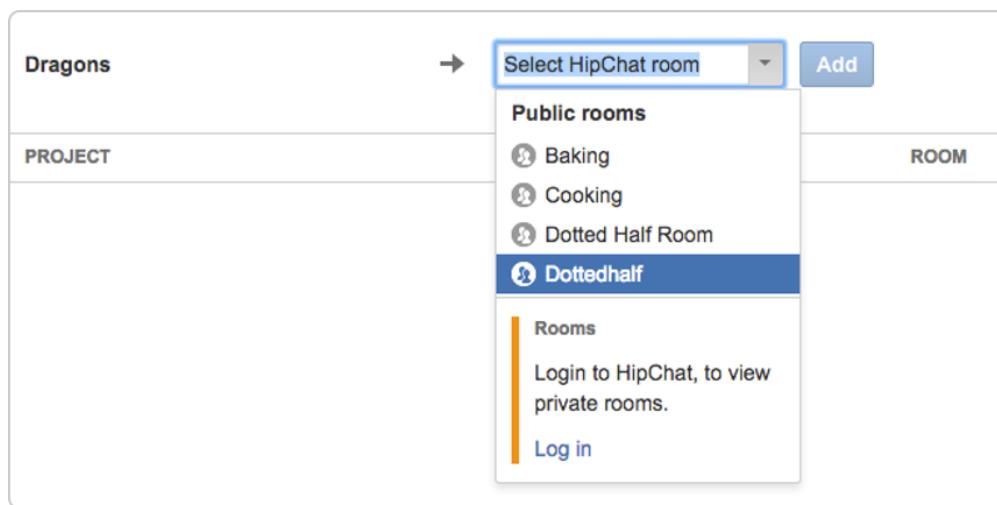
1. You must be a logged in as an Administrator or a Project Administrator.

2. Choose  > **Projects**.
3. Select a project.
4. In the Project settings menu, select **HipChat Integration**.
5. Choose a HipChat room and select **Add**.
6. Select the Issue Type, Priority, or select **Advanced** to enter a JIRA [JQL Query](#).
7. Select the actions that will send a notification to your room (issue created, assignee changed, new comments, and issue transitions).
8. Select to notify users (using HipChat notifications) when a message is sent to the room.
9. Changes are saved automatically, continue browsing your project to continue.

Notify Users in This Room uses HipChat notifications (playing a sound, popups, and bouncing dock icon) to alert users of new messages sent from JIRA. This functionality is only available in the web and IOS clients.

Private rooms

Private rooms in HipChat are by invitation only. In order to connect JIRA to a private room in HipChat you will need to authorize HipChat from the **HipChat Integration** setup screen.



Once you have authorized JIRA, all of the private rooms that you are a member of will be displayed in the room selector drop-down menu. When your JIRA project and room are integrated, everyone in the private room will be able to see the notifications that are sent to that room.

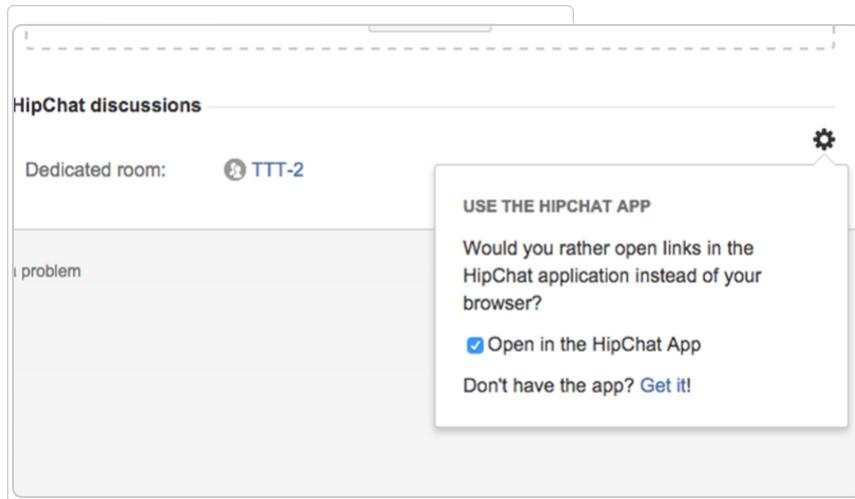
Invite users

If you have administrator permissions, you can invite users to join HipChat directly from the Integration screen. Follow the instructions in [Linking JIRA and your HipChat site](#) above, to access the integration screen. You must have at least one project integrated with a room to see the invite users link. Select the link to send an email inviting users to HipChat. To invite users, you will need to confirm access to your HipChat account to give JIRA permission to invite users.

You can remove this access by following the instructions in [Removing OAuth Permissions](#).

Discuss issues in rooms

You can focus your discussion by creating or selecting a HipChat room to discuss an issue. When JIRA is integrated with HipChat and you are in the issue screen, you can select to "Create a room" or "Choose a room" in the **HipChat discussions** panel. This will associate the current issue and the room and any changes to the issue will send a notification to that room.



You can also select to have links open in your HipChat App (OSX only) when you select a link. In the issues screen, select the cog icon in the HipChat discussion to enable opening links in the application.

Issue preview

With issue preview enabled, if you enter an issue key as part of a message, or paste a URL for the issue in any room in HipChat, you can receive a preview of the issue. This way, the entire room can see and be on the same page when discussing an issue, without ever having to leave the discussion.

Joe Wong hey can anyone pick this up? PROJ-3

JIRA PROJ-3 : Issue preview sample from JIRA Created by admin
 Type: Task Priority: Major Status: To Do
 Assignee: admin

Connectivity requirements for JIRA and/or HipChat Server customers

For this feature to work, HipChat needs to be able to talk to JIRA, which means that your JIRA instance must be addressable and accept inbound connections via HTTPS.

A note on JIRA permissions

If this feature is enabled for a project, a preview will be posted in HipChat for any issue key/URL for that project. If a project contains sensitive information you don't want shared in HipChat, make sure to disable this feature for this project.

Configuring issue previews

If you are logged in as a JIRA Administrator, you can enable or disable issue preview for all projects. A Project Admin can also override issue preview by individually enabling or disabling this setting for each project.

As a JIRA Administrator

1. Log in as a user with the **JIRA Administrators** global permission.
2. Choose > **Applications > HipChat**.
3. Select **Advanced Settings**.

4. Select the checkbox to enable or disable the Issue Preview globally.
5. Select **Save** to exit.

As a Project Administrator

1. You must be a logged in as a **Project Administrator**.
2. Choose  **Projects**.
3. Select a project.
4. In the Project settings menu, select **HipChat Integration**.
5. Select **Advanced Settings**.
6. Select the checkbox to enable or disable Issue Preview for your current room.
7. Select **Save** to exit.

Remove OAuth Permissions

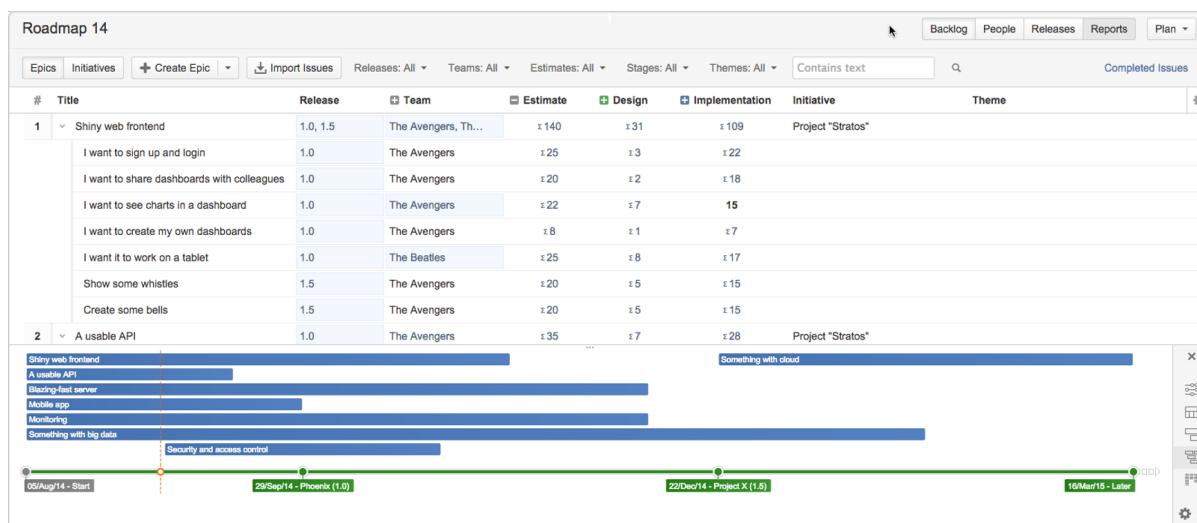
You can remove permissions that you have granted to allow JIRA to access HipChat. For instance, if you have given JIRA permission to invite users on HipChat's behalf.

1. Select your avatar to access your profile.
2. Click **Profile**.
3. Select **Tools**.
4. Click **HipChat OAuth Sessions**.
5. Select **Remove Access**.

Using JIRA applications with Portfolio for JIRA

Portfolio for JIRA provides a single, accurate view for planning and managing initiatives across multiple teams and projects with ease. See [our guide](#) to how JIRA and Portfolio for JIRA work together.

- Plan top-level business initiatives and break them down into lower level deliverables for development teams
- Track investments across strategic themes to ensure that those investments align with business priorities
- Generate realistic delivery forecasts with automatic scheduling algorithms
- View the progress of any initiative based on real-time, accurate data from JIRA
- Drive accurate and realistic capacity planning by defining teams and allocating work based on skills and availability
- Scope releases in a matter of clicks
- Make fast prioritization and tradeoff decisions to instantly see the impact of plan changes
- Minimize productivity loss by easily reacting to the ever-changing needs of your business in real time



Plan automatically

- Set priorities, estimates, and target dates to instantly see when you can ship releases based on your

commitments

- Automatically optimize your plan and get suggestions on ideal resource allocation to create a realistic forecast
- Account for dependencies, resources, parallel vs. sequential activities and the realistic number of people that can work on a single item to create an optimized roadmap
- Use themes to categorize your backlog items by strategic focus areas, value streams, or investment categories to see relative resource allocation between themes

Epics Initiatives + Create Initiative Import Issues Releases: ... Teams: All Estimates:... Stages: All Themes: All Contains text							
#	Title	Release	Team	Members	Estimate	Design	UI/UX Design
8	Something with cloud	Later	The Avengers, Th...	Black Widow, George, John	± 20	± 20	15
▼	Go Mobile!	1.0, Later	The Beatles, The ...	Bodie, Cowley, Doyle, George	± 85	± 30	± 15
6	Mobile app	1.0, Later	The Beatles, The ...	Bodie, Cowley, Doyle, George	± 85	± 30	± 15
	Explore the basics	details ×	The Beatles		± 50	± 25	10
	1 missing skill in team 'The Beatles'. Technical Design Enable Social sharing of smart things	Later	The Professionals	Bodie, Cowley, Doyle	± 22.5	± 2.5	2.5
		1.0	The Beatles	George	± 12.5	± 2.5	2.5
▼	Project "Stratos"	1.0, Later	The Avengers, Th...	Black Widow, Cowley, George...	± 142	± 37	± 12.5
5	Security and access control	1.0	The Avengers, Th...	Hawkeye, Hulk, John	± 10	± 2.5	2.5
4	A usable API	1.0, Later	The Avengers, Th...	Black Widow, Hawkeye, John	± 22	± 7	
	REST API - Allow to submit data	1.0	The Avengers	Black Widow, Hawkeye	± 7	± 2	

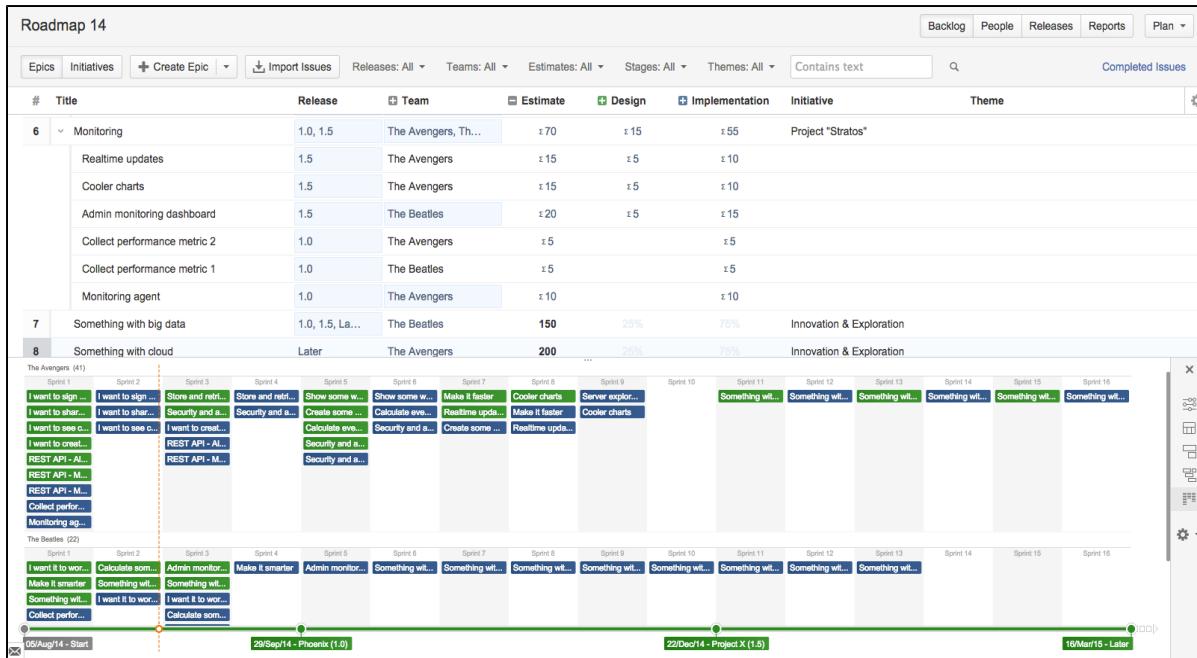
Avoid bottlenecks

- Identify and avoid bottlenecks and potential holdups by accounting for dependencies across teams and projects
- Model skills and define who can do which type of work to avoid unrealistic resource loads
- Automatically account for team member availability, including time off, training and inter-team commitments

Roadmap 2014											Plan	
		Weekly Hours	Design	UI/UX Design	Technical Design	Implementation	WEB	MOBILE	Backend	Database Stuff	Testing	
> The Avengers		Ø 32.5	3	3	3	5	3	3	2	1	3	
▼ The Professionals		Ø 40	1	1	1	3	1	3	1	1	3	
Doyle		40	✓	✓	✓	✓	✓	✓			✓	
Cowley		40				✓		✓			✓	
Bodie		40				✓		✓	✓	✓	✓	
▼ The Beatles		Ø 35	3	2	1	4	3	3	3	2	3	
George		40	✓	✓		✓		✓				
John		40	✓	✓		✓	✓	✓	✓	✓	✓	
Paul		40	✓		✓	✓	✓	✓	✓	✓	✓	
Ringo		20				✓	✓	✓	✓		✓	
...												
Phoenix (1.0) 14/Sep/14 Dynamic end date		Project X (1.5) 28/Sep/14 Fixed end date at 06/Oct/14										
10%	11d	97d	18%	20d	89d							
Free Capacities Testing, MOBILE, WEB												

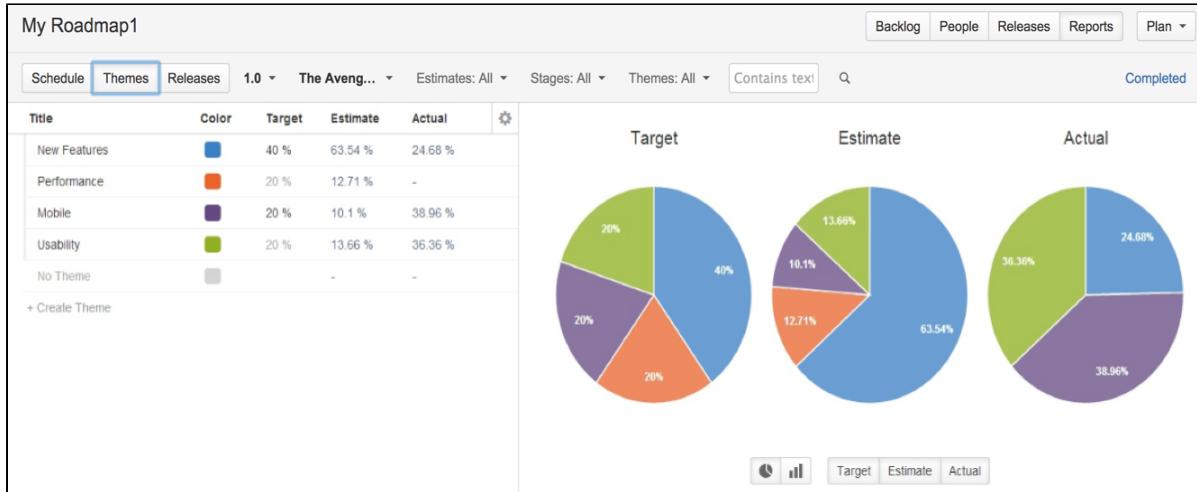
Keep up to date

- Keep the long term plan in sync with reality by adjusting delivery dates, team member resources and dependencies using up to date data
- Changes are displayed in real-time across all projects, giving you a comprehensive view of your entire roadmap
- Having an up-to-date schedule gives your team a transparent understanding of what's next, lending clarity to your decision-making



Make realistic commitments

- Confidently make commitments for scope and ship dates by using your more reliable forecasts
- Adjust dependencies to quickly check the impact across all of your projects, for example, if a critical feature is taking longer than expected



Model changes

- Quickly visualize what different scenarios and decisions mean to your projects
- Evaluate new change requests by seeing the impact on scope, dates, resources and cost commitments
- Model different scenarios without affecting the underlying data in your JIRA projects

The screenshot shows a 'Create Team' dialog. On the left, a sidebar lists teams: 'The Avengers', 'The Professionals', and 'The Beatles'. Under 'The Beatles', members 'George', 'John', 'Paul', and 'Ringo' are listed. A '+ Create Team' button is at the bottom. On the right, the 'George' team details are shown. It includes tabs for 'General Availability' and 'Availability in Team'. The 'Presence' section shows an interval from 02/Jun/14 to 'Unlimited' with the description 'George joins us June 2nd.' The 'Absence' section shows intervals from 09/Jun/14 to 13/Jun/14 ('Onboarding') and from 13/Oct/14 to 17/Oct/14 ('Vacation'). Buttons for 'Add Interval' and a gear icon are visible.

You can find more information about managing your portfolio here: [Portfolio for JIRA](#).

Using JIRA applications with Confluence

What is Confluence?

Confluence is a content creation and collaboration platform that connects teams with the content, knowledge, and coworkers they need to get work done, faster. Confluence spaces are great for creating and organizing rich content related to JIRA projects using Confluence pages – meeting notes, project plans, requirements documents, release notes, roadmaps, and more.

The screenshot shows the 'Development Home' page in Confluence. The left sidebar has links for 'Pages', 'Blog', 'SPACE SHORTCUTS' (IRKD 1.0 Release, IRKD Roadmap, Requirements, IRKD Releases, Meeting Notes, Shared Files, JIRA Backlog, Agile Board), 'Space tools', and 'Configure sidebar'. The main content area shows the 'Development Home' page with a 'Recently Updated' section listing changes by 'Jerry' and 'Samantha'. Below this is a calendar for February 2013. On the right, there's a 'Team' section with profiles for Matt (Engineering), Tim (Product Management), Jessie (Marketing), Samantha (Engineering), Sarah (Documentation), and Jens (QA). At the bottom, there's a 'Like' button and a note 'Be the first to like this'.

Why use Confluence with JIRA?

Here are some of the reasons we think you might like to add Confluence to your JIRA instance:

For a...	You can...
Bug	Create a knowledge base article to document a workaround for a bug.
New Feature	Create a product requirements document for a new feature.
General JIRA Use Case	Document and collaborate with your team on an issue in Confluence.

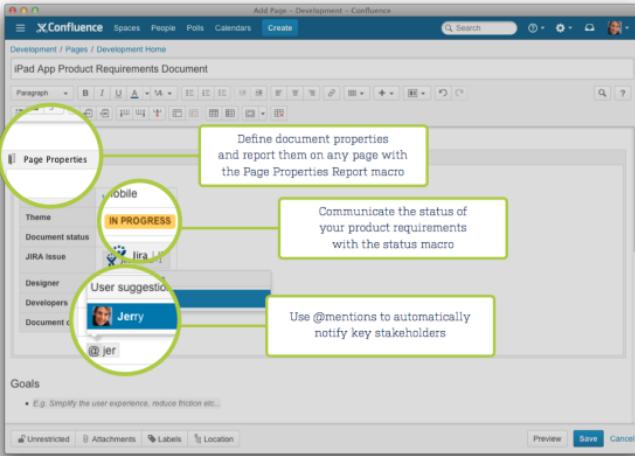
And here are just a few of the things Confluence allows you to do:

- Collaborative commenting, especially through the use of @mentions
- Share pages
- Watch pages
- 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

Confluence features for JIRA users

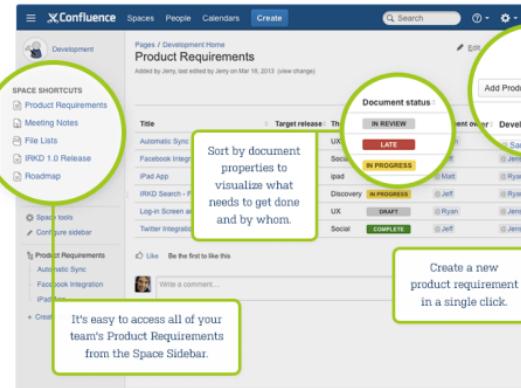
Here are some of the best features of Confluence that JIRA users would benefit from.

Define product requirements



Many of our customers write product requirements documents using Confluence to plan new products. The Product Requirements Blueprint helps development teams collaboratively create, discuss, and organize product requirements. It's easy to link your product requirements in Confluence to issues in JIRA.

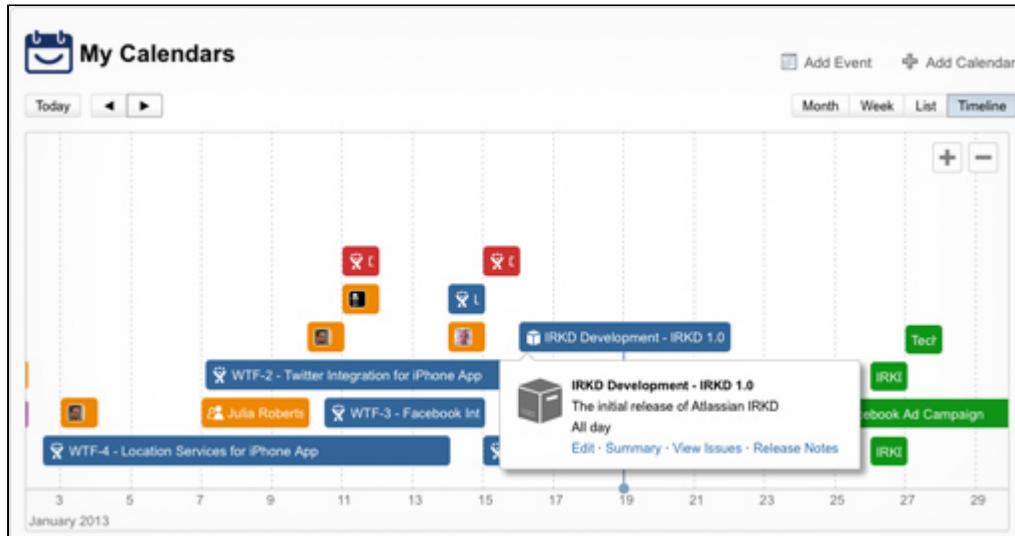
For more information, see [Blueprints](#).



Team Calendars: Your Birds-Eye View of JIRA

Surface everything your development team is working on in JIRA to the teams that live in Confluence with Team Calendars.

- **Timeline Calendar:** View plans 3 months ahead of time.
- **JQL Support:** Track your versions, issues, and agile sprints.
- **Date Ranges:** Visualize issues over time to understand upcoming workload.



To install this feature, please visit [Atlassian Marketplace](#).

Insert issues on any Confluence page using the JIRA Issues macro

Any JIRA search result can be embedded in a Confluence page using the JIRA Issues macro with your choice of included fields and field ordering. With the JIRA Issues macro, you can:

- Display a table of issues on your page, based on the results of a search using [JIRA Query Language \(JQL\) syntax](#) or a JIRA URL.
- Display a single issue from the JIRA site, or a subset of selected issues from your JIRA search results.
- Display a count of issues from the JIRA site.
- Create a new issue on the JIRA site and display that issue on your page.

Insert JIRA Issue

Search

[Create New Issue](#)

[Recently Viewed](#)

<input checked="" type="checkbox"/> Key	Summary
<input checked="" type="checkbox"/> S CONFDEV-2362	Implement "What's New"
<input checked="" type="checkbox"/> C CONFDEV-508	Add new shortcuts to RTE
<input checked="" type="checkbox"/> R CONFDEV-3493	Existing table header row styling has been removed
<input checked="" type="checkbox"/> R CONFDEV-1982	Migration has lost the table header styling
<input checked="" type="checkbox"/> I CONFDEV-11694	Unable to save or preview page with the "page index" macro on it

Display options

Display as Total issue count
Display total number of issues as a link. E.g. 185 issues

Table
Customise your columns below.

Columns to display

Hint: type "Cmd+Shift+J" in the editor to quickly access this dialog.

Autoconvert pasted issue links

Autoconvert makes producing reports of issues, backlogs, and tasks as easy as copy and paste. With JIRA and Confluence connected, you can paste individual issues or JIRA query URLs into the editor and watch them immediately transform into the JIRA Issues macro.

Automatic links

Whenever an issue is mentioned in a Confluence page using the JIRA Issues macro, JIRA will create an issue link to that page for you, automatically. [Specs to issues](#), [knowledge base articles](#) to support tickets, project outlines to tasks – it all works.

Gadgets

You can embed a Confluence activity stream or a Confluence page in JIRA's dashboard. Likewise, JIRA gadgets can be rendered on a Confluence page. See these topics for information on how to set up these gadgets:

- [Add a Confluence Gadget to a JIRA Dashboard](#)
- [Add JIRA Gadgets to a Confluence Page](#)

JIRA Software overview

Welcome to JIRA Software! This chapter gives you a rundown of the awesome features of JIRA Software, as well as how JIRA Software fits in with the JIRA family of applications.

What is JIRA Software?

JIRA Software is a powerful platform that combines issue collection and agile project management capabilities into a single application. Using JIRA Software helps you plan and organize tasks, workflows, and reports for your agile team more efficiently.

What is JIRA Admin?

The powerful issue collection and agile project management capabilities of JIRA Software are built on top of JIRA Admin. JIRA Admin also contains the administrative functionalities of JIRA Software. Many application settings are configured in JIRA Admin, like creating and editing workflows, or configuring permissions. Whenever you are asked to perform functions like these, which require administrative access to JIRA Admin, we will direct you to the JIRA administration documentation to accomplish these tasks.

Using this documentation

This documentation will teach you how to set up, configure, and start using JIRA Software. We've conveniently divided the documentation into chapters based on typical roles, and the tasks normally associated with those roles.

Leading an agile project

If you are a product manager, product owner, development manager, Scrum master, team lead, or if you hold any role that requires agile project management, then this chapter is for you. The topics in this chapter will help you configure your existing JIRA Software instance to suit your agile development process — from starting a new project and building a backlog to releasing a version and preparing reports for your team.

[Learn more...](#)

Working as a team member in an agile project

If you are a developer, designer, QA engineer, technical writer, or anyone else who would use (but wouldn't configure) JIRA Software, head on to this chapter. The topics in this chapter will help you use JIRA Software to manage your development work — from finding and working on your issues to creating branches for your work and collaborating with your team on issues in JIRA Software.

[Learn more...](#)

Administering JIRA Software

For system administrators who will set up, configure, and maintain instances of JIRA Software. The topics in this chapter will help you administer JIRA Software for your agile team — from configuring permissions to managing your JIRA Software settings and data.

[Learn more...](#)

JIRA Software is a highly customizable application, so your organization's instance of JIRA Software may appear different than shown here. In addition, different features and functionality are available to users depending on their permissions. In these instances, we will highlight the permissions required to complete a task.

JIRA applications overview

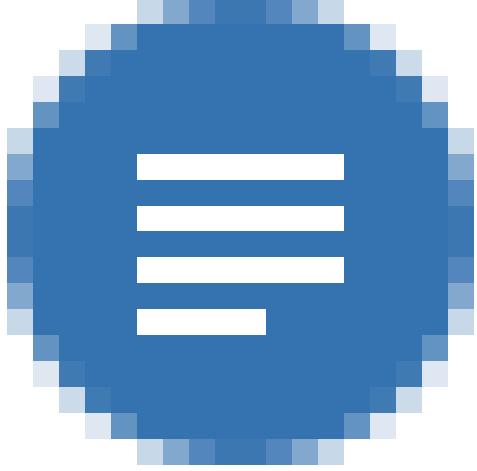
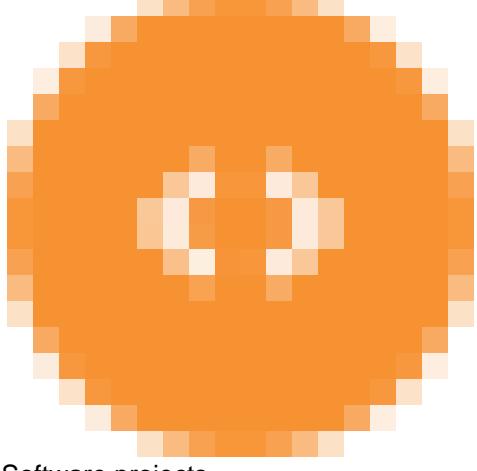
JIRA application overview

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

If you're a JIRA administrator, check out more information on [Licensing and application access](#).

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	<ul style="list-style-type: none">Available to all licensed users
JIRA Software	 Software projects	<ul style="list-style-type: none">Integration with development toolsAgile boardsRelease hub for software versions

JIRA Service Desk		<ul style="list-style-type: none"> • Service Level Agreements (SLAs) • A customizable web portal for customers • Permission schemes allowing customer access
	Service Desk projects	

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 tools, such as Bitbucket and FishEye, as well as 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 Software-specific features, like agile boards or the information from linked development tools. Likewise, a JIRA Software user would not be able to see any JIRA Service Desk application-specific features on a Service Desk project, only a basic view of the project and its issues.

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

A list of the applications, their user roles, and their project's application-specific features can be found below:

		JIRA Core	JIRA Software	JIRA Service Desk	
			JIRA-Core-user	JIRA-Software-user	JIRA-ServiceDesk-agent
 Business Projects	Project level	Create	✓	✓	✓
		View	✓	✓	✓
	Issue level	Create	✓	✓	✓
		View	✓	✓	✓
		Comment	✓	✓	✓
		Transition	✓	✓	✓
	JIRA Gadgets	View	✓	✓	✓
 Software Projects	Project level	Create	✗	✓	✗
		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	Create	✗	✗	✓
		View	✓	✓	✓
	Issue level	Create	✗	✗	✓
		View	✓	✓	✓
		Comment	✓	✓	✓
		Transition	✗	✗	✓
	SLA level	Create	✗	✗	✓
		View	✗	✗	✓
	Queue level	Create	✗	✗	✓
		View	✗	✗	✓
	JIRA Service Desk gadgets	View	✗	✗	✓

Leading an agile project

You may be the iteration manager, product owner, agile coach, or team lead for your team — if you are responsible for running an agile project with JIRA Software, you are in the right place! The topics in this chapter will help you configure your existing JIRA Software instance to suit your agile development process.

In the wrong place? See the following chapters instead:

- If you are a team member working with JIRA Software, see [Working as a team member in an agile project](#).
- If you are an administrator installing or configuring the JIRA Software server, see [Administering JIRA Software](#).

Where do I start?

- If this is the first time you're using JIRA Software, read our [Getting Started guide](#) before you start reading other topics.
- If you're already familiar with JIRA Software, start with [Starting a new project](#) to set up your project, or

use the search box below to find the topic you need.

Search the topics in 'Leading an agile project':

Overview

The topics in this chapter cover the activities that you will need to do in each stage of the agile development lifecycle:

Starting a new project

Time to get everything set up! This is the stage where you create a new JIRA Software project, configure your development tools (e.g. create repositories in Bitbucket, create projects in Bamboo, etc), and determine how you will manage documentation and team calendars.

[Learn more...](#)

Building a backlog

A backlog is a list of outstanding work. Before starting new work, you will build a new backlog or groom an existing backlog. This involves creating, ranking, and estimating issues, so that you have a prioritized list of tasks to work from.

[Learn more...](#)

Planning a version

Ready to start work on a new version? To plan a version, you need to set up the version in JIRA Software and configure your development tools appropriately (e.g. branch your repository, set up Bamboo plans, etc).

[Learn more...](#)

Getting to work

All systems are go! Once work begins, you will be running sprints and tracking your team's progress. For teams using Scrum, this involves planning and starting sprints, monitoring sprints via boards and charts, and completing sprints. For teams using Kanban, this involves monitoring the work in progress, and limiting work to the constraints for your team, via boards and charts.

[Learn more...](#)

Releasing a version

This is the culmination of your team's hard work. Before releasing a version, you would typically check if there is outstanding work for the version. If everything's good to go, you can then deploy the version, and create the related release reports and documentation.

[Learn more...](#)

Reporting

Reporting is an activity that occurs throughout a development cycle. You may need to make reports available to different stakeholders or use them yourself. JIRA Software has a number of reports that you can use to provide different information about the progress of your team's work.

[Learn more...](#)

Starting a new project

It's time to start your project! You know what you need to do, and you have a team that is excited to start working. You want to get everyone off to the best start, so let's make sure that your systems are set up and ready to go.

The documentation in this section will help you configure JIRA Software for a new agile development project. This includes creating a new JIRA Software project, configuring a board (Scrum/Kanban), configuring your development tools to work with JIRA Software, and more.

Search the topics in 'Starting a new project':

Overview

Note: it is assumed that you already have a JIRA Software instance. If you don't have JIRA Software, install JIRA Software Server ([Installing JIRA applications](#)) or sign up for a JIRA Software [Cloud site account](#).

Creating or configuring a JIRA Software project

A JIRA Software project is simply a collection of issues (stories, bugs, tasks, etc). You would typically use a project to represent the development work in JIRA Software. This could be the work related to a product, a team, etc.

Learn more: [Configuring a project](#)

Setting up your repositories, build plans, and review projects

The start of a project is where you would set up new repositories in Bitbucket or FishEye, create build plans in Bamboo, and perhaps create new review projects in Crucible (if you are not using Bitbucket). If you're using alternative tools, consult your vendor's documentation.

Learn more: [Create a repository \(Bitbucket Cloud\)](#), [Creating repositories \(Bitbucket Server\)](#), [Configuring plans \(Bamboo\)](#), [Creating a project \(Crucible\)](#)

Connecting your development tools to JIRA Software

JIRA Software can be connected to a range of development tools to help you keep your project tracking in sync with your development work. For example, if you have Bitbucket connected to JIRA Software, you can configure issues to automatically transition when commits are made.

Learn more: [Configuring development tools](#)

Connecting your collaboration tools to JIRA Software

Every software project requires collaboration, whether it be via chat rooms, online documentation, or team calendars. If you have [HipChat](#) (chat), [Confluence](#) (wiki) and/or [Team Calendars for Confluence](#) (calendars), you can connect them to JIRA Software to help you plan and run your project more efficiently. For example, you could get issue notifications in your chat rooms, link specifications to epics or retrospective notes to sprints, include team events on your project calendar, and more.

Learn more: [Configuring collaboration tools](#)

Already set up your project?

Next: Start [building a backlog](#)



Configuring a project

Your issue tracker should be the hub of your development project. When properly configured, JIRA Software helps your team members prioritize and organize work better, so they can spend more time developing great software instead of wrangling issues.

A project is simply a collection of issues (stories, bugs, tasks, etc). You would typically use a project to represent the development work for a product, project, or service in JIRA Software.

Scrum or Kanban? Scrum has an iteration-based approach, which is generally a good fit for teams developing products, particularly if your team is releasing new versions on a regular schedule. Kanban is better suited for a continuous flow of work (e.g. service-oriented teams), where its constraint-based approach helps prevent your team from being overloaded.

Before you begin

Ideally, you should be a **JIRA administrator** (i.e. someone with the 'JIRA Administrators' global permission), if you want to set up a new project in JIRA Software. Many of the tasks involved can only be performed by a JIRA administrator, such as creating a project, modifying a workflow, etc. For more information, see [Permissions overview](#).

Overview

The topics in this section cover the activities that you will need to do when configuring a project:

Create a project

Your first task is to create a Software project



and configure it according to the needs and requirements of your development team. You need to be a JIRA administrator to create a project, but you can configure most project details (e.g. name, avatar, etc) if you are a project administrator.

Learn more: [Defining a project \(JIRA Admin documentation\)](#)

Configure the board for your project

A board is the central tool for working with issues in a Scrum or Kanban development project. A Scrum or Kanban board will be created when you create a Scrum or Kanban development project respectively. You can add more boards (of either type) to the project, if you like.

Learn more: [Configuring a board](#)

Configure project permissions

You can control access to your project by configuring a permission scheme. A permission scheme maps users, user groups, roles, etc to the project functions (e.g. Assign issues). Your new project will be pre-configured with the default permission scheme.

Learn more: [Managing project role memberships](#)

Tweak issue types, workflow, screens, and fields

Your project is pre-configured with issue types, a workflow, screens, and fields. You can change these to suit any development process. For example, you may want to create a 'Design' issue type or add a 'QA review' step to your workflow.

Learn more: [Defining issue type field values, Workflows, Defining a screen, JIRA custom fields \(JIRA Admin documentation\)](#)

Create components

Components can be used to group issues in a project. However, in an agile project, you should consider using epics to group issues (stories) instead, as there are more features that support working with epics, like the epic column on boards, reports, etc.

Learn more: [Managing components](#)

Creating a board

You need a board so that you can view and work on issues in JIRA Software. A board displays issues from one or more projects. You can either copy a board that someone else has created, or create a new board for yourself. You can create as many boards as you like.

There are two types of boards:

- **Scrum boards** are for teams that plan their work in sprints.
- **Kanban boards** are for teams that focus on managing and constraining their work-in-progress. Because work is not planned in advance using discrete time periods or versions, Kanban boards do not have a Backlog screen.

Learn more: [What is a board?](#)

On this page:

- Before you begin
- Creating a new board
- Copying an existing board
- Example JQL queries for board filters
- Next steps

Before you begin

Any user can create a board, but certain permissions are required to share the board with other users:

- If you create a board via **Boards** (in header) > **View All Boards**, you will not be able to share it, unless you have the 'Create Shared Objects' global permission.
- If you create a board via the following methods, you do not need the 'Create Shared Objects' global permission to share the board:
 - Creating a project (where a board is created for the project by default)
 - Setting up JIRA Software for the first time (where you're prompted to create a project, which also creates a board for the project)
 - Copying a board (the copied board will be shared with the same users as the original board)

Creating a new board

1. Log in to JIRA Software.
2. If you're setting up JIRA Software for the first time, do either of the following actions. Otherwise, please continue.
 - *If you're creating either a **Scrum software development project** or a **Kanban software development project**, follow the prompts to create your project. A board is created for your project by default.*
 - *If you're creating a **Basic software development project**, follow the prompts to create your project. After your project is created, click **Create board** under your project name, and follow the prompts to create your board.*
3. Select **Boards** > **View All Boards** from the top navigation bar.
4. Click **Create board** at the top-right of the page, and choose whether to create a Scrum board or Kanban board, as described below. Note that you cannot change the board type after creation (that is, a Scrum board cannot become a Kanban board, and vice versa).

▼ To create a Scrum board based on projects

- a. Click **Create a Scrum board**, then choose whether to base your board on a new Software project ('Browse Projects' permission required) or an existing project.

*Note, you can also choose to **Create a Scrum board with sample data** ('Browse Projects' permission required). This will base your board on a new project pre-populated with sample data.*

- b. Follow the prompts to set up your board and project (if you are basing your board on a new Software project). This will create a pre-configured Scrum board containing all the issues in your chosen project(s). Your Scrum board will have an issue filter with the following query:

```
project = "[YOUR PROJECT(S)]" ORDER BY Rank ASC
```

▼ To create a Kanban board based on projects

- a. Click **Create a Kanban board**, then choose whether to base your board on a new Software project ('Browse Projects' permission required) or an existing project.

*Note, you can also choose to **Create a Kanban board with sample data** ('Browse Projects' permission required). This will base your board on a new project pre-populated with sample data.*

- b. Follow the prompts to set up your board and project (if you are basing your board on a new Software project). This will create a pre-configured Kanban board containing all the issues in your chosen project(s) that do not belong to a released version. Your Kanban board will have an issue filter with the following query:

```
project = "[YOUR PROJECT(S)]" AND (fixVersion in
unreleasedVersions() OR fixVersion is EMPTY) ORDER BY Rank ASC
```

▼ To create a Scrum or Kanban board that is based on a filter

Before you begin: You must have access to at least one saved issue filter (either your own filter, or one that someone else has shared with you). If you don't, first create and save a new issue filter (see the documentation on [issue filters](#) and [JQL](#) if you need help).

Click **Create a Kanban board** and base your board on an 'existing Saved Filter'. Follow the prompts to set up your board. Your new board will be available to all users who have access to your chosen filter. See the [examples](#) below for sample JQL for your board's filter.

Note, if you are the owner of the issue filter, you can edit it via the **Edit Filter** link. Otherwise, this link will not appear.

5. Your new board will be shown. At the top is a link that you may want to send to other people so that they can use your board.

Congratulations — you have created a new board!

Copying an existing board

If you would like to create a board that is similar to one you are already using, you can simply create a copy.

Your new board will be based on the same [issue filter](#) as the original board. You will be the administrator of the new board, but not necessarily of the filter, so you may not be able to edit the filter. However, once your new board is created, you can easily choose a different filter (e.g. to view a different project) — see [Configuring filters](#).

1. Select **Boards > View All Boards** from the top navigation bar.
2. The **All boards** screen will be displayed. Click the **Copy** link corresponding to the board of interest.

Alternatively, to quickly create a copy of the board you are currently viewing, simply select **Copy** from the **Board** menu at the top right of the page.

Example JQL queries for board filters

Here are a few common queries that you can use for your board's filter:

- Select all issues that are Unscheduled or in an Unreleased Fix Version:

```
project = GHS AND (fixVersion in unreleasedVersions() or  
fixVersion is empty)
```

- Select all issues you are interested in:

```
(assignee = currentUser() or reporter = currentUser()) AND  
(fixVersion in unreleasedVersions() or fixVersion is empty)
```

- Show all issues that you have participated in and have been updated in the last week.  This requires the [JIRA Toolkit add-on](#).

```
updatedDate > -7d AND Participants = currentUser()
```

- Select all issues for a team (using a Label custom field named 'Team')

```
(team = ateam or team = dreamteam or team = engineroom) AND  
(fixVersion in unreleasedVersions() or fixVersion is empty)
```

- Only select my bugs for a bugfix team

```
project = GHS AND team = bugfix AND issuetype = bug AND  
(fixVersion in unreleasedVersions() or fixVersion is empty)
```

- Try `fixVersion = earliestUnreleasedVersion(PROJECT KEY)` to see all issues in the next Fix Version to be released:

```
fixVersion = earliestUnreleasedVersion(PROJECT KEY)
```

See [JQL](#) for more information on using JQL for your board. Let your imagination run wild!

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- To learn more about your new board, see [What is a board?](#)
- If you wish to make any changes to your new board, see [Configuring a board](#).
- If you wish to make any changes to the issue filter for your board, see [Configuring filters](#).
- To learn more about adding a column to your board, see [Configuring columns](#).

What is a board?

A board displays issues from one or more projects, giving you a flexible way of viewing, managing, and reporting on work in progress. There are two types of boards in JIRA Software:



Scrum board — for teams that plan their work in sprints



Kanban board — for teams that focus on managing and constraining their work-in-progress

You can use a board that someone else has created, or create your own — you can create as many boards as you like. A project can have multiple boards, and a combination of Scrum boards and Kanban boards, if you choose.

Accessing a board

1. Click **Boards** (in header) > select your desired board.
2. On the Project sidebar, click one of the following, depending on what you need to do:
 - **Backlog**, **Active sprints**, or **Reports**, if you are working on a Scrum board
 - **Kanban board** or **Reports**, if you are working on a Kanban board

- Accessing a board
- Using the project sidebar
- Accessing a cross-project board
- Next steps

On a Scrum board

Backlog

The **Backlog** of a Scrum board shows the issues for your project(s) grouped into a backlog and sprints. In the Backlog, you can create and update issues, drag and drop issues to rank them, or assign them to sprints, epics, or versions, manage epics, and more. You would typically use the Backlog when building a backlog, planning a new version, and planning a sprint.

Active sprints

The Active sprints of a Scrum board displays the issues that your team is currently working on. You can create and update issues, and drag and drop issues to transition them through a workflow.

Reports

Reporting is an activity that you will be doing throughout a project. JIRA Software has a range of reports that you can use to show information about your project, versions, epics, sprints, and issues.

On a Kanban board

Kanban board

The *Kanban board* is a board that was created using

Reports

Reporting is an activity that you will be doing

the "Kanban" preset (see [Creating a board](#)).

Kanban is based on the continuous delivery of work. Rather than plan iterations, the flow of work is constantly monitored to ensure that there are always tasks being worked on. This means that when tasks are completed, new tasks are pulled into work-in-progress.

Use the Kanban board if your team focuses on managing and constraining work-in-progress.

Using the project sidebar

The project sidebar is your one-stop shop for anything and everything that concerns your project. Any tab you click on the project sidebar opens a page that contains information for the project that you're currently working on. Let's say you're working on an agile project and you're currently on a Scrum board for that project. When you click **Backlog** on the project sidebar, it opens the Backlog page, which contains the issues, stories, epics, and other details of your project.

The screenshot shows the 'Backlog' page for the 'Teams in Space' project. The sidebar on the left includes links for Backlog, Active sprints, Releases, Reports, Components, and Issues, along with project shortcuts and feedback options. The main content area displays a backlog with a summary for Sprint 6 (12 issues) and a list of user stories. Each story includes a description, an assignee, and a progress bar indicating completion status.

Accessing a cross-project board

A cross-project board is not specific to a single software project. That is, the board's filter may include a non-JIRA Software project, or it may include multiple projects. How you navigate to a cross-project board, and the navigation options that you'll see on the project sidebar of the board are different, compared to a board with a single JIRA Software project.

There are two ways to navigate to a cross-project board:

From Projects in header

If you navigate to **Projects** (in the header), then select the cross-project board, the project sidebar will display project-centric options, like the **Issues** link and project shortcuts.

Screenshot: Project sidebar (navigating to the board via Projects menu)

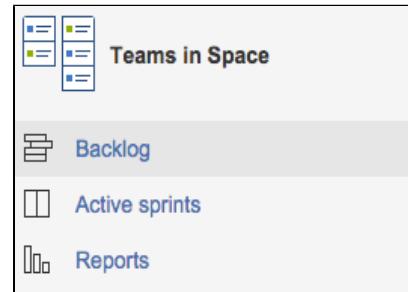
throughout a project. JIRA Software has a range of reports that you can use to show information about your project, versions, epics, sprints, and issues.

From Boards in header

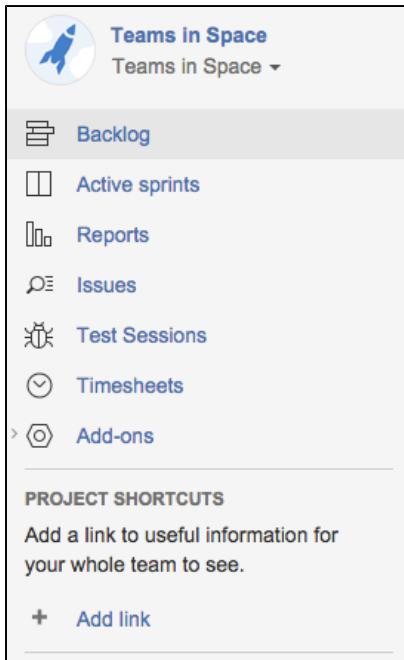
If you navigate to **Boards** in the header, then select the cross-project board, the project sidebar will *not* display project-centric options, like the **Issues** link and project shortcuts.

This is because JIRA Software cannot determine the project context for the cross-project board.

Screenshot: Project sidebar (navigating to the board via Boards menu)



As a result, we recommend that you navigate to a cross-project board via the **Projects** menu.



Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Creating a board](#)
- [Configuring a board](#)
- [Using the backlog](#)
- [Using active sprints](#)
- [Reporting](#)

Configuring a board

Configuring a board allows you to edit the mapping of workflow statuses to columns of a board, as well as edit the columns, swimlanes, and quick filters of a board, and customize the card colors and displayed issue fields.

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to modify its configuration.

On this page:

- [Before you begin](#)
- [Accessing a board's configuration](#)
- [Renaming a board](#)
- [Changing a board's administrators](#)
- [Sharing a board](#)
- [Next steps](#)

Accessing a board's configuration

1. Navigate to the desired board, then click **Board > Configure**.
2. On the **Board Configuration** screen, select the desired tab (**Columns**, **Swimlanes**, etc).

Screenshot: the 'Board Configuration' screen of a Scrum board ('General' tab).

Configure Scrum sample

Board ▾

CONFIGURATION General Columns Swimlanes Quick Filters Card colors Card layout Estimation Working days Issue Detail View	General and filter <p>The Board filter determines which issues appear on the board. It can be based on one or more projects, or custom JQL depending on your needs.</p> <p>General</p> <p>Board name Scrum sample</p> <p>Administrators Ashley Icamen [Administrator] (aicamen)</p> <p>Filter</p> <p>Saved Filter Filter for Scrum Sample Edit Filter Query</p> <p>Shares Project: Sample Scrum Project Edit Filter Shares</p> <p>Filter Query project = SSP ORDER BY Rank ASC</p> <p>Ranking Using Rank</p>
---	--

Screenshot: the 'Board Configuration' screen of a Kanban board ('General' tab).

Configure Sample Kanban Project

Board ▾

CONFIGURATION General Columns Swimlanes Quick Filters Card colors Card layout Working days Issue Detail View	General and filter <p>The Board filter determines which issues appear on the board. It can be based on one or more projects, or custom JQL depending on your needs.</p> <p>General</p> <p>Board name Sample Kanban Project</p> <p>Administrators Ashley Icamen [Administrator] (aicamen)</p> <p>Filter</p> <p>Saved Filter Filter for Sample Kanban Project Edit Filter Query</p> <p>Shares Project: Sample Kanban Project Edit Filter Shares</p> <p>Filter Query project = SKP ORDER BY Rank ASC</p> <p>Ranking Using Rank</p> <p>Kanban board sub-filter fixVersion in unreleasedVersions() OR fixVersion is EMPTY <small>Further filtering of issues for unreleased work.</small></p>
---	--

Renaming a board

1. Navigate to the desired board, then click **Board > Configure**.
2. On the **Board Configuration** screen, click the 'pencil' icon



(which will appear when you hover) to edit the name of your board. Press the **Enter** key when finished.

Changing a board's administrators

A board's *administrator* is the only person who can change the configuration of a board along with JIRA administrator users. By default, the administrator of a board includes the person who created it. If you are a board administrator or a JIRA administrator, you can change the administrators of a board. See [Permissions overview](#) for more information.

1. Navigate to the desired board, then click **Board > Configure**.
 2. On the **Board Configuration** screen, click any of the name(s) in the **Administrators** field (a 'pencil' icon
-
- will appear when you hover over them).

 3. Enter the names of the desired users or groups. If you start typing the name, a list of users and groups will be suggested. Press the **Enter** key when finished.

Note, the suggested user list will not display if you don't have the 'Browse Users' global permission. However, you can still enter the user names and group names manually. See [Permissions overview](#) for more information.
- Created in 2016 by Atlassian. Licensed under a Creative Commons Attribution 2.5 Australia License.

You may also want to give the new administrators rights to manage the JIRA filter on which the board is based – see the JIRA Admin documentation on [Managing shared filters](#).

Sharing a board

A board is available to all users who can view the saved filter on which the board is based. (Note that they will also need 'Browse' project permission for the project(s) whose issues are shown on the board. See [Permissions overview](#) for more information.)

If you wish to share a board with different people, you will need to either edit the saved filter (see the documentation on [issue filters](#)), or choose a different filter (see [Configuring filters](#)).

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following topics for more information about configuring your board:

- **Configuring filters** — Your board's filter is an `_JS_issue` filter (a `JQL` query) that specifies which issues are included on your board. For example, your board may include issues from multiple projects, or from only one project, or from a particular component of a project.
- **Enabling ranking** — Enabling ranking allows you to rank issues on a board by drag-and-drop, and to create sprints. It also enables `_JS_GHname` to group sub-tasks together underneath their parent issue.
- **Configuring columns** — The Active sprints of a Scrum board and a Kanban board display vertical columns. The default columns in the Active sprints of a Scrum board are 'To Do', 'In Progress', and 'Done'. On a Kanban board, the default columns are 'Backlog', 'Selected for Development', 'In Progress', and 'Done'. You can add, delete, rename, or move these columns if you wish. You can also choose which JIRA workflow status(es) each column is mapped to, and whether any constraints apply to each column.
- **Configuring swimlanes** — A swimlane is a horizontal categorization of issues in the Active sprints of a Scrum board, or on a Kanban board. You could use swimlanes to help you distinguish tasks from different workstreams, users, application areas, etc.
- **Configuring Quick Filters** — Quick Filters allow you (or anyone else using this board) to further filter the collection of issues appearing on a Scrum board or Kanban board. Use Quick Filters to switch between different issue types (e.g. show only bugs), or to show team-specific views of a common backlog.
- **Customizing cards** — You can change the card colors to help people quickly identify cards on your board as being of a particular issue type, priority, assignee, or — thanks to the power of JQL — practically anything you choose. You can also add up to three additional fields to display on cards in the Backlog and Active sprints of your Scrum board.
- **Configuring estimation and tracking** — The estimation statistic (e.g. story points, business value, etc) and the time tracking settings (remaining time estimates) can be customized to suit how you estimate and track work in your project.
- **Configuring the issue detail view** — The 'Details' tab of the issue detail view can be customized to show additional fields. For example, you may want to show an issue's Resolution, Environment, Security Level, custom fields, etc.
- **Configuring working days** — The Working Days setting for your board is used for different reports and gadgets. You change this setting to filter out weekends, holidays, and other times during which your team might not be working on your board's project(s).

Configuring filters

Your board's filter is an [issue filter](#) (a `JQL` query) that specifies which issues are included on your board. For example, your board may include issues from multiple projects, or from only one project, or from a particular component of a project.

On this page:

- Before you begin
- Choosing a different filter for your board
- Editing a board's filter
- (Kanban only) Adding a sub-filter
- Next steps

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to modify its configuration.

Choosing a different filter for your board

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the name of the filter displayed in the **Saved Filter** field. A pencil icon  will appear when you hover over it.
3. Choose a different filter for your board. Press the **Enter** key when finished.

Screenshot: the 'Board Configuration' screen — 'General' tab

General and filter

The Board filter determines which issues appear on the board. It can be based on one or more projects, or custom JQL depending on your needs.

General

Board name	Scrum: Teams in Space
Administrators	Alana Grant (agrant)

Filter

Saved Filter	Filter for Scrum: Teams in Space
	Edit Filter Query
Shares	 Project: Teams in Space
	Edit Filter Shares
Filter Query	project = "Teams in Space" ORDER BY Rank ASC
Ranking	Using Rank

Editing a board's filter

1. Navigate to the desired board, then click **Board > Configure**.
2. In the **General** tab:
 - To change the filter's JQL query, click **Edit Filter Query**. For more details, see the JIRA documentation on [JQL](#).
 - To change the filter's name, description, or shares, click **Edit Filter Shares**. For more details, see the JIRA documentation on [issue filters](#).

(Kanban only) Adding a sub-filter

If you are using a Kanban board, you can add a sub-filter that refines the issues returned by your board's filter. The sub-filter enables the **Kanban board** to show different data to the **Reports**.

The default sub-filter is:

```
fixVersion in unreleasedVersions() OR fixVersion is EMPTY
```

... this will result in issues released using the 'Release' button no longer appearing in the **Active sprints** (although the reports in **Reports** will continue to show them).

To add a sub-filter to your board:

1. Navigate to the desired board, then click **Board > Configure**.
2. Edit the **Kanban board sub-filter** field and enter a JQL query. For information on JQL syntax, see [JQL](#) (JIRA Admin documentation).

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Enabling ranking

Enabling ranking allows you to [rank issues](#) on a board by drag-and-drop, and to [create sprints](#). It also enables JIRA Software to group sub-tasks together underneath their parent issue.

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to enable ranking.

Enabling ranking

1. Navigate to the desired board, then click **Board > Configure**.
Note that only the administrator of a board (or a person with the 'JIRA Administrators' global permission) can configure a board.
2. Click the **Add Rank** button. This will append the following to your filter's query:

```
ORDER BY Rank ASC
```

Notes:

- JIRA Software automatically creates a custom field called **Rank**, of type **Global Rank**. Please ensure that your JIRA instance contains only one custom field called **Rank**, and that the **Rank** field is assigned to the global context (i.e. not to specific issues or projects).
- The sort order must be ascending. Descending sort order is not supported.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Configuring columns

The Active sprints of a Scrum board and a Kanban board display vertical columns. The default columns in the Active sprints of a Scrum board are 'To Do', 'In Progress', and 'Done'. On a Kanban board, the default columns are 'Backlog', 'Selected for Development', 'In Progress', and 'Done'. You can add, delete, rename, or move these columns if you wish. You can also choose which JIRA [workflow](#) status(es) each column is mapped to, and whether any constraints apply to each column.

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the

board to configure its columns.

On this page:

- Before you begin
- Editing columns
- Mapping statuses to columns
- Editing the mapping of JIRA workflow statuses to columns of a board
- Adding a new status
- Deleting a status
- Setting column constraints
- Next steps

Editing columns

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Columns** tab.
3. Edit the columns as described in the following table and the [screenshot](#) (below).

Add a new column	Click the Add Column button at the right of the page. A new column named ' New Column ' is added in the second to the last column position. If you are using Simplified Workflow , then a new status will automatically be created to match your new column.
Change the name of a column	Click in the name area of the column, modify the existing name, and press Enter .
Show the 'Days in column' indicator on cards	<p>Click the Days in column checkbox. This shows a series of dots on each card (up to the width of the card or a maximum of 32), representing the number of days that the issue has been in the column. This can help you see issues that are stagnating — this is particularly useful when your board is displayed as a wallboard.</p> <p>Note that if you move an issue back to one of the columns that you moved it into previously, the indicator reflects a cumulative value of the number of days the issue stayed in that column.</p> <p>For example, you move an issue to the 'In Progress' column and it stays there for 2 days. You then move the issue to the 'Code Review' column and it stays there for 1 day. During code review, you receive feedback, which requires more development work for the issue, so you move the issue back to the 'In Progress' column and it stays there for 1 day. In this example, the indicator reflects the cumulative value of 3 days for the issue in the 'In Progress' column.</p> <p>Disabling this indicator can improve performance. If you have a large instance (i.e. 300,000+ issues, 100+ projects, 100+ boards, or 100+ open sprints), we recommend that you disable this.</p> <p>This indicator is disabled by default for Scrum boards and enabled by default for Kanban boards.</p>

Delete a column	Click delete in the "Drag to rearrange or delete" text at the top of the column. Any JIRA workflow statuses that had been mapped to the deleted column are moved back to the ' Unmapped ' column.
Move a column	Hover over the top of the name of a column, then drag the column left or right to its new position. Wait until other columns have shifted position before dropping the selected column to its new position.

Screenshot: the 'Board Configuration' screen— 'Columns' tab.

Each column has a blue, yellow, or green color, as shown in the screenshot above.

- The single left-most column will always be blue, representing items in 'new' state (as per our [design guidelines](#)).
- The single right-most column will always be green, representing items in a 'successful' state.
- The middle column (or columns if you add more) are always yellow, representing items in progress.

You will see these colors shown in a number of places in JIRA Software, e.g. in [gadgets for JIRA applications](#)

Mapping statuses to columns

By default, a board's columns are mapped to the default JIRA statuses, as shown in the following tables. You can change this if you wish. For example, if you are using additional, customized JIRA statuses, you will probably want to map them to appropriate columns in your board.

If your board's project is using the JIRA default workflow:

Default column	Default mapped JIRA statuses
To Do	Open, Reopened
In Progress	In Progress
Done	Resolved, Closed

If your board's project is using [Simplified Workflow](#):

Default column	Default mapped JIRA statuses
To Do	To Do
In Progress	In Progress
Done	Done

Editing the mapping of JIRA workflow statuses to columns of a board

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Columns** tab.
3. Change the mapping statuses, as described in the following table and [Screenshot \(above\)](#).

Map a status to a column	Drag a status from the Unmapped column to the appropriate column on the right.
Unmap a status from a column	Drag a status from its current column on the right to the Unmapped column on the left.
Change the column mappings of a status	Drag a status from its original column to its relevant new column.

Note, all statuses configured in the JIRA server are available from the board Configuration page. However, some statuses (in particular, custom statuses) may not be available for issues on your board if the JIRA [workflows](#) used by these issues do not utilize those statuses.

Adding a new status

If your board is using Simplified Workflow:

1. Navigate to the desired board, then click **Board > Configure**.
 2. Click the **Columns** tab.
 3. Click the **Add Status** button at the right of the page.
- i** Note that the **Add Status** button is only available if you have the JIRA '**Project Administrator**' permission for this board's project. See [Permissions overview](#) for more information.

Deleting a status

You can only delete a status if:

- you are the 'Project Administrator' for this board's project(s), and
- there are no issues that currently have that status.

If any other workflows are using this status, then the status will be removed from your workflow, but not deleted.

If your board is using Simplified Workflow:

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Columns** tab.
3. Drag the status to the **Unmapped Statuses** column.
4. Click the **Remove Status icon (x)** for the status you wish to remove.

If your board is *not* using Simplified Workflow, you can only delete a status via the JIRA administration interface. See [Workflows](#) for more information.

Setting column constraints

Setting constraints on each workflow state is a crucial part of Kanban, so that you can ensure that work is continually flowing through the pipeline. You may also find it handy in Scrum, so that you can see bottlenecks early, and adjust the scope of your sprint.

Constraints trigger a visual indicator when a maximum or minimum number of issues appear in a column. You can specify constraints for all columns, or just some of them. If a column constraint is exceeded, then the column will change color, as follows:

- Column colored red — Maximum number of issues exceeded
- Column colored yellow — Minimum number of issues not met

Please note:

- The values of the column constraints will appear at the top of each column you set them for. These values don't have any impact on the number of issues displayed in the columns.
- Column constraints apply to the total number of issues in a column, regardless of whether issues are hidden by a Quick Filter.

To set column constraints of a board:

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Columns** tab.
3. Edit the constraints, as described in the following table.

Enable column constraints	In the Column Constraint drop-down, select either Issue Count or Issue Count, excluding sub-tasks . Note, enabling column constraints will show the issue count in each column. You may want to exclude sub-tasks (particularly Scrum teams), so you can constrain the number of stories for your team, rather than the number of individual tasks.
Set a column's 'Min' or 'Max' constraint	Edit the fields at the top of the desired column: - Enter a minimum value in the yellow box. - Enter a maximum value in the red box. Delete an existing value to remove the constraint.

Note, you can also remove constraints for all columns. In the **Column Constraint** drop-down, select **No constraint**.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Read [Using the Simplified Workflow](#) to find out more about this workflow and how to change to it.
- Read [Transitioning an issue](#) for information on how issues transition from column to column.

Using the Simplified Workflow

Every board can represent one or more projects; however, the Simplified Workflow can only be used if a board represents a single project. Every project uses either a **JIRA Workflow** or the **Simplified Workflow** to control the transitioning of issues from one status to another. The workflow determines which statuses are available.

Your board will be using the Simplified Workflow if your board was created as part of creating a project, (including via the JIRA Software Getting Started page).

On this page:

- Before you begin
- Benefits of using the Simplified Workflow
- Switching to Simplified Workflow
- Next steps

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to switch it to the Simplified Workflow.

Benefits of using the Simplified Workflow

The Simplified Workflow offers the following benefits over a JIRA workflow:

Simplified Workflow	JIRA Workflow
<ul style="list-style-type: none"> • Has three default steps for Scrum boards: To Do, In Progress, Done. • Has four default steps for Kanban boards: Backlog, Selected for Development, In Progress, Done. 	<ul style="list-style-type: none"> • Has more steps than are typically needed on a board (e.g. see the default JIRA workflow).

<ul style="list-style-type: none"> Allows issues to be dragged freely between columns. 	<ul style="list-style-type: none"> Has workflow "conditions" that prevent issues from being dragged freely between all columns.
<ul style="list-style-type: none"> Displays no screens on any transitions – all transitions will happen instantly. Automatically sets a resolution of 'Done' when issues are transitioned to the 'successful' column, i.e. statuses mapped to the successful column (see Configuring columns). 	<ul style="list-style-type: none"> Displays screens for Resolve Issue, Close Issue, and Reopen Issue.
<ul style="list-style-type: none"> Can be edited via the board configuration (see <i>Adding a new status</i> on the Configuring columns page), provided that you are the project administrator for the project that is on the board. 	<ul style="list-style-type: none"> Can only be edited via workflow configuration, see Workflows for more information.

For information about workflow steps, transitions, and conditions, see [Workflows](#) for more information.

Switching to Simplified Workflow

How can I tell if my board is using Simplified Workflow?

To check this, view the **Columns** configuration for your board (described below) — If you are using the Simplified Workflow, you will see the words 'Using Simplified Workflow' (see [Screenshot](#)).

Otherwise, you will see the 'Simplify workflow' button or the words 'Simplified Workflow unavailable'.

If your project is currently using a JIRA workflow, you may want to switch to the Simplified Workflow. You will then be able to easily add new statuses from within JIRA Software. Note, the default steps for the Simplified Workflow (described in table above) *will not* be created when you switch from a JIRA workflow. Your existing workflow steps will be included in the workflow instead. The default steps are only created if the Simplified Workflow is chosen when the board and project are created in JIRA Software.

Project pre-requisites

You will only be able to switch to Simplified Workflow if:

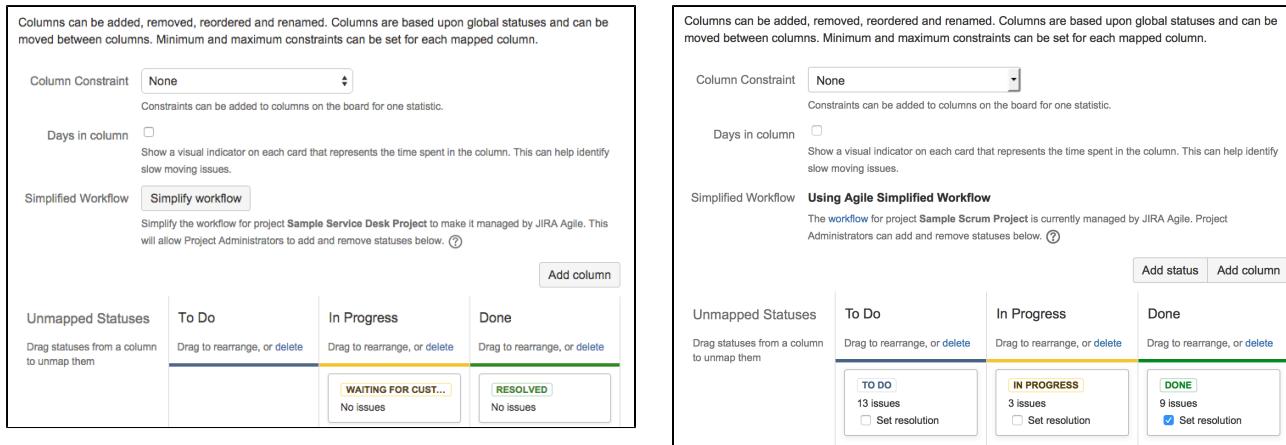
- There is only one project being viewed by your board (to check this, look at the board's filter)
- That project uses a JIRA [workflow scheme](#), which only has one workflow for all issue types
- Your workflow only uses Post Functions, Validators, and Conditions, which are provided by Atlassian (not any of which are provided by add-ons)
- The existing workflow has at least one outgoing transition for each status

To switch to Simplified Workflow:

- Before you begin, back up your JIRA data, see [Backing up data](#) (JIRA Admin documentation). Switching to the Simplified Workflow will migrate all of the project's issues to the new workflow, which may take some time. The project's original workflow and scheme not be changed, so any other projects using these will not be affected.
- Navigate to the desired board, then click **Board > Configure**.
- Click the **Columns** tab.
- Click the **Simplify workflow** button, as shown in first [Screenshot](#) below. If this button does not display, check that your project meets the pre-requisites described above.
- Your project(s) will switch to use Simplified Workflow, as shown in the second [Screenshot](#) below. The **Add Status** button will now be shown.

Screenshot: the 'Board Configuration' screen ('Columns' tab) — this board is not using Simplified Workflow.

Screenshot: the 'Board Configuration' screen ('Columns' tab) — this board is using Simplified Workflow:



Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Read [Configuring columns](#) to learn how to configure other settings for your columns, e.g. mapping statuses to columns.
- Read [Transitioning an issue](#) for information on how issues transition from column to column.
- Read [Configuring filters](#) for information on how to check a board's filter.

Configuring swimlanes

A swimlane is a horizontal categorization of issues in the Active sprints of a Scrum board, or on a Kanban board. You could use swimlanes to help you distinguish tasks from different workstreams, users, application areas, etc.

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to configure its swimlanes.

On this page:

- [Before you begin](#)
- [Choosing a different type of swimlane](#)
- [Modifying your Query-based swimlanes](#)
- [Next steps](#)

Choosing a different type of swimlane

You can choose to set up your swimlanes in a variety of ways, as shown in the following table.

Base your swimlanes on...	Explanation
Queries	<p>One JQL query per swimlane (see below for examples). By default, two swimlanes will be created:</p> <ul style="list-style-type: none"> • Expedite — this swimlane is based on the following JQL query: <code>priority = Blocker</code> • Everything Else — this swimlane is always at the bottom of the screen, and cannot be deleted. It acts as a "catch-all" for issues that do not match the JQL of any of the swimlanes above it, hence it has no JQL specified. <p>You may want to create additional swimlanes that map to other values of your JIRA 'Priority' field, or use a different field to categorize your swimlanes.</p>
Stories	One parent issue per swimlane (i.e. each swimlane contains all of the parent's sub-tasks), with issues that have no sub-tasks appearing below.

Assignees	One assignee per swimlane, with unassigned issues appearing either above or below the swimlanes (your choice).
Epics	<p>One epic per swimlane, with issues that don't belong any to epics appearing below the swimlanes.</p> <p>If you want to change the order of the swimlanes on your board, navigate to the Backlog of the board and drag and drop the epics as desired.</p>
No Swimlanes	No horizontal categorization of issues in the Active sprints of a Scrum board, or on a Kanban board.

To choose a different type of swimlane:

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Swimlanes** tab.
3. In the **Base Swimlanes on** drop-down, select either **Queries**, **Stories**, **Assignees**, or **No Swimlanes**, as described above.

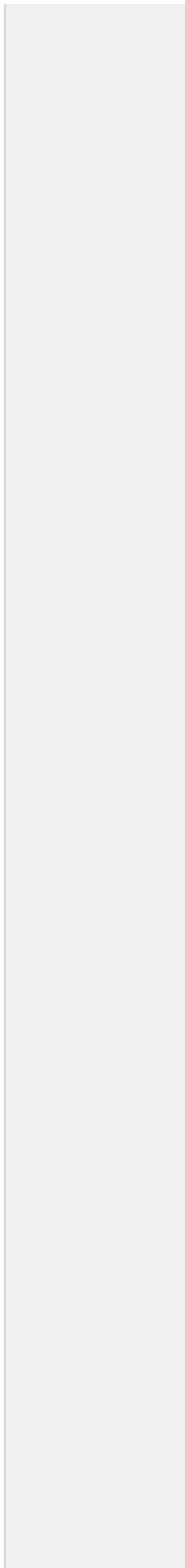
Screenshot: the 'Board Configuration' screen — 'Swimlanes' tab

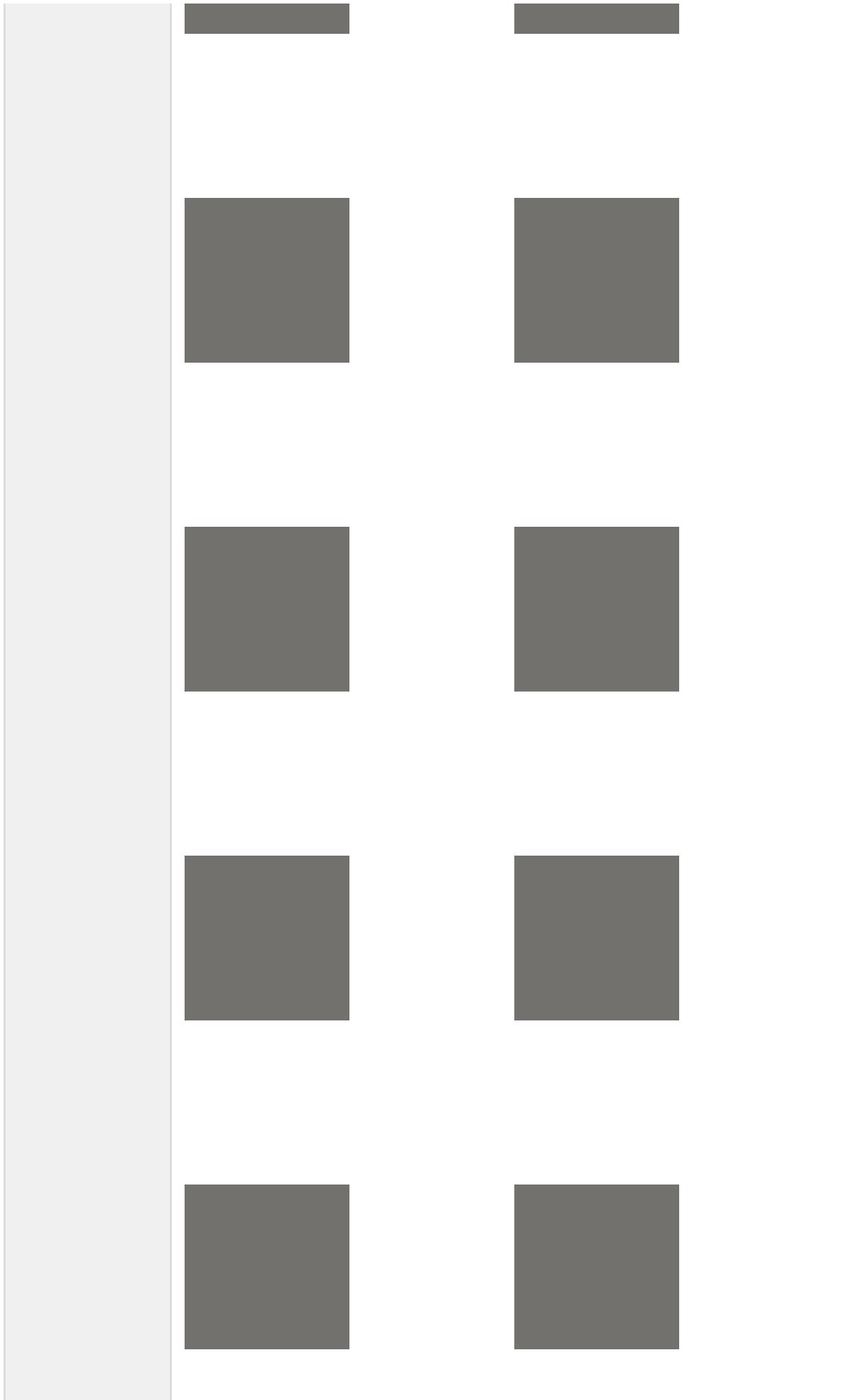
Modifying your Query-based swimlanes

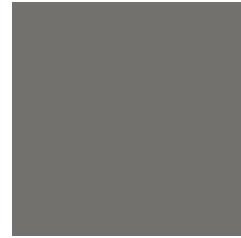
If your swimlanes are based on JQL queries (rather than on stories or assignees), you can create, delete, or change them:

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Swimlanes** tab.
3. If your swimlanes are based on **Queries**, you can edit your swimlanes, as described in the following table and the screenshot above.

Add a new swimlane	In the blue area, type the Name , JQL , and optional Description , then click the Add button. Your new swimlane is added in the top swimlane position.
Change the name of a swimlane	Click in the Name area of the swimlane, modify the existing name, and click the Update button.
Change the JQL of a swimlane	<p>Click in the JQL area of the swimlane, modify the existing JQL, and click the Update button.</p> <p>See the examples below for some suggestions. For information on JQL syntax, see JQL (JIRA Admin documentation).</p> <p><i>Note, the JQL 'ORDER BY' clause is not used by the swimlane, as it defaults to order by rank.</i></p>
Delete a swimlane	Click the Delete button at the right of the swimlane.
Move a swimlane	Hover over the vertical 'grid'







icon, then drag and drop the swimlane up or down to its new position.

Some example JQL queries for your swimlanes:

- Show all issues that belong to a particular component, e.g. 'User Interface':

```
project = "Angry Nerds" AND component = "User Interface"
```

- Show all issues that are due in the next 24 hours:

```
due <= "24h"
```

- Show all issues that have a particular priority, e.g.:

```
priority = "Minor"
```

and

```
priority = "Major"
```

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Configuring Quick Filters

Quick Filters allow you (or anyone else using this board) to further filter the collection of issues appearing on a Scrum board or Kanban board. Use Quick Filters to switch between different issue types (e.g. show only bugs), or to show team-specific views of a common backlog.

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to configure its Quick Filters.

About the default Quick Filters

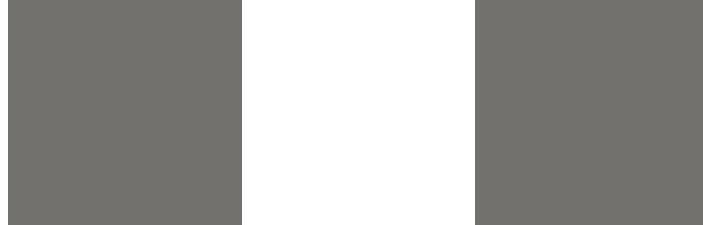
By default, a board contains two Quick Filters, called 'Only My Issues' and 'Recently Updated':

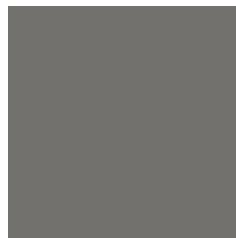
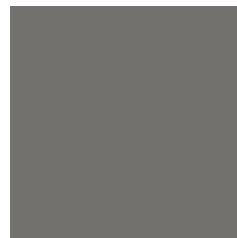
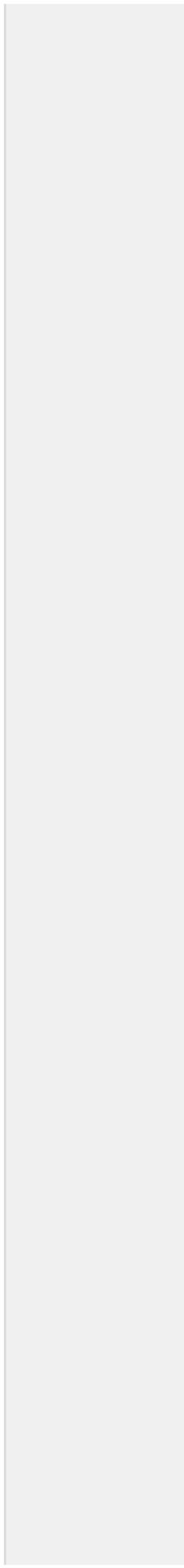
Default Quick Filter	Default JQL	Notes
Only My Issues	<code>assignee = currentUser()</code>	Displays issues assigned to the person who is currently viewing this board
Recently Updated	<code>updatedDate >= -1d</code>	Displays issues that have been updated in the last 24 hours

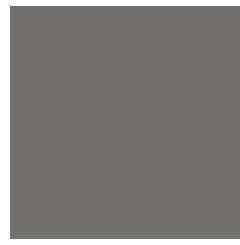
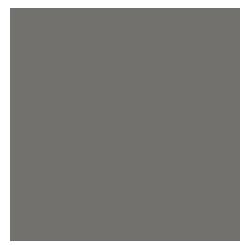
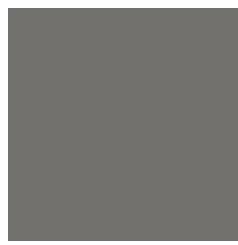
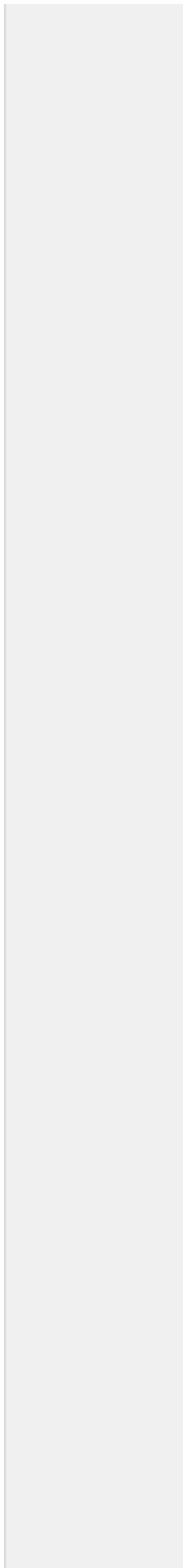
Editing Quick Filters

You can create additional Quick Filters or edit existing Quick Filters. Any additional Quick Filters that you create will appear as buttons next to the 'Only My Issues' and 'Recently Updated' buttons on the board.

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Quick Filters** tab.
3. Edit the Quick Filters, as described in the following table and screenshot (below).

Add a new Quick Filter	In the blue area, type the Name , JQL , and a Description (optional), then click the Add button. Your new Quick Filter will be added in the top Quick Filter position.
Change the name of a Quick Filter	Click in the Name area of the Quick Filter, modify the existing name, and click the Update button.
Change the JQL of a Quick Filter	Click in the JQL area of the Quick Filter, modify the existing JQL, and click the Update button. See the examples below for some suggestions. For information on JQL syntax, see JQL .
Delete a Quick Filter	Click the Delete button at the right of the Quick Filter.
Move a Quick Filter	Hover over the vertical 'grid'   





icon, then drag and drop the Quick Filter up or down to its new position.

When this board is used, the top-most Quick Filter appears in the left-most position.

Screenshot: the 'Board Configuration' screen — 'Quick Filters' tab

Name	JQL	Description	Action
Only My Issues	assignee = currentUser()	Displays issues which are currently assigned to the current user	Delete
Recently Updated	updatedDate >= -1d	Displays issues which have been updated in the last day	Delete

Sample JQL you might wish to use for your Quick Filters:

- Show all issues that are assigned to members of the "bugfix" group:

```
assignee in membersOf("bugfix")
```

- Show all issues of type 'Bug':

```
type = "Bug"
```

- Use an issue filter in your Quick Filter:

```
savedfilter = "My Filter"
```

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Read [Configuring a board](#) to learn how to configure other parts of your board, like columns or swimlanes.
- Read [JQL](#) for information on JQL syntax.

Customizing cards

Don't rely on people clicking cards to view the issue details. Customizing the cards for your boards can help you bring the right information to your team's attention, at a glance. You can **change the card colors** to help people quickly identify cards on your board as being of a particular issue type, priority, assignee, or — thanks to the power of JQL — practically anything you choose. You can also **add up to three additional fields** to display on cards in the Backlog and Active sprints of your Scrum board.

On this page:

- Before you begin
- Configuring card colors
- Adding fields to cards
- Printing issue cards
- Next steps

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to customize its cards.

Configuring card colors

You can base your card colors on issue types, priorities, assignees, or JQL queries. Once you have chosen a method, you can change or delete the colors for each type of card. This can be configured per board (not globally).

1. Navigate to the desired board, then click **Board > Configure**.
2. Click **Card Colors** and change the **Colors based on** drop-down as desired. If you change to a different method of card coloring, your settings for the old method will be retained so you can switch back to them later if you wish.

▼ [Read more about the card coloring methods](#)

Base your card colors on....	Explanation
Issue types	<p>One color per issue type. A default color will be allocated to every issue type that matches issues on the board. Note, the issue type must already exist in your project to configure the color for it. Also, the color values won't display until an issue is created on your board. You need to create an issue on your board first, to configure the colors for the issue types.</p> <p>Default issue types and colors: Improvement, Task, New Feature, Bug</p>
Priorities	<p>One color per priority. The default colors are the same as used for priorities in JIRA.</p>
Assignees	<p>One color per assignee. A default color will be allocated to every user who is or has been an assignee of issues on this board. Note, the color values won't display until an issue is created on your board. You need to create an issue on your board first, to configure the colors for the assignees.</p>

<p>Queries</p> <p>One color per JQL query. You can specify whatever queries you wish. Issues that do not match any of your JQL queries will be shown in grey. See JQL for more information about JQL syntax.</p> <p>Example queries:</p> <ul style="list-style-type: none"> • Show all issues that belong to a particular component, e.g. 'User Interface': <div style="border: 1px dashed #ccc; padding: 10px; margin-top: 10px;"> <pre>project = "Angry Nerds" AND component = "User Interface"</pre> </div> <ul style="list-style-type: none"> • Show all issues that are due in the next 24 hours: <div style="border: 1px dashed #ccc; padding: 10px; margin-top: 10px;"> <pre>due <= "24h"</pre> </div> <ul style="list-style-type: none"> • Show all issues created by a particular user, e.g.: <div style="border: 1px dashed #ccc; padding: 10px; margin-top: 10px;"> <pre>reporter = "Jane"</pre> </div> <p>and</p> <div style="border: 1px dashed #ccc; padding: 10px; margin-top: 10px;"> <pre>reporter = "Bob"</pre> </div>

3. Once you have picked a method to base your colors on, customize the colors as follows:

- **Pick a different color for a card** — Click the **Color** square, e.g.



- **Delete a card color** — Click **Delete**. This effectively resets the card to the default color. If you still have issues matching that card type, refresh the screen to reset the deleted card color to the default color).
- **Move a card color (Query-based only)** — Hover over the vertical 'grid'



icon, then drag and drop the color up or down to its new position.

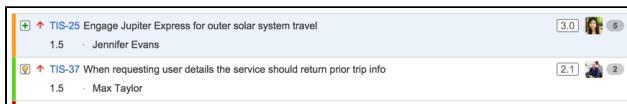
Note: For Query-based colors, the order is important, as each issue will be colored according to the first query that it matches (e.g. if your first row first row has query "issuetype = bug" and is colored red, and your second row has query "assignee = kevin" and is colored green, then bugs assigned to Dave will appear red).

Adding fields to cards

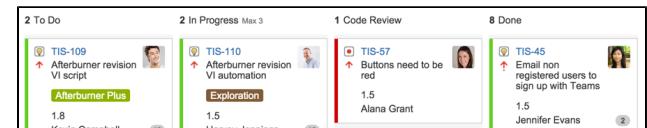
You can configure cards on a board to display up to three additional fields. The fields can be different for the Backlog and Active sprints, if you are using a Scrum board.

1. Navigate to the desired board, then click **Board > Configure**.
2. Click **Card layout**, then add or delete fields as desired.

Screenshot: Cards displaying additional fields in the Backlog



Screenshot: Cards displaying additional fields in Active sprints



Printing issue cards

Whether you're planning work or working on issues on your Scrum or Kanban board, it may also be good to print out these issue cards. You and your team can use the printed cards on a physical board, which can be a replication of your board on JIRA Software. You can print a single issue card or multiple issue cards, if you want. You can also print all issue cards in your current board.

The printed issue cards include the following issue details:

- Summary
- Issue type
- Issue key
- Issue priority
- Estimate
- Assignee
- Epic (optional in Active sprints/Kanban board)
- Version (when printing from the Backlog)
- Up to 3 extra fields, depending on your card layout configuration

The printed issue cards fit on A4-, A3-, or Letter-sized pages in both portrait and landscape modes.

See [Printing issue cards](#) for more information.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Read [Printing issue cards](#) to learn how to print issue cards and use a physical replication of your board in JIRA Software.

Configuring estimation and tracking

The estimation statistic (e.g. story points, business value, etc) and the time tracking settings (remaining time estimates) can be customized to suit how you estimate and track work in your project.

Many Scrum teams separate **estimation** (which is used for measuring the size of a backlog and calculating velocity) from **tracking** (which is often the burndown of hours used during the Sprint to make sure that they are on track to complete the stories during the Sprint period), and use different units for each. A common approach is to *estimate* tasks in Story Points, then *track* tasks using hours. JIRA Software gives you the flexibility to set your estimation and tracking statistics differently, depending on what best suits your team.

On this page:

- Before you begin
- Setting estimation statistic and time tracking
- More information about estimation and time tracking
- Next steps

Before you begin

- You can only configure estimation and time tracking for Scrum boards.
- You must be a **JIRA administrator** or a **board administrator** for the board to configure estimation or time tracking for it.

Setting estimation statistic and time tracking

1. Navigate to the desired board, then click **Board > Configure**.
Note, only the administrator of a board (or a person with the 'JIRA Administrators' global permission) can configure a board.
2. Click the **Estimation** tab. The Estimation page will be displayed.

Estimation

Issues can be estimated when in the Backlog to get an idea of how much work is being committed to in a sprint. [Read more about estimation and tracking.](#)

Estimation Statistic [Story Points](#)

Estimate issues in the Backlog by entering values for **Story Points**. Your velocity from sprint to sprint will be measured against these estimates.

Time Tracking **None**
Issues will burn down their **Story Points** value upon completion.
 Remaining Estimate and Time Spent
Track time against issues using JIRA's **Remaining Estimate** and **Time Spent** fields. The Burndown Chart will be based on these two values.

3. In the **Estimation Statistic** field, choose one of the following options:

Estimation statistic	Explanation
Story Points	<p>Estimation will be based on the number of Story Points per issue. This is the most commonly used option.</p> <p>By default, the Story Points field is only available to issues of type 'Story' or 'Epic' (not 'Bugs' etc). If you want to change this, do the following:</p> <ol style="list-style-type: none"> Associate the Story Points field with other issue types, see custom field context (JIRA Admin documentation). Specify the screens that the Story Points field should be displayed on, see Defining a screen (JIRA Admin documentation).
Original Time Estimate	Estimation will be based on the JIRA ' Original Time Estimate ' field (see Logging work on issues for more information). By default, this is specified in minutes, but you can use hours, days, or weeks, depending on your JIRA system configuration. See Configuring time tracking (JIRA Admin documentation).
Business Value	Estimation will be based on the Business Value of each issue.
Issue Count	Estimation will be based on the number of issues in the sprint. The ' Estimate ' field will not be editable.
<Custom Field>	Estimation can be based on any numeric custom field in your JIRA system.

4. In the **Time Tracking** field, choose one of the following options:

Tracking Statistic	Explanation
None	Tracking will be based on the Estimation Statistic.
Remaining Estimate and Time Spent	<p>Tracking will be based on the JIRA 'Remaining Estimate' and 'Time Spent' fields (see Logging work on issues for more information). By default, these fields are specified in minutes, but you can use hours, days, or weeks, depending on your JIRA system configuration, see Configuring time tracking (JIRA Admin documentation).</p> <p>Note that this is fundamentally different from using the Estimation Statistic for burndown, in that values do not burn down when an issue is completed — instead, values only burn down when users enter Time Spent or set the Remaining Estimate to a new value.</p>

More information about estimation and time tracking

Product teams often need to be able to estimate how long a product will take to deliver. This is difficult because the backlog may stretch many months into the future, so the team can only provide a very rough estimate in conditions of uncertainty without wasting days breaking the work down. However, from sprint to sprint, as they work through the stories, the team will develop a cadence of completing <x> units of work they had 'roughly estimated', i.e. their *velocity*.

This means that they can accurately estimate how long portions of the backlog will take to get done with simple rough estimates. However, to make this work, the team needs to estimate stories with a consistent level of uncertainty. The team also needs to track the amount of estimation units they have actually fully completed from sprint to sprint, because this number tells us with relative certainty how much we can fit into each future sprint.

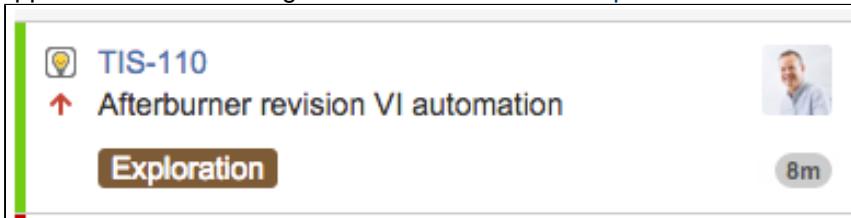
How the Estimation Statistic and Tracking Statistic affects your project

In JIRA Software, you can choose which type of units (e.g. Story Points, Issue Count) will be used for estimating and tracking issues. You do this by choosing an Estimation Statistic, then choosing to either use the same units for your Tracking Statistic or to use time-tracking. Each board can have a different type of Estimation Statistic and Tracking Statistic.

- The type of Estimation Statistic you select affects the units that are used by the **Estimate** field, which appears at the right of each issue in the [Backlog](#): (Note that the **Estimate** field is [editable](#) when an issue is in the [Backlog](#), but is not editable once the issue moves into Active sprints.)

TIS-109 Afterburner revision VI script	Afterburner Plus	12
TIS-110 Afterburner revision VI automation	Exploration	12
TIS-45 Email non registered users to sign up with Teams In Space	Afterburner Plus	2

- The type of Tracking Statistic you select affects the units that are used by the **Remaining** field, which appears at the bottom right of each issue in [Active sprints](#):



How you can view your Velocity and Burndown

A team's velocity is based on the Estimation Statistic — i.e. for each sprint, the velocity is the sum of the Estimation Statistic for completed stories. Velocity is shown in the [Velocity Chart](#) and also on the [Sprint Report](#), in the Estimate Statistic column header of the "Completed Issues" table (e.g. "Story Points (12)" means that 12 Story Points were completed in that sprint). Please note that the values for each issue are recorded at the time when the issue moves into the sprint. Changing the Estimate value afterwards will not be reflected in the [Sprint Report](#), but will be shown as scope change in the burndown. Velocity is also used in the [Version Report](#), to predict Release Dates.

The [Burndown Chart](#) is based on the Tracking Statistic. If you are using Story Points as your Tracking Statistic, then the Burndown Chart shows the Story Points per story (i.e. stories burning down the Estimate Statistic are only burnt down on the graph as they are completed); whereas if you choose the Time-tracking option, you are shown partial burndown (i.e. the number of hours currently used and remaining each day).

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Learn more about estimating issues: [Estimating an issue](#)
- Learn about some of the charts that estimation and time tracking affects: [Burndown Chart](#), [Velocity Chart](#), [Version Report](#)

Configuring the issue detail view

The 'Details' tab of the [issue detail view](#) can be customized to show additional fields. For example, you may want to show an issue's Resolution, Environment, Security Level, custom fields, etc.

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to configure the issue detail view.

Adding, removing, or moving a field

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Issue Detail View** tab.
3. Add or remove the desired fields, or use the vertical 'grid'  icon to drag and drop fields up or down into a different order.
4. Your fields will then appear on the **Details** tab in the [Issue Detail View](#). Note that fields will only appear on an issue if they have been associated with the relevant issue type, and are not "hidden". For details, please see the following pages (JIRA Admin documentation):
 - for built-in fields: [field configuration](#)
 - for custom fields: [custom field context](#)

Screenshot: the 'Board Configuration' screen — 'Issue Detail View' tab

Add, delete or reorder fields on the Issue Detail View.

General Fields

Field Name

Business Value	▲	Add
Status		Delete
Component/s		Delete
Labels		Delete
Affects Version/s		Delete
Fix Version/s		Delete
Epic		Delete

Date Fields

Field Name

Due Date	▲	Add
Created		Delete
Updated		Delete

People

Field Name

Votes	▲	Add
Reporter		Delete
Assignee		Delete

Next steps

-  **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Configuring working days

The **Working Days** setting for your board is used for different reports and gadgets. You change this setting to filter out weekends, holidays, and other times during which your team might not be working on your board's project(s).

Before you begin

You must be a **JIRA administrator** or a **board administrator** for the board to configure estimation or time tracking for it.

Configuring working days

1. Navigate to the desired board, then click **Board > Configure**.
2. Click the **Working days** tab.
3. If your team has a different time zone from that of the server, select the team's **Time Zone**.
4. In **Standard Working Days**, select all the weekdays during which your team typically works.
5. In **Non-Working Days**, click **Add Date** to specify holidays or one-off dates during which your team will not be working.

To remove a **Non-Working Day**, click the 'x' next to the date.

These settings will be reflected in the following reports: Burndown Chart, Sprint Report, Epic Report, Version Report, Control Chart; and the following gadgets: Sprint Health Gadget, Burndown Gadget.

Screenshot: the 'Board Configuration' screen — 'Working days' tab

Working days

These settings allow you to highlight or exclude non-working days from reports and gadgets.

They apply to the following reports: Burndown Chart, Sprint Report, Epic Report, Version Report, Control Chart; and the following gadgets: Sprint Health Gadget, Burndown Gadget.

Time Zone

Region: Australia

Time Zone: (GMT+10:00) Sydney

Standard Working Days

Every Monday
 Tuesday
 Wednesday
 Thursday
 Friday
 Saturday
 Sunday

Non-Working Days

Dates: No dates

Add Date

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Learn more about the charts that working days affect: [Version Report](#), [Control Chart](#), [Burndown Chart](#), [Sprint Report](#), [Epic Report](#)
- Learn how working days affect the Sprint Health Gadget, Burndown Gadget: [Gadgets for JIRA applications](#)

Workflows

All JIRA projects contain issues that your team can view, work on, and transition through stages of work — from creation to completion. The path that your issues take is called a workflow. Each JIRA workflow is composed of

a set of statuses and transitions that your issue moves through during its lifecycle, and typically represents work processes within your organization.

In addition, JIRA uses workflow schemes to define the relationship between issue types and workflows. Workflow schemes are associated with a project, and make it possible to use a different workflow for different combinations of project and issue types.

If you need to edit or create a more advanced workflow to match how your team or organization works, you can log in as a **JIRA Administrator** with global permission to access and create your workflow.

What you can do...	Documentation
<ul style="list-style-type: none"> Edit existing workflows Create new workflows Configure existing workflows 	Working with workflows
<ul style="list-style-type: none"> Add a workflow scheme Configure a workflow scheme Managing workflow schemes 	Configuring workflow schemes
<ul style="list-style-type: none"> Import and export workflows Activate and deactivate workflows 	Managing your workflows
<ul style="list-style-type: none"> Adding custom events Configuring the initial status Working in text mode Configuring workflow triggers Using validators and custom fields Using XML to create a workflow Workflow properties 	Advanced workflow configuration

Organizing work with components

Components are used to group issues in a project, effectively creating sub-sections of a project. For example, you could set up components to group issues related to different workstreams, like the user interface or APIs. You can also set a default assignee for a component, which will cause all new issues to be assigned to that user. For example, you want all documentation issues to be assigned to your technical writer.

You need to have the project-specific **Administer Projects** permission or the **JIRA Administrator** global permission to be able to:

- Add — create a new component against which issues can be aligned
- Edit — change a component's details
- Delete — remove a component

Once a component has been created for a project, the 'Component' field becomes available for your issues. If you cannot see this field on your issue, your project may not have any components yet, or the field is hidden from view.

On this page:

- [Managing a project's components](#)
- [Adding a new component](#)
- [Selecting a default assignee](#)
- [Editing a component's details](#)
- [Deleting a component](#)

Managing a project's components

The easiest way to manage a project's components is through the Components page.

1. Choose



> **Projects**, and click the name of the project.

2. Choose **Components** in the sidebar. The **Components** page is displayed, showing a list of components and each component's details. From here, you can manage the project's components as described below.

Adding a new component

1. The Add Component form is located at the top of the 'Components' screen.
2. Enter the **Name** for the component. Optionally, enter a **Description** and select a **Component Lead** and **Default Assignee** (see [options](#) below).
3. Click **Add**.

Selecting a default assignee

You can optionally set a default assignee for a component. This will override the project's default assignee for issues in that component. If an issue has multiple components, and the default assignees of components clash, the assignee will be set to the default assignee of the component that is first alphabetically.

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

Editing a component's details

1. On the 'Components' screen, open the menu in the **Actions** column for the component you want to edit, and select **Edit**.
2. Edit the component's **Name**, **Description**, **Lead**, and **Default Assignee** in the **Edit component** dialog.
3. Click the **Save** button to save your changes.

Searching for a component

If you need to find a component in a long list, it's easiest to search for it. Start typing text into the search box that you know the component contains, and your list will automatically be filtered for you.

Deleting a component

1. On the 'Components' screen, open the menu in the **Actions** column for the component you want to delete, and select **Delete**.
2. You will be prompted to associate any issues assigned to this component with another component if

you wish.

3. Click the **Delete** button to delete the component.

Customizing the issues in a project

Issues are the packets of work that need to be completed in a project. These issues are made up of issue fields, and the issue fields contain data about the issue. This data is important, as it helps define the issue, and can contain important information about the issue, such as a summary, a description, due dates, and when and where the work is required. JIRA Software allows you to customize issues by changing the configuration and behavior of these fields to suit your team's needs. You may choose to:

- Change a field's behavior (such as change a field's description, make a field hidden or visible, or make a field required or optional)
- Add your own values for fields that have default values assigned (e.g. Resolution and Status)
- Create new 'custom' fields
- Configure different renderers 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)

A simple example of how customizing an issue could benefit your team could be marking fields as 'Required' when an issue is created. This would ensure you always capture the required information you need to get the work done to resolve the issue. If you couple this with positioning the required fields at the top of the screen, and even hiding fields you know the issue creator won't use, you'll make sure your users can see and complete the required fields as quickly as possible.

You can make this...	...into this!
<p>Create Issue</p> <p>Project* <input type="button" value="Development Strategy (DSE)"/></p> <p>Issue Type* <input type="button" value="Task"/></p> <p>Field Tab Group</p> <p>Summary* <input type="text"/></p> <p>Account <input type="button" value="Please select"/> Tempo Account custom field</p> <p>Due Date <input type="text"/></p> <p>Priority <input type="button" value="Medium"/></p> <p>Environment* <input type="button" value="Style B I U A"/></p> <p><small>For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).</small></p> <p>Description <input type="text"/></p> <p><small>For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).</small></p> <p><input type="checkbox"/> Create another <input type="button" value="Create"/> <input type="button" value="Cancel"/></p>	<p>Create Issue</p> <p>Project* <input type="button" value="Development Strategy (DSE)"/></p> <p>Issue Type* <input type="button" value="Task"/></p> <p>Summary* <input type="text"/></p> <p>Environment* <input type="button" value="Style B I U A"/></p> <p><small>For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).</small></p> <p>Description <input type="button" value="Style B I U A"/></p> <p><small>For example operating system, software platform and/or hardware specifications (include as appropriate for the issue).</small></p> <p><input type="checkbox"/> Create another</p>

To customize your issues, you need to be a JIRA administrator. You can review more conceptual information on [customizing issues](#) in the JIRA administrator's documentation.

Configuring development tools

JIRA Software can be connected to a range of development tools to help you keep your project tracking in sync with your development work. Here are some of the features that you can unlock by connecting JIRA Software to different tools:

- Connect to repository manager, like **Bitbucket** or **GitHub**, and you can configure issues to automatically transition when events occur in the repository, e.g commit and branches created, pull requests merged and declined, etc. You can also create branches directly from issues in JIRA Software. In addition, every issue will show its related commits, branches, and pull requests.
- Connect to **Crucible** for code reviews, and you can configure issues to automatically transition when a review changes status. In addition, every issue will show its related reviews related to the issue.
- Connect to **Bamboo**, and every issue will show its related builds and deployments.

Connecting JIRA Software to your development tools

Before you begin:

- Make sure that you have the right permissions. You need to be a JIRA administrator to connect JIRA Software to another application. You will need to be an administrator for your development tool as well.

Follow the instructions on [Integrating with development tools](#) (*JIRA Admin documentation*) to connect JIRA Software to your development tools.

Configuring JIRA Software and your development tools

Some features will be automatically enabled when you connect JIRA Software to your development tools. Others require some additional configuration. Each feature is listed below, including the configuration work required to set it up.

Development panel on issues

The screenshot shows the JIRA Software interface with the 'Development' panel open for issue FUSE-113. The panel displays information such as branches (8 branches), commits (62 commits), pull requests (4 pull requests, one merged), and builds (1 build). It also shows a link to 'Deployed to Production' and a 'Create branch' button. A cursor is hovering over the 'Create branch' button.

The **Development panel** is shown on every issue, and helps you to evaluate the development status of an issue at a glance. If you need to investigate further, click any item to display more details, or drill down into the development tool.

You don't need to do any further configuration once you have connected JIRA Software to your development tools. However, your team will need to know how to reference issues in their development work.

Learn more: [Referencing issues in your development work](#)

View issue information in your developer tools

The screenshot shows the commit history for issue FUSE-113. It lists four commits from different authors: Albert Wang, Trey Shugart, and two commits from Albert Wang. Each commit includes the author, commit ID, message, commit date, and the number of issues it resolved.

Author	Commit	Message	Commit date	Issues
Albert Wang	dbd10dd047fb	Accidentally duplicated form-notification-test for Notification states changing, also fixed up jsf...	12 Aug 2014	
Albert Wang	ec656645907b	Merged in AUI-2694-quint-lt-mocha (pull request #1010) Merging QUnit to Mocha epic branch	12 Aug 2014	AUI-2694
Albert Wang	459716b5583	AUI-2694 - fixed merge conflicts and added async test	12 Aug 2014	AUI-2694
Trey Shugart	748db23bd1	Merged in AUI-2751-skate-0-9 (pull request #1007) Upgrade to Skate 0.9.	12 Aug 2014	AUI-2751

If you have connected JIRA to one of the supported developer tools, like Bitbucket, Bamboo, FishEye/Crucible, you will be able to view information about the issues in those tools. For example, in Bitbucket, you'll be able to see the issues that were referenced in a changeset; or in Bamboo, you'll be able to see the issues linked to a plan's builds.

Learn more: [Bitbucket Server](#), [Bitbucket Cloud](#), [Bamboo](#), [FishEye](#), [Crucible](#)

Create a branch from an issue

The screenshot shows the 'Create branch' dialog for issue FUSE-113. It asks where to create the branch and lists two options: 'Bitbucket Server' and 'Bitbucket Cloud'. A tooltip 'Choose repo Select the repository to create the branch in.' points to the repository selection dropdown. Another tooltip 'Create branch Click this link on your issue.' points to the 'Create branch' button at the bottom.

If you use Bitbucket Cloud or Bitbucket Server to manage your code repositories, you can create code branches directly from issues in JIRA Software via the **Create branch** link. The link will open your connected repository application and launch the process of creating a branch for you. If you have multiple applications connected, then you can choose where you'd like to create the branch.

Workflow triggers

You can configure triggers in workflows that respond to events in your linked development tools. This helps your team keep their issue statuses in sync

The screenshot shows the 'Workflows / TIS: Software Development Workflow (Draft)' configuration. A specific transition named 'Start Progress' is selected. The transition starts at 'TO DO' and ends at 'IN PROGRESS'. Below the transition, there are sections for 'Triggers', 'Conditions', 'Validators', and 'Post Functions'. A note states: 'Triggers are events which initiate an automatic workflow transition for a JIRA issue. Examples of triggers are pull request creation, or code review rejection.' Another note says: 'Conditions, validators, and permissions will be ignored for automatic transitions but operate as usual for manual transitions. Global transitions can be automated with triggers, if you follow some simple guidelines. Read our best practices page on automatic issue transitioning. Send feedback on triggers.' At the bottom, there's an 'Add trigger' button and a note about committing changes.

with their development work, rather than lagging behind. For example, when a developer creates a branch in Bitbucket to start work on an issue, they can reference the issue in the branch name, and the issue will automatically be transitioned from 'Open' to 'In progress'.

You will need to add triggers to your workflow(s) after connecting JIRA Software to your development tools. We recommend that you read our guide on configuring workflow triggers.

Learn more: [Configuring workflow triggers \(JIRA Admin documentation\)](#)

Development information in the Release Hub

The screenshot shows the 'Release Hub' board for 'Version 2.0'. It displays a summary of issues: 3 Warnings, 54 issues in version, 26 done, 8 in progress, and 20 to do. On the left, there's a sidebar with project navigation links like 'Stacking', 'Active sprints', 'Releases', 'Reports', 'Issues', 'Components', 'Project Shortcuts', 'Dev status repo', 'Bamboo Builds', 'Add link', 'Invite your team', and 'Give feedback'. The main board area shows a list of issues with columns for 'Assignee', 'Status', and 'Development'. A note at the bottom says: 'Warnings indicate when the status of a JIRA issue doesn't reflect related development activity. For example, an issue marked complete that has an open pull request should be marked as still being in progress.' There's also a section for 'Failing Builds'.

The Release Hub on your board lets you see the progress of your release (of a version) and determine what is likely to ship, at a glance. You can see all of the issues in the version, broken down by status category, as well as problems that could affect the release of the version (e.g. issues that should be in the release but are part of unmerged pulled requests, issues that haven't had code reviews, broken builds, etc).

You don't need to do any further configuration once you have connected JIRA Software to your development tools. However, your team will need to know how to reference issues in their development work.

Learn more: [Referencing issues in your development work](#)

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Configuring collaboration tools

Integrating collaboration tools with JIRA Software can help you plan and run your project more efficiently. Connect to HipChat (group and private chat), Confluence (wiki), or Team Calendars for Confluence to get issue notifications in your chat rooms, link project documentation to stories and epics, centralize your team leave, and more.

Connecting JIRA Software to your collaboration tools

Follow the instructions on [Integrating with collaboration tools \(JIRA Admin documentation\)](#) to connect JIRA Software to Confluence and/or HipChat.

Note:

- You need to be a JIRA administrator to connect JIRA Software to Confluence. You will need to be an administrator for Confluence as well.
- You need to be a JIRA administrator or project administrator to connect JIRA projects to HipChat rooms.

Feature overview

If you have JIRA Software, [HipChat](#), [Confluence](#), and [Team Calendars for Confluence](#), you can use them together to build stronger user stories and better plans for sprints or releases, while streamlining your team's work. Create meeting notes on the fly, wrap up a sprint with a retrospective report, and more.

Here are some of the features that you'll unlock by integrating JIRA Software and Confluence. For a full feature overview, see [Using JIRA applications with HipChat](#) and [Using JIRA applications with Confluence](#).

Plan epics

If you have linked your JIRA Software instance to a [Confluence](#) instance, you can create and link Confluence pages to your epics. For example, you may want to link your epic to a specification or design document in Confluence.

[Learn more...](#)

Plan sprints

Team Calendars for Confluence acts as your single source of truth for managing team leaves, tracking projects, and planning events. If you have Team Calendars installed in Confluence, planning your sprints around leaves and events is even easier. Add a sprint event to your team's calendar in Confluence, and as you plan your sprints, you will be able to see whether team leaves or events will impact your work.

[Learn more...](#)

Manage sprints

If you have linked your JIRA instance to a [Confluence](#) instance, you can create and link Confluence pages to your sprints via JIRA Software. For example, you may want to write up the sprint meeting notes in Confluence and link them to the sprint. You may also want to write your team's retrospective notes in Confluence and link them to the Sprint Report for the completed sprint.

[Learn more...](#)

If you have linked your project to HipChat as well, notifications will be posted in your chat room whenever issues are created or updated, helping everyone keep track of what's going on in the sprint.

[Learn more...](#)

Manage releases

The release is underway. Provide your stakeholders with updates on how the release is going, by creating JIRA Software status reports in Confluence. The JIRA Software status report provides a snapshot of the release at a point in time. Stakeholders can monitor progress, even if they don't have access to JIRA Software.

[Learn more...](#)

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Building a backlog

You've set up your project and configured your systems. It's time to prepare the backlog of issues that your team will work from.

A backlog is simply a list of features, which could be for your product, service, project, etc. These features are not detailed specifications. Rather, they are usually described in form of user stories, which are short summaries of the functionality from a particular user's perspective. This is a common template for a user story: *As a <type of user>, I want <goal> so that I <receive benefit>*. For example, "As a developer, I want to log time against my tickets in the issue tracker during a sprint, so that I can show my progress."

The documentation in this section will help you build a backlog in JIRA Software.

Search the topics in 'Building a backlog':

Overview

Building a backlog in a Scrum project

In Scrum, it is critical to have a prioritized backlog, so the highest priority items are ready to be included in the next sprint.

The **Backlog of your Scrum board** is designed to help you manage this. You can quickly add, update, and rank issues on the board to build the backlog. When it's time to start work, pushing issues from your backlog into sprints is as simple as dragging and dropping.

Learn more: [Using the backlog](#)

Building a backlog in a Kanban project

In Kanban, you should also prioritize the backlog so that high priority items can be quickly picked up by your team.

The **Backlog column of your Kanban board** allows you to easily add issues, rank them, and categorize them. For example, you may use swimlanes to group high priority issues for great visibility, then rank the rest of the issues manually.

Learn more: [Monitoring work in a Kanban project](#)

Already built a backlog?

Next: [Plan a new version](#)



Using the backlog

The **Backlog** of a Scrum board shows the issues for your project(s) grouped into a backlog and sprints. In the Backlog, you can create and update issues, drag and drop issues to rank them, or assign them to sprints, epics, or versions, manage epics, and more. You would typically use the Backlog when building a backlog, planning a new version, and planning a sprint.

- Before you begin
- About the Backlog
- Accessing the Backlog
- What can I do in the Backlog?
- Next steps

Before you begin

- The Backlog is not available for Kanban boards.
- Different functionalities in the Backlog require different permissions. For example, to start a sprint, you need to be a project administrator for all projects that match the board's filter. See [Permissions overview](#) for more information.

About the Backlog

An issue will only be visible in the Backlog if:

- the issue is not a sub-task,
- the issue matches the board's saved filter,
- the issue's status maps to one of the board's columns (but not the **Done** column), and
- there is at least a status being mapped to the **right-most** column.

*e.g. If you have the columns **To Do**, **In Progress**, and **Done**, ensure that you have a status mapped to **In Progress** at least. If you map all the statuses to the first column (**To Do**), you will not be able to see any issues in the **Backlog**.*

Accessing the Backlog

Navigate to the **Backlog** of your desired board.

Screenshot: the Backlog of a Scrum board (with an issue selected)

The screenshot shows the JIRA Backlog interface. On the left, there's a sidebar with 'VERSIONS' and 'EPICS' buttons. The main area shows a backlog of issues under 'Sprint 6'. One issue, 'TIS-109 Afterburner revision VI script', is selected and highlighted in green. To the right of the backlog, a detailed view of this issue is shown in a panel:

- Teams in Space / TIS-109**
- Afterburner revision VI script**
- Estimate:** 12
- Remaining:** Unestimated
- Details** section with fields: Status (OPEN), Component/s (None), Labels (None), Affects Version/s (1.8), Fix Version/s (None), and Epic (Afterburner Plus).

What can I do in the Backlog?

Task	Instructions
Add issues to the backlog	<p>Click Create in the header to open the 'Create Issue' dialog and create your issue. The issue will be added to the backlog under the currently selected issue, or at the top of the backlog if no issue is selected.</p> <p>Tip: Tick the Create another checkbox in the 'Create Issue' dialog to keep it open, if you are creating multiple issues.</p>
Prioritize the backlog	Drag and drop an issue to rank it. You can also right-click the issue to open a menu that allows you to send it to the top or the bottom of the backlog.
View an issue's details	<p>Click the desired issue on the board. The issue's details will display in a panel on the right of the board. For information on how to use this issue detail panel, see Searching for Issues.</p> <p>If you want to select an issue, rather than open the issue's details, use ctrl-left click or command-left click.</p>
Estimate stories	<p>Use the 'J' and 'K' keys to move through issues in the backlog and show the details on the right-hand side of the screen. Use the 'E' key to edit an issue, then update estimates or story points as you go.</p> <p>By default, the Story Points field is only available to issues of type 'Story' or 'Epic'. See Configuring estimation and tracking for details.</p>
Identify the workload for specialists	The avatars for users (specialists) who have work assigned to them in a sprint are shown at the top of a sprint. Click ... (next to avatars) to view the sprint workload for assignees.
Create sub-tasks	<p>Select the desired issue, press the '.' key, and type 'create sub-task' to open the 'Create Sub-task' dialog. Create the sub-task as desired.</p> <p>Sub-tasks are useful for breaking a story (issue) down into implementable chunks.</p>
Flag an issue	Right-click an issue to open a menu and select Add flag .

Filter issues	Click the desired Quick Filter (below the board name), e.g. Only My Issues . See Configuring Quick Filters for details.
Organize stories into epics	<p>Click EPICS (aligned vertically, left side of the board) to show the 'EPICS' panel. You can create epics, drag and drop issues into epics, and filter by epics via this panel.</p> <p>An epic is essentially a large user story, used for grouping smaller stories. For example, you may have a 'Performance' theme for your release, which you could capture as an epic.</p>
Plan versions	<p>Click VERSIONS (aligned vertically, left side of the board) to show the 'VERSIONS' panel. You can create and edit versions, assign issues to versions via drag-and-drop, and filter by versions via this panel.</p> <p>Note, you need to have at least one version in your project for this panel to display.</p>
Plan sprints	<p>Click Create Sprint, then drag and drop issues into your new sprint. You can also drag and drop the horizontal divider to add or remove multiple issues. The sprint footer will display the number of issues and total estimated work.</p> <p>When you're happy with the issues for the sprint, click Start Sprint, and the stories will move into the Active sprints.</p> <p>While a sprint is active in the Active sprints, you can still plan subsequent sprints in the Backlog, but you won't be able to start them until the active sprint is completed. (You can, however, drag and drop an issue in the Backlog onto the active sprint.)</p>

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Running sprints in a Scrum project
- Ranking an issue
- Estimating an issue
- Working with epics

Planning a version

You've got a backlog full of exciting stories that are just waiting to be turned into an amazing application. It's time to plan a version!

The documentation in this section will help you set up a new version for your JIRA Software project. This includes creating the version, setting up epics, and assigning work to a version.

Search the topics in 'Planning a version':

Overview

Creating and configuring a version

A version is a set of features and fixes released together as a single update to your application. Assigning issues to versions helps you plan the order in which new features (stories) for your application will be released to your customers.

Learn more: [Configuring versions in a Scrum project](#) and [Configuring versions in a Kanban project](#)

Working with epics (Scrum boards only)

An *epic* captures a large body of work. It is essentially a large user story that can be broken down into a number of smaller stories. It may take several sprints to complete an epic. You might use epics to capture broader themes in

Branching your repository

a release, e.g. performance-related work.

Learn more: [Working with epics](#)

You can use a number of different branching strategies, including task branching, feature branching, and release branching. You'll need to decide how much work should be contained in a branch before merging, when deciding on a strategy.

If you decide to use the release branching model, you'll create your branches at the start of a version. For example, you might create a branch for your last version, so you can continue with bugfix work on that branch, while new development work is done on the master branch.

Learn more: [Using branches in Bitbucket Server \(Bitbucket Server documentation\)](#), [Branching a repository \(Bitbucket Cloud documentation\)](#), [Creating and deleting branches \(GitHub documentation\)](#)

Tip: You can create a branch via an issue, if JIRA Software is [connected to Bitbucket Cloud, Bitbucket Server, or GitHub](#).

Already planned a version?

Next: Get to work!



Configuring versions in a Scrum project

A version is a set of features and fixes released together as a single update to your application. Assigning issues to versions helps you plan the order in which new features (stories) for your application will be released to your customers. In JIRA Software, you can view your issues according to which version they belong to. You can also drag-and-drop issues into a version to assign them to it, which you should do before you start work on the issues. This helps you plan your upcoming versions, which may span multiple sprints.

On this page:

- Before you begin
- Managing versions
- Tips
- Next steps

Before you begin

- This page does not apply to Kanban projects. See [Configuring versions in a Kanban project](#) instead.
- The functions for managing versions require different permissions. For example, you need the 'Edit Issues' permission to add an issue to a version. Read [Permissions overview](#) for more information.

Managing versions

1. Navigate to your board.
2. Click **Backlog**. If the Versions panel is not showing, click **VERSIONS** on the left side of the board (aligned vertically) to open it.

Add a new version

Click **Create version** (you will need to hover over the 'VERSIONS' panel to show this link), enter the version details, and create it.

* The **Start Date** is used to give you a more accurate [Version 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.

* The **End Date** is used to calculate the days remaining in a release on the [Release Hub](#).

Update a version's details	For the version name, click the arrow next to the name, then choose Edit name . For other fields (e.g. Description), click the field to edit it.
Add an issue to a version	Drag and drop the issue onto the version in the 'VERSIONS' panel.
Remove an issue from a version	Drag and drop the issue onto Issues without versions in the 'VERSIONS' panel.
Filter issues by version	Click the version in the 'VERSIONS' panel to show only issues in that version. Click All issues to remove the filter. <i>Alternatively, click Clear all filters next to the sprint's name.</i>

Screenshot: viewing versions in the Backlog

The screenshot shows the JIRA Software Backlog page. On the left, there is a sidebar titled 'VERSIONS' with a dropdown menu. The dropdown menu has several items: 'All issues' (selected), '1.9', '2.0', '2.1', '2.2', '3.0', and 'Issues without versions'. To the right of the sidebar, there is a main area titled 'Sprint 6' which contains '12 issues'. Below this, there is a date range '19/Jun/15 10:05 PM • 03/Jul/15 10:05 PM'. Underneath the date range, there is a list of issues with their titles, status, assignee, reporter, and labels. Some issues have a green progress bar under them. At the bottom right of the main area, there are three buttons: 'Afterburner Plus' (12), 'Exploration' (12), and 'Afterburner Plus' (2).

Tips

- The blue horizontal bar under the version's name (in the 'VERSIONS' panel) indicates progress towards completing the work estimated for the version. Note that this bar only represents progress on the estimated issues in a version.
- You can use the JQL `fixVersion` field to search for a version's issues — see [JQL \(JIRA Admin documentation\)](#) for details.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Using the backlog](#)
- [Version Report](#)
- [Deploying a release](#)

Configuring versions in a Kanban project

A version is a set of features and fixes released together as a single update to your product. Kanban boards do not require issues to be pre-assigned to versions (unlike Scrum boards which do require issues to be pre-assigned). This is because Kanban is designed for a continuous flow of work, rather than set iterations. For more information about Kanban versus Scrum, see [Configuring a project](#).

On a Kanban board, you can choose to release a version at any point in time — the version will contain all issues that are complete at that time. You also specify the name of the version at the time of the release.

For information on how to release a version on a Kanban board, see [Deploying a release](#).

Working with epics

An epic captures a large body of work, e.g. performance-related work in a release. It is essentially a large user story that can be broken down into a number of smaller stories. An epic can span more than one project, if multiple projects are [included in the board](#) to which the epic belongs.

Unlike sprints, scope change in epics is a natural aspect of agile development. Epics are almost always delivered over a set of sprints. As a team learns more about an epic through development and customer feedback, some user stories will be added and removed to optimize the team's release time.

Before you begin

- The functions for managing epics require different permissions. For example, you need the 'Edit Issues' permission to add an issue to an epic. Read [Permissions overview](#) for more information.

On this page:

- Before you begin
- Managing epics
- Linking a Confluence page to an Epic
- Tips
- Next steps

Managing epics

To manage epics in a Scrum project:

1. Navigate to the **Backlog** of your desired board.
2. If the Epics panel is not showing, click **EPICS** on the left side of the board (aligned vertically) to open it.

Add a new epic	Click Create epic (you will need to hover over the 'EPICS' panel to show this link), enter the epic details, and create it.
Update an epic's details	For the epic name, click the arrow next to the name, then choose Edit name . For other fields, click the arrow next to the epic's name, then choose View epic details . You can then edit the epic like any other issue.
Change an epic's color on the board	Click the arrow next to the epic's name, then choose the color from the menu.
Add an issue to an epic	To add an existing issue, drag and drop the issue onto the epic in the 'EPICS' panel. To add a new issue, click Create issue in epic (if this does not show, you need to expand the epic details in the 'EPICS' panel).
Remove an issue from an epic	Drag and drop the issue onto Issues without epics in the 'EPICS' panel. <i>Alternatively, view the detailed view of the issue on the board, locate the Epic field, then click the 'x' in the epic name lozenge (this method also works in the Active sprints of a board).</i>
Filter issues by epic	Click the epic in the 'EPICS' panel to show only issues in that epic. Click All issues to remove the filter. <i>Alternatively, click Clear all filters next to the sprint's name.</i>
Complete an epic	Click the arrow next to the epic's name, then choose Mark as Done from the menu. <i>This will set the epic's Epic Status field to "Done", but will not affect the epic's workflow or its Status field, and none of the epic's issues will be affected.</i>

To manage epics in a Kanban project:

1. Configure your Kanban board to base the swimlanes on epics. See [Configuring swimlanes](#) for details.
2. View your board. Every epic that has issues is shown as a swimlane on the Kanban board.
3. If you want to view the epic itself, open any issue in the epic in a new browser tab (i.e. not in the issue

detail view on the board), then click the lozenge in the **Epic link** field of the issue.

Screenshot: viewing an epic in Backlog

The screenshot shows the JIRA Backlog board. On the left, there are filters for Product, UI, Server, Only My Issues, and Recently Updated. The main area displays two sprints:

- Sprint 6**: Contains 6 issues. Issues include:
 - TIS-109 Afterburner revision VI script (Afterburner Plus)
 - TIS-110 Afterburner revision VI automation (Exploration)
 - TIS-80 As an operator I want to have a turbo button (Exploration)
 - TIS-57 Buttons need to be red
 - TIS-68 Homepage footer uses an inline style - should use a class
 - TIS-49 Draft network plan for Mars Office
- Sprint 7**: Contains 1 issue, TIS-8.

Screenshot: viewing epic details in the Issue Detail view

The screenshot shows the JIRA Issue Detail view for epic TIS-128. The details section includes:

- Type: Epic
- Priority: Major
- Affects Version/s: None
- Component/s: None
- Labels: None
- Epic Name: Exploration

The description section says "Click to add description". The "Issues in Epic" section lists:

- TIS-110 Afterburner revision VI automation (Status: IN PROGRESS, Reporter: Harvey Jennings)
- TIS-80 As an operator I want to have a turbo button (Status: OPEN, Assignee: Alana Grant)

Other sections include People (Assignee: Alana Grant, Reporter: Alana Grant, Votes: 0, Watchers: 1), Dates (Created: 13/Jul/15 11:55 AM, Updated: 13/Jul/15 11:57 AM), Development (Create branch), and Agile (View on Board).

Linking a Confluence page to an Epic

If you have linked your JIRA Software instance to a Confluence instance, you can create and link Confluence pages to your epics. For example, you may want to link your epic to a specification or design document in Confluence.

Learn more: [Linking a Confluence page to an epic](#)

Tips

- You can also use the JQL `Epic Link` field to search for an epic's issues — see [JQL](#) for details.
- There is no way to reopen an epic from the board. However, if you are a **JIRA admin**, you can reopen an epic via the following workaround:
 1. Add the **Epic Status** field to the **View** and **Edit** screens of your project.

2. View the epic issue in JIRA.
3. Edit the epic issue and set **Epic Status** to either "To Do" or "In Progress".
4. (optional) Undo step 1.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Using the backlog
- Epic Report

Linking a Confluence page to an epic

If you have linked your JIRA Software instance to a Confluence instance, you can create and link Confluence pages to your epics. For example, you may want to link your epic to a specification or design document in Confluence.

On this page:

- Creating a new linked Confluence page for an epic
- Linking an existing Confluence page to an epic
- Next steps

Screenshot: linked pages for an epic (Scrum board)

The screenshot shows a JIRA Scrum board titled 'Backlog'. On the left, there's a sidebar with 'EPICS' selected. A specific epic, 'Afterburner Plus', is highlighted. Below it, under 'Issues', there's a card for 'TIS-111 Afterburner reporting capability'. At the bottom of this card, there's a 'Linked pages' section. A modal window titled 'Linked pages' is open, listing three items: 'Teams in Space Marketing Launch', 'Teams in Space Roadmap', and 'Teams in Space Home'. There are 'Link page' and 'Create page' buttons at the bottom of the modal. The main board area shows a sprint summary for 'Sprint 6' with 12 tasks, 12 in progress, and 5 completed. The date range is 19/Jun/15 10:05 PM - 03/Jul/15 10:05 PM. The board has 'Board' and 'Up' buttons in the top right corner.

Creating a new linked Confluence page for an epic

Note, these instructions only apply to Scrum boards.

1. Log in to JIRA.
2. Click **Boards** (in header) > select your desired board.
3. Click **Backlog** and click **EPICS** (aligned vertically, left side of the board) to show the 'EPICS' panel.
4. Click **Linked pages** at the bottom of the epic details in the **Epics** panel.
5. Click **Create page**. The 'Create' dialog in Confluence will be displayed, with the **Product Requirements blueprint** selected (select a different blueprint, if you don't want to use it).

Note, you cannot create a new linked Confluence page when viewing an epic in JIRA Software, although you

can link an existing Confluence page (see instructions below).

Linking an existing Confluence page to an epic

... in Backlog (Scrum boards only):

1. Log in to JIRA.
2. Click **Boards** (in header) > select your desired board.
3. Click **Backlog** and click **EPICS** (aligned vertically, left side of the board) to show the 'EPICS' panel.
4. Click **Linked pages** at the bottom of the epic details in the **Epics** panel.
5. Click **Link page**, search for the page you want, and select it.

... when viewing the epic in JIRA Software (Scrum and Kanban boards):

1. View the issue or story in JIRA. Note, you need to view the issue in its own browser tab, i.e. not in the issue detail view on the board.
2. Click **More** > **Link**. This displays the Link dialog.
3. Click the **Confluence Page** tab in the dialog, and enter the details for the page you want to link to.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read these related topics:

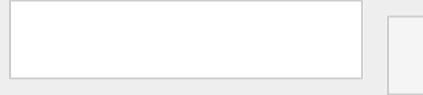
- [Using JIRA applications with Confluence](#)
- [Working with epics](#)

Getting to work

Ready to start working? You should have built a backlog and planned your version and epics by this stage. Time to get your team working!

The documentation in this section will help you use JIRA Software to coordinate your team's work. For Scrum, these tasks include planning and running sprints. For Kanban, these tasks include monitoring work in progress. Both Scrum and Kanban use boards (Active sprints/Kanban board) and wallboards to keep your team on track.

Search the topics in 'Getting to work':



Overview

Working in a Scrum project

Scrum projects are based around working in sprints. A sprint is an iteration of work (two to four weeks) that is typically part of a version. Each sprint starts with a planning meeting where the work is estimated and assigned to the sprint. At the end of each sprint, a retrospective or review meeting is held before the next sprint is planned.

In JIRA Software, sprints are planned using the **Backlog** of a board. Once a sprint has started, the sprint is monitored using the **Active sprints** of a board.

Learn more: [Running sprints in a Scrum project](#), [Checking the progress of a version](#), and [Using Active sprints](#)

Working in Kanban project

Unlike Scrum, Kanban doesn't use the concept of sprints. Rather, a Kanban project is based on the continuous delivery of work. When tasks are completed, more tasks are pulled into the work-in-progress pipeline. Constraints are placed on the amount of work that can be done at one time, so that the team is not overburdened.

In JIRA Software, work is managed and monitored using the **Kanban board**

Learn more: [Monitoring work in a Kanban project](#)

Already finished work for the version?

Next: [Release your version](#)



Running sprints in a Scrum project

A *sprint* — also known as an *iteration* — is a short period in which the development team implements and delivers a discrete and potentially shippable application increment, e.g. a working milestone version. If you haven't run sprints before, we recommend using a fixed two-week duration for each sprint. It's long enough to get something accomplished, but not so long that the team isn't getting regular feedback.

Note, sprints do not apply to Kanban projects.

In JIRA Software, you view sprints on a board and assign issues to sprints. You can search for issues in upcoming sprints using [JQL \(Sprint field\)](#). You can view the **Sprint** field on each individual issue as well, to see the sprint that an issue is part of.

To learn more about running sprints, read the following topics:

- To learn how to create and modify a sprint, add issues to a sprint, and start a sprint, see [Planning sprints](#).
- To find out how you can use Active sprints, wallboards, reports, and more, to track a sprint that is in progress, see [Monitoring the progress of a sprint](#).
- To learn how to close a sprint and complete related tasks, like reporting, see [Completing a sprint](#).
- To learn how to reopen a sprint that's already closed, see [Reopening a sprint](#).
- To see how you can link Confluence pages (e.g. 'Retrospective notes') to a sprint, see [Linking a Confluence page to a sprint](#).

Planning sprints

Every sprint starts with a planning meeting. When planning a sprint, your team would typically commit to deliver a set of stories that are pulled from the top of the backlog. In JIRA Software, this involves creating a sprint, assigning stories to the sprint, and starting the sprint. The instructions on this page will help you complete these activities.

Before you begin

- Sprints only apply to Scrum boards.
- You must have ranking enabled on your board to use sprints. See [Enabling ranking](#).
- In general, sprint actions require the Manage Sprints permission. However, there are some sprint actions (e.g. adding issues to sprints, removing issues from sprints) that require the Schedule Issues and Edit Issues permissions. See [Permissions overview](#) for more information.

On this page:

- Before you begin
- Creating a sprint
- Adding issues to a sprint
- Starting a sprint
- Editing, reordering, or deleting a sprint
- Viewing the issues in a sprint
- Next steps

Creating a sprint

You can create a sprint for your current iteration, or multiple future sprints if you want to plan several iterations ahead.

1. Navigate to the **Backlog** of your desired board.
2. Click the **Create Sprint** button at the top of the backlog.

Once you have created a sprint, you can add issues to it, as described in 'Adding issues to a sprint' below.

 **Tip:** Need to record your planning meeting notes somewhere? If you have JIRA Software connected to Confluence, you can create a 'Meeting Notes' page via the [Linked pages](#) link. See [Linking a Confluence page to a sprint](#) for details.

Adding issues to a sprint

In Scrum, scope creep during a sprint should be avoided. In addition, the team is supposed to deliver a working piece of software at the end of the sprint. This means that you need to know your team's capacity for work, as well as the amount of work they are committing to, when adding issues to a sprint.

Typically, your team would estimate issues before adding them to the sprint, so that you can see the total estimated work for the sprint (in the sprint footer). You can match this against your team's capacity for work by looking at past sprints. Tools like the [Velocity Chart](#) and [Burndown Chart](#) can help you with this. Don't worry if you don't have any historical data — you'll get a good idea of your team's velocity once they start completing sprints.

Note:

- An issue cannot belong to more than one sprint.
- A sprint can include issues from more than one project, if the board filter includes multiple projects.
- If you add an issue to a sprint on one board, that particular sprint will also appear on all other boards that contain that particular issue.
- Sub-tasks cannot be moved independently of their parents.

There are three ways to add an issue to a sprint:

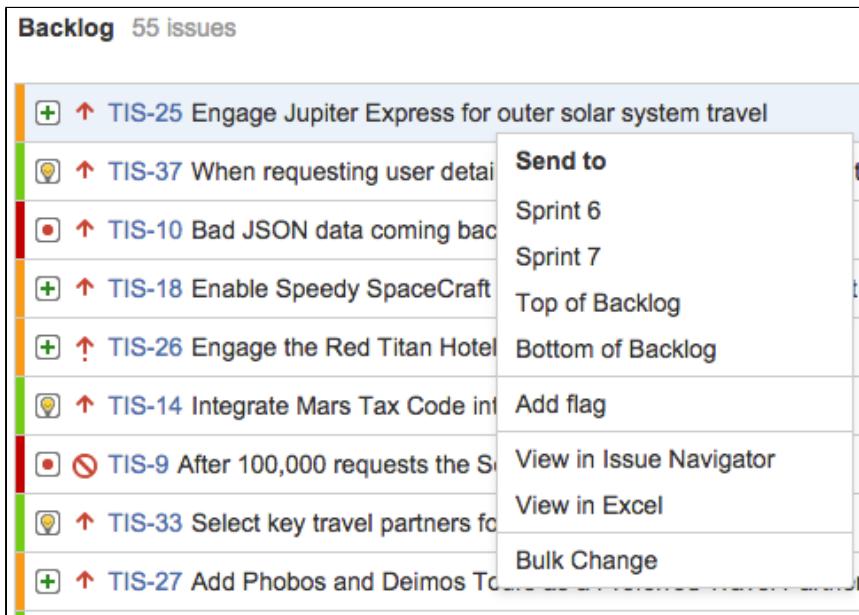
▼ [Add existing issues to a sprint...](#)

Use this when the issue already exists, and you want to add the issue to an active or future sprint.

In the **Backlog**, drag and drop the issues onto the relevant sprint.

If you want to add multiple issues you can either:

- select the issue(s) (use Shift+Click or Ctrl+Click), right-click, then select the relevant sprint, or
- drag the sprint footer down to include issues from the backlog.



The screenshot shows the JIRA Backlog screen with 55 issues listed. The first issue, TIS-25, is selected and has a context menu open. The menu options are:

- Send to: Sprint 6, Sprint 7, Top of Backlog, Bottom of Backlog
- Add flag
- View in Issue Navigator
- View in Excel
- Bulk Change

▼ [Create then add an issue to an active sprint...](#)

Use this when the issue doesn't exist, and you want to quickly add it to an active sprint.

After creating an issue in the **Active sprints**, click the **Add to <sprint name>** link in the confirmation dialog that displays. **<Sprint name>** will be the name of the sprint that you are currently viewing on the board.

Note, if you do not have the 'Edit Issues' and 'Schedule Issues' permissions for all projects included by

the board's filter, the issue will be added to the backlog instead of the sprint.



Edit the Sprint field for an issue...

Use this when editing an issue, and you know the name of the sprint (active and future sprints only).

Create or edit an issue and enter the sprint name in the **Sprint** field. If the Sprint field doesn't display on the Create Issue or Edit Issue dialog, choose **Configure Fields**, then select the **Sprint** field.

The screenshot shows the 'Edit Issue : TIS-49' dialog. The 'Sprint' field dropdown is open, displaying three suggestions: 'Sprint 6 (Active sprint)', 'Sprint 6 (Active sprint)', and 'Sprint 7 (Future sprint)'. The first suggestion is highlighted with a blue background.

Note, if you update the Sprint field for multiple issues via [bulk operations](#), you need to enter the Sprint ID, not Sprint name. To find the Sprint ID, navigate to an issue in the sprint, hover over the sprint name, and look in the URL for number in the **sprint** parameter.

If you want to remove an issue from a sprint, just drag it to the **Backlog**, or right-click and select **Send to Top of Backlog/Bottom of Backlog**. You can also edit or bulk edit issues to remove the value from the Sprint field of the issues.

Once you have chosen which issues to include in your next sprint, you are ready to start the sprint, as described in the 'Starting a sprint' section below.

Starting a sprint

You can only start a sprint, if:

- You haven't started one already. If you want to have more than one active sprint at a time, try the [Parallel Sprints](#) feature.
- The sprint is at the top of the backlog. If you want to start a planned sprint that is lower down, you will need to reorder your sprints to move it to the top. (Note, this doesn't apply if Parallel Sprints is enabled).

To start a sprint:

1. Navigate to the **Backlog** of your desired board.
2. Find the sprint that you want to start and click **Start Sprint**.
If you haven't estimated your issues, you will receive a warning when starting the sprint (unless you are using Issue Count, as this is calculated automatically).
3. Update the **Sprint Name**, if desired, and select the **Start Date** and **End Date** for the sprint.
Note, the default duration of a sprint is two weeks.

You will be taken to the **Active sprints**, where issues in your newly started sprint will be shown.

You can still view your active sprint in the **Backlog**, and even add and remove issues from it. Be aware that

adding and removing an issue from the active sprint is considered to be a 'scope change' and will be reflected in some reports, e.g. the Burndown Chart.

Editing, reordering, or deleting a sprint

Sometimes, you may need to modify sprints after they have been created, particularly if you are managing a large backlog. You can edit the name and dates of a sprint, reorder a sprint, or delete a sprint in the Backlog.

To edit the name or dates for a sprint, click on the sprint name or date fields to edit them. Note, you can only edit sprints that have not been completed yet. Also, the date fields are not specified until you start a sprint.

To delete a sprint, click the



icon (next to **Start Sprint**) for the sprint that you want to move, then click **Delete sprint**. Note, you cannot delete the active sprint, you can only complete it. You also cannot delete closed (i.e. completed) sprints.

To reorder a sprint, click the



icon (next to **Start Sprint**) for the sprint that you want to move, then choose **Move sprint up** or **Move sprint down**. Note, you can only reorder planned sprints (not the active sprint or closed sprints). Also, if you have the same sprints (or some subset of them) on different boards, the sprint order will be reflected across the boards.

Viewing the issues in a sprint

Use the Backlog of a board to view planned sprints. If you want to view a sprint in progress, use the Active sprints of a board instead. You can also use **JQL** in the issue search (**Issues > Search for issues > Advanced**) to search for a sprint's issues.

For example, to search for issues that are in the sprint, "February 1", use this query:

```
sprint = "February 1"
```

For more information on how to search for issues by sprint:

- See the [JQL](#) documentation on the `sprint` field ('Fields Reference' section),
- See [Advanced searching functions](#) on the `closedSprints()` and `openSprints()` functions.

Next steps

Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Using the backlog](#)
- [Estimating an issue](#)
- [Burndown Chart](#)
- [Monitoring the progress of a sprint](#)
- [Completing a sprint](#)

Using Active sprints

The Active sprints of a Scrum board displays the issues that your team is currently working on. You can create and update issues, and drag and drop issues to transition them through a workflow.

Before you begin

- Active sprints are only available on Scrum boards. If you are using a Kanban board, see [Monitoring work in a Kanban project](#) instead.
- Different functions on the Active sprints of a board require different permissions. For example, to remove an issue from a sprint, you

need to have the Schedule Issues and Edit Issues permissions. See [Permissions overview](#) for more information.

On this page:

- Before you begin
- Using Active sprints of a Scrum board
- What can I do in the Active sprints of a Scrum board?
- Next steps

Using Active sprints of a Scrum board

Navigate to the **Active sprints** of your desired board.

Screenshot: Active sprints of a Scrum board

Active sprints: Sprint 6			
QUICK FILTERS: Product UI Server Only My Issues Recently Updated			
2 To Do	2 In Progress Max 3	1 Code Review	2 Done
TIS-109 Afterburner revision VI script TIS-80 As an operator I want to have a turbo button	TIS-110 Afterburner revision VI automation TIS-68 Homepage footer uses an inline style - should use a class	TIS-57 Buttons need to be red	TIS-49 Draft network plan for Mars Office TIS-127 Initial draft for review

About Active sprints:

- An issue will only be visible in Active sprints if:
 - the issue's status maps to one of the board's columns,
 - the issue is in an active sprint (for Scrum boards), and
 - the issue matches the board's saved filter.
- Each column maps to one or more JIRA statuses.
- Sub-tasks are shown slightly indented, with their parent issue key shown above.
- Swinlanes are based on criteria of your choice. Issues hidden by a Quick Filter will be excluded from the swimlane count.
- The board will be refreshed every 30 seconds, if the browser window doesn't currently have focus. If you have focus on the window, it will show a message, asking you to refresh.
- If you've enabled the 'Days in column' indicator for your board (via [configuring columns](#)), dots will show at the bottom of each issue (up to the width of the card or a maximum of 32 dots). The dots indicate the number of days that the issue has been in its current column. Hover over the dots to see the total number of days.

[What can I do in the Active sprints of a Scrum board?](#)

Task	Instructions
------	--------------

Add issues to the 'To Do' column	Click Create in the header to open the 'Create Issue' dialog and create your issue. The issue will be added to the To Do column. Tip: tick the Create another checkbox in the 'Create Issue' dialog to keep it open, if you are creating multiple issues.
View an issue's details	Click the desired issue on the board. The issue's details will display in a panel on the right of the board. For information on how to use this issue detail panel, see Searching for Issues . You can also select multiple issues, right-click, then select View in Issue Navigator for a different view of the issues.
Prioritize an issue	Drag and drop an issue within a column to rank it.
Transition an issue	Drag and drop an issue between columns.
Create Sub-tasks	Select the desired issue, press the '.' key, and type 'create sub-task' to open the 'Create Sub-task' dialog. Create the sub-task as desired. Sub-tasks are useful for breaking a story (issue) down into implementable chunks.
Flag issues	Right-click (or select multiple issues and right-click) to open a menu and select Add flag .
Filter issues	Click the desired Quick Filter (below the board name), e.g. Only My Issues .
Bulk change issues	Right-click (or select multiple issues and right-click) to open a menu and select Bulk Change .
Remove issues from a sprint	Right-click (or select multiple issues and right-click) to open a menu and select Remove from Sprint .

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read these related topics:

- [Configuring columns](#)
- [Configuring filters](#)
- [Configuring Quick Filters](#)
- [Transitioning an issue](#)
- [Configuring swimlanes](#)

Using Parallel Sprints

The Parallel Sprints feature lets you enable multiple active, parallel sprints. For example, if you have two teams working from the same backlog, each team can now work on their own sprint.

Please note the following caveats when using Parallel Sprints:

- The Velocity Chart will not show the velocity per team.
- The current implementation assumes that the teams perform estimation identically, which is unlikely in practice.

Enabling Parallel Sprints

1. Log in as a user with the 'JIRA Administrators' [global permission](#).
2. Select **JIRA Administration** from the top bar > **Applications**, then scroll down the page to the **JIRA Software** section.

3. Under **JIRA Software configuration**, select the **Parallel Sprints** checkbox.

Monitoring the progress of a sprint

You've started a sprint with grand plans. How do you make sure that your team is on track to deliver what they've committed to?

JIRA Software provides a number of tools that you can use to monitor the progress of a sprint. If you run into problems or things are just not on track for the target date, JIRA Software can help visualize or highlight problems so your team can take actions to remedy them.

On this page:

- Active sprints
- Development panel for issues
- Wallboards and gadgets
- Burndown chart
- Real-time chat notifications for your project

Active sprints

The Active sprints of a scrum board just may be the central workspace for a sprint that is in progress. You and your team would usually check Active sprints at least once a day, while a sprint is running — perhaps your team has its daily stand-ups around a monitor displaying Active sprints.

Learn more: [Using Active sprints](#)

Active sprints: Sprint 6			
0 days remaining Complete Sprint Board X			
QUICK FILTERS: Product UI Server Only My Issues Recently Updated			
2 To Do	2 In Progress Max 3	1 Code Review	8 Done
TIS-109 Afterburner revision VI script Afterburne... 12	TIS-110 Afterburner revision VI automation Exploratio... 12	TIS-57 Buttons need to be red Exploratio...	TIS-45 Email non registered users to sign 2
TIS-80 As an operator I want to have Exploratio...	TIS-68 Homepage footer uses an inline		TIS-30 Create Saturn Summer 2
			TIS-49 Draft network plan for Mars 5
			TIS-127 Initial draft for review

Teams in Space / TIS-110 [...](#) [X](#)

Afterburner revision VI automation

Estimate: 12

Details

Status: **IN PROGRESS**

Component/s: None

Labels: None

Affects Version/s: None

Fix Version/s: None

Epic: **Exploration** [X](#)

Development panel for issues

If you have JIRA Software connected to your development tools, a 'Development' panel is shown on each issue that shows you information about the commits, branches, pull requests, builds, and more, that are related to the issue. This can help you see discrepancies at a glance — for example, issues that have related commits, but are still in the 'To Do' status.

Learn more: [Configuring development tools](#)

The screenshot shows a JIRA wallboard with a sidebar on the left listing four issues: TIS-45, TIS-30, TIS-49, and TIS-127. TIS-45 is expanded to show its details. The details pane for TIS-45 includes:

- Summary:** Teams In Space / TIS-45
- Description:** Email non registered users to sign up with Teams In Space
- Estimate:** 2
- Development:**
 - 2 branches
 - 1 commit
 - 1 pull request MERGED
 - 1 build ✓
- Actions:** Create branch

Wallboards and gadgets

A wallboard can act as an "information radiator" for your team, letting them know how they are progressing during a sprint, alerting them when builds are broken, reminding them about upcoming team events, and more. You just need to add the desired gadgets to a dashboard, then display it as a wallboard (**Tools > View as Wallboard** on the dashboard).

Learn more: [Configuring dashboards](#)

The screenshot shows a JIRA dashboard with a burndown chart on the left and a list of recent builds on the right.

Burndown Chart Data:

Category	Value
Overall sprint progress (Story Points)	5 / 13
Time elapsed	91%
Work complete	72%
Scope change	0%
Blocker	0
Flagged	0

Recent Builds (Today):

- Teams In Space > Apollo UI > TIS-64-a-demo-bug, #33 > Integration Tests was successful. Scheduled build, build took 8 seconds.
- Teams In Space > Apollo UI > TIS-64-a-demo-bug, #33 > Functional Tests was successful. Scheduled build, build took 8 seconds.
- Teams In Space > Apollo UI > #57 > Functional Tests was successful. Scheduled build, build took 8 seconds.
- Teams In Space > Apollo UI > #57 > Integration Tests was successful. Scheduled build, build took 8 seconds.
- Teams In Space > Apollo UI > TIS-64-a-demo-bug, #33 > Package was successful. Scheduled build, build took 10 seconds.

Burndown chart

The Burndown chart shows the work that has been completed in a sprint and estimates how long it will take to complete the remaining work. This can help you determine the likelihood of your team completing a sprint on time — allowing you to visualize or highlight problems so your team can take actions to remedy them.

Learn more: [Burndown Chart](#)



Real-time chat notifications for your project

If you really want to keep your finger on the pulse for your sprint, connect your project to your team's HipChat room. Notifications will be sent to the room whenever an issue is created or updated (filtered according to your preference), helping everyone keep track of what's going on.

If you want to discuss an issue in more detail, you can even create a HipChat room on the fly, directly from the issue. This will associate the current issue and the room, and any changes to the issue will send a notification to that room.

Learn more: [Using JIRA applications with HipChat](#)

Completing a sprint

The end of a sprint is usually the time where your team takes stock of its progress. This usually includes demonstrations of the work completed during the sprint, followed by a sprint retrospective to analyze where improvements can be made.

As the team lead, Scrum master, or product owner, you can also use this time to check how your team is progressing against the overall version, and provide feedback to your stakeholders.

On this page:

- Before you begin
- Completing the active sprint
- Other tasks to complete at the end of a sprint
- Next steps

Before you begin

- Sprints only apply to Scrum boards. If you are using a Kanban board, you may want to read this instead: [Deploying a release](#).
- To complete a sprint, you must be a JIRA Administrator or a user with the [Manage Sprints](#) permission.

If you can't complete a sprint, your board's filter may be complex, and JIRA Software is unable to determine which projects will be returned by the query. See [Using Manage Sprints permission for advanced cases](#) if you need help.

Completing the active sprint

1. Navigate to the **Active sprints** of your desired board.
2. If necessary, select the sprint you want to complete from the sprint drop-down.
3. Click **Complete Sprint**.

If you have multiple sprints in the Active sprints of your board, the 'Complete Sprint' button will not appear until you select one of the multiple sprints.

When you complete the sprint, the following actions will occur:

- Your completed issues will move out of Active sprints.
- If you have future sprints, you can move any issues not completed at the end of the sprint to either one of the future sprints or the Backlog.
- If you don't have any future sprints, any issues not completed at the end of the sprint will be returned to the Backlog.
- For sprints that have parent issues and sub-tasks:
 - If you have parent issues that are 'Done' but sub-tasks that are not 'Done', you won't be able to end the sprint. You must complete the sub-tasks first.
 - If you have parent issues that are not 'Done' but have sub-tasks that are all 'Done', the parent issues will still be moved to the selected future sprint or to the Backlog. If these parent issues are part of another active sprint, the previously completed sub-tasks are still 'Done'.
- Your issues won't be marked with the date the sprint was closed; however, you can always view the sprint for an issue to find out when the sprint ended.

Other tasks to complete at the end of a sprint

Conduct a retrospective and link your notes to the sprint

Typically, at the end of a sprint, you conduct a sprint retrospective. This is a meeting where your team discusses the completed sprint, including the things they need to start doing, stop doing, or continue doing. The outcome of this meeting should be actions for improvements in future sprints.

When you complete a sprint in JIRA Software, you will be shown the 'Sprint Report', which provides helpful data about your sprint for your team's retrospective. If you have Confluence linked to JIRA Software, you can also create your retrospective notes as Confluence pages and link them to your Sprint Report. See [Linking a Confluence page to a sprint](#) for more information.

Check the progress of your version

The end of a sprint is a great time to check how your version is progressing. See [Checking the progress of a version](#) for more information.

Release the sprint as a version (optional)

Your team may not need to release a version at the end of a sprint — but if you need to, it's easy to do. In the **Completed Issues** section of the Sprint Report, just click **View in Issue Navigator**. You should have already pre-assigned issues to a version before starting work on them, (which results in better reports). However, if you need to specify a version for issues at this stage, use JIRA's **Bulk Edit** to assign all of the issues to the relevant version (see [Editing multiple issues at the same time](#) for more information). Note that you will not be able to do this if your 'Done' column sets an issue's status to 'Closed', as issues are not editable once they are 'Closed' (but 'Resolved' is fine). Also, note you can only **Bulk Edit** the 'Fix Version' for issues from one project at a time.

Start planning your next sprint!

Once you have completed a sprint in JIRA Software, you can start getting ready for a new one.

Note, you can only have one active sprint at a time, unless you've enabled [Parallel Sprints](#).

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Sprint Report](#)
- [Planning sprints](#)
- [Reopening a sprint](#)

Reopening a sprint

JIRA Software lets you reopen sprints that have been previously closed.

This is a handy feature when:

- You've mistakenly completed the wrong sprint.
- You completed the correct sprint, but you completed it too early.

Moreover, sprints that are mistakenly closed, or closed too early will affect your team's velocity, as the data in these sprints will still be included in your board. Your Velocity Chart will then be reflecting inaccurate data.

On this page:

- Before you begin
- General points when reopening sprints
- Scenarios where you can reopen sprints
- Reopening a sprint
- Next steps

Before you begin

- Sprints only apply to Scrum boards.
- To reopen a sprint, you must be a JIRA Administrator or a user with the [Manage Sprints](#) permission.

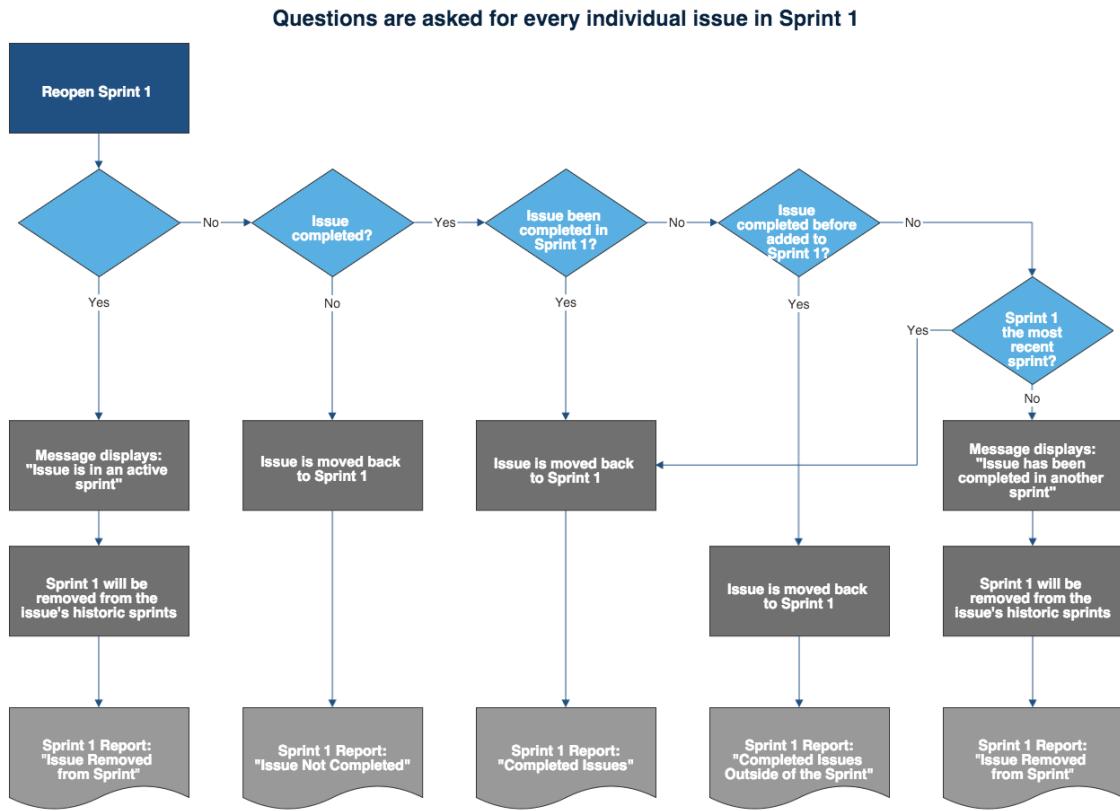
General points when reopening sprints

In general, it's worthy to keep these points in mind when you're trying to reopen a sprint:

- When the Parallel Sprints feature is not enabled and there is a currently active sprint, you cannot reopen the sprint you previously completed.
- When issues in the completed sprint have already been moved to other active sprints, you can still reopen the sprint, but it will not contain the issues that are already in the active sprints.
- For issues that are completed outside of a sprint, but are then pulled back into a sprint you just reopened, the Sprint Report for the opened sprint will show these issues as 'completed outside of this sprint'.

An in-depth look at reopening sprints

The diagram below shows you the possible flows when you're reopening sprints, depending on certain scenarios.



Scenarios where you can reopen sprints

In the following sample scenarios listed below, let's assume the following details:

- You've created a sprint and named it 'Sprint 1'.
- You added some issues to Sprint 1, and then started it.
- Your team is currently working on the issue in Sprint 1.
- You have a future planned sprint named 'Sprint 2'.

Simple scenarios

Below are some examples of simple scenarios where you may want to reopen a sprint, and the results you can expect.

Scenario details	Expected results
<ul style="list-style-type: none"> • You closed Sprint 1, and there were some completed issues in it. • All incomplete issues were moved back to the Backlog. • All these incomplete issues are still in the Backlog when you try to reopen Sprint 1. 	<ol style="list-style-type: none"> 1. Sprint 1 is reopened successfully. 2. Sprint 1 contains all the completed and incomplete issues that were in Sprint 1 when it was closed.
<ul style="list-style-type: none"> • You closed Sprint 1, and there were no completed issues in it. • All incomplete issues were moved back to the Backlog. • During sprint planning, you moved all of these issues to the future sprint, Sprint 2. • You started Sprint 2, and Sprint 2 is a currently active sprint when you try to reopen Sprint 1. • The Parallel Sprints feature is enabled. 	Sprint 1 cannot be reopened because all incomplete issues in Sprint 1 when it was closed are now in the currently active Sprint 2.

▼ [Complex scenarios](#)

In some cases, you may be able to reopen a sprint, but the results may not be as what you expect them to be. Or you may not be able to reopen a sprint altogether. Below are some examples of complex scenarios where you may want to reopen a sprint, and the results you can expect.

Scenario details	Expected results
<ul style="list-style-type: none"> You closed Sprint 1, and there were some completed issues in it. All incomplete issues were moved back to the Backlog. During sprint planning, you moved some of these issues to the future sprint, Sprint 2. You started Sprint 2, and Sprint 2 is a currently active sprint when you try to reopen Sprint 1. The Parallel Sprints feature is enabled. 	<ol style="list-style-type: none"> Sprint 1 is reopened successfully. Sprint 1 contains all the completed and incomplete issues that were in the Sprint 1 when it was closed. The incomplete issues that were in Sprint 1 when it was closed, but are currently in the active Sprint 2 will remain in Sprint 2. This is because issues in an active sprint are not moved during the reopen sprint process.
<ul style="list-style-type: none"> You closed Sprint 1, and there were some completed issues in it. All incomplete issues were moved back to the Backlog. During sprint planning, you moved some of these issues to the future sprint, Sprint 2. You started Sprint 2, and Sprint 2 is a currently active sprint when you try to reopen Sprint 1. The Parallel Sprints feature is not enabled. 	<p>Sprint 1 cannot be reopened for the following reasons:</p> <ul style="list-style-type: none"> There is a currently active sprint, Sprint 2. The Parallel Sprints feature is not enabled.
<ul style="list-style-type: none"> You closed Sprint 1, and there were no completed issues in it. All incomplete issues were moved back to the Backlog. During sprint planning, you moved all of these issues to the future sprint, Sprint 2. You started Sprint 2, and Sprint 2 is a currently active sprint when you try to reopen Sprint 1. The Parallel Sprints feature is enabled. 	<p>Sprint 1 cannot be reopened because all of its issues are now in the currently active sprint, Sprint 2.</p>

Reopening a sprint

- Click **Boards** (in header) > select your desired board.

2. Click **Reports**, then select **Sprint Report**.
3. Select the relevant sprint from the sprint drop-down.
4. Click **Reopen Sprint**. The Reopen Sprint dialog will be displayed, showing you the outcome you can expect when the sprint is reopened.
5. Click **Reopen**.

Note the following details that take place when a sprint is completed then reopened:

- When you complete a sprint, this event of completing the sprint will be recorded in the corresponding **Sprint Report**.
- When you reopen that sprint, this event of reopening the sprint will also be recorded in the Sprint Report. However, the report will now use the new end date moving forward.
- When the reopened sprint is completed, the last completion date will appear as the end date in the Sprint Report.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Sprint Report](#)
- [Planning sprints](#)
- [Completing a sprint](#)

Linking a Confluence page to a sprint

If you have linked your JIRA instance to a Confluence instance, you can create and link Confluence pages to your sprints via JIRA Software. For example, you may want to write up the sprint meeting notes in Confluence and link them to the sprint. You may also want to write your team's retrospective notes in Confluence and link them to the Sprint Report for the completed sprint.

Screenshot: linked pages for a sprint



On this page:

- Before you begin
- Creating a new linked Confluence page for a sprint
- Linking an existing Confluence page to a sprint
- Next steps

Before you begin

- These instructions only apply to Scrum boards.

Creating a new linked Confluence page for a sprint

... in Active sprints:

1. Navigate to the **Active sprints** of your desired board.
2. Find the sprint that you want to create a linked page for. Click **Linked pages** at the top right of the sprint.
3. Click **Create page**. The 'Create' dialog in Confluence will be displayed, with the **Meeting Notes** blueprint selected (select a different blueprint, if you don't want to use it).

... from the Sprint Report:

1. Navigate to the Sprint Report, as described above.
2. Click **Linked pages** above the report chart.
3. Click **Create page**. The 'Create' dialog in Confluence will be displayed, with the **Retrospectives** blueprint selected (select a different blueprint, if you don't want to use it).

Linking an existing Confluence page to a sprint

... in Active sprints:

1. Navigate to the **Active sprints** of your desired board.
2. Find the sprint that you want to create a linked page for. Click **Linked pages** at the top right of the sprint.
3. Click **Link page**, search for the page you want, and select it.

... from the Sprint Report:

1. Navigate to the Sprint Report, as described above.
2. Click **Linked pages** above the report chart.
3. Click **Link page**, search for the page you want, and select it.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Using JIRA applications with Confluence](#)
- [Creating issues and sub-tasks](#)
- [Working with epics](#)

Checking the progress of a version

You've been monitoring the progress of your sprints and keeping your team on track for each iteration. How do know if you are on track to deliver the version?

JIRA Software provides a number of tools that you can use to monitor the progress of a version. These tools can help you see problems early, as well as determine the likelihood of releasing a version on time.

On this page:

- [Before you begin](#)
- [Release Hub](#)
- [Release Burndown report](#)
- [Status report](#)

Before you begin

- This page only applies to Scrum projects.

Release Hub

The Release Hub helps you understand the status of work for your version at a point in time. It shows a summary and a breakdown of the issues in the version, grouped by status. In addition, if you have connected JIRA Software to your development tools, this page also warns you if your development tools are out of sync with JIRA Software — for example, completed issues with open pull requests, completed issues with failing builds, completed issues that have unreviewed commits, etc.

1. Navigate to your project.
2. Click **Releases** > your desired version.

Learn more: [Using the release page to check the progress of a version](#)

P	T	Key	Summary	Assignee	Status	Development
↑	BUG	TIS-40	Update FlightController to handle multiple travel providers in one reservation	Max Taylor	OPEN	2 branches
↑	BUG	TIS-41	Update LodgingController to handle multiple travel providers in one reservation	Max Taylor	CODE REVIEW	
↑	BUG	TIS-42	Extend booking experience in UI to include multiple hotels on one reservation	Kevin Campbell	OPEN	1 branch
↑	BUG	TIS-43	Extend booking experience in UI to include multiple flights on one reservation	Kevin Campbell	OPEN	2 branches
↑	BUG	TIS-44	Reward Customers an extra 5-10% when they book a large trip	William Smith	OPEN	2 branches
↑	BUG	TIS-45	Email non registered users to sign up with Teams In Space	Jennifer Evans	CLOSED	MERGED

Release Burndown report

The 'Release Burndown' report helps you understand how your team is progressing towards the completion of the version. The report shows your team's velocity (relative to the current version), how scope changes have affected progress, and the estimated number of sprints needed to complete the remaining work.

Learn more: [Release Burndown](#)



Status report

If you have Confluence linked to JIRA Software and you need to provide status reports to external stakeholders, you can generate a 'Status Report' in Confluence that displays the progress of a version. See [JIRA Report Blueprint \(Confluence documentation\)](#).

Using the release page to check the progress of a version

P	T	Key	Summary	Assignee	Status	Development
↑	BUG	TIS-40	Update FlightController to handle multiple travel providers in one reservation	Max Taylor	OPEN	1 branch
↑	BUG	TIS-41	Update LodgingController to handle multiple travel providers in one reservat...	Max Taylor	CODE REVIEW	
↑	BUG	TIS-42	Extend booking experience in UI to include multiple hotels on one reservation	Kevin Campbell	OPEN	1 branch
↑	BUG	TIS-43	Extend booking experience in UI to include multiple flights on one reservation	Kevin Campbell	OPEN	1 branch
↑	BUG	TIS-44	Reward Customers an extra 5-10% when they book a large trip	William Smith	OPEN	1 branch
↑	BUG	TIS-45	Email non registered users to sign up with Teams In Space	Jennifer Evans	CLOSED	MERGED

Screenshot: release page for a JIRA version

Frequently asked questions

How do I find the 'Release' page for a version?

Navigate to your project > **Releases** > your desired version.

What warnings are shown on the 'Warnings' tab?

Issue warnings tell you when your issue data is potentially out of sync with your development data. Depending on which development tools are connected to JIRA Software, the Release page will display the following warnings:

Warning type	Required development tool	When will a warning display?
Open pull requests	Bitbucket	An issue is done, but has an open pull request
Open reviews	Crucible	An issue is done, but has an open review
Unreviewed code	Crucible, Bitbucket	An issue is done, but its commits are not part of a pull request or code review
Failing builds	Bamboo	An issue is done, but is linked to a failing build

What statuses are considered to be 'To Do', 'In Progress', and 'Done'?

On the Release page for a version, issues are grouped into tabs, depending on whether they are 'To Do', 'In Progress', or 'Done'. Also, issue warnings are evaluated, depending on whether an issue is 'done' or not.

'To Do', 'In Progress', or 'Done' map to the status category of an issue's status. The status category of an status is defined when you create a status. See [Defining status field values](#) (JIRA Admin documentation) for more information.

For example, the status 'Open' may have its status category set to 'To Do'. Therefore, it will display on the Release page in the 'Issues to do' tab.

Why can't I see any warnings?

Warnings will only display on the Release page if:

- JIRA Software is connected to your Atlassian development tools, e.g. Bitbucket, Bamboo, FishEye/Crucible; or GitHub. See [Integrating with development tools](#) for more information.
- You have the 'View Development Tools' permission. See [Permissions overview](#) for more information.
- Warnings are enabled. You can see which warnings are enabled by clicking the **Manage warnings** button.

Manage Warnings

These warning settings affect all existing and future versions in this project.

Open pull requests These issues have been marked complete but have open pull requests.	<input checked="" type="checkbox"/> Enabled
Open reviews These issues have been marked complete but have open reviews	<input checked="" type="checkbox"/> Enabled
Unreviewed Code These issues have been marked complete but the commits are not part of a pull request or review.	<input checked="" type="checkbox"/> Enabled
Failing Builds These issues have been marked complete but have failing builds.	<input checked="" type="checkbox"/> Enabled

Save **Cancel**

What information will I see in the 'Development' column?

Version 3.0 UNRELEASED

Start date not set Release: 14/Sep/15 [Release Notes](#)

Version 3

63 days left

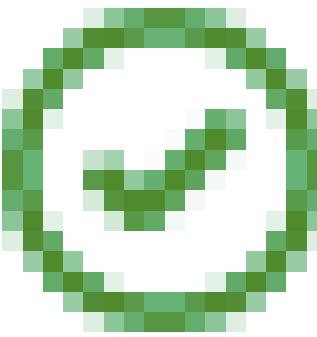
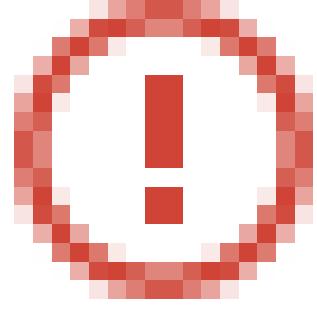
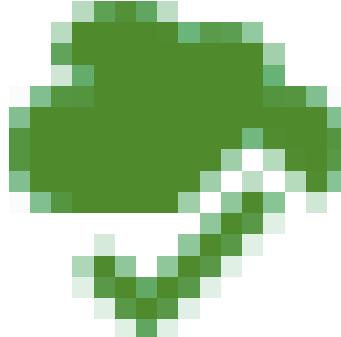
0 Warnings 16 Issues in version 3 Issues done 2 Issues in progress 11 Issues to do

1–16 of 16 [View in Issue Navigator](#)

P	T	Key	Summary	Assignee	Status	Development
↑	⌚	TIS-40	Update FlightController to handle multiple travel providers in one reservation	Max Taylor	OPEN	1 branch
↑	⌚	TIS-41	Update LodgingController to handle multiple travel providers in one reservat...	Max Taylor	CODE REVIEW	
↑	⌚	TIS-42	Extend booking experience in UI to include multiple hotels on one reservation	Kevin Campbell	OPEN	1 branch
↑	⌚	TIS-43	Extend booking experience in UI to include multiple flights on one reservation	Kevin Campbell	OPEN	1 branch
↑	⌚	TIS-44	Reward Customers an extra 5-10% when they book a large trip	William Smith	OPEN	1 branch
↑	⌚	TIS-45	Email non registered users to sign up with Teams In Space	Jennifer Evans	CLOSED	MERGED

The 'Development' column will display the commits, builds, and deployments related to an issue, provided that JIRA Software is connected to the required development tool.

Information	Required development tool	Examples
Commits	Bitbucket, GitHub, FishEye	3 commits
Branches	Bitbucket, GitHub	1 branch
Pull requests	Bitbucket, GitHub	2 pull requests OPEN

Builds	Bamboo	 or 
Deployments	Bamboo	

Click any of the symbols to view a detailed dialog. The dialog will also provide links to the commits, builds, or deployments in the relevant development tool.

How do I release a version?

Releasing the version will mark it as 'Released' in JIRA Software. You will have the option of setting a release date and moving any incomplete issues to another version. If JIRA Software is connected to Bamboo, you can also choose to run a build when the version is released. See [Deploying a release](#) to know more about releasing a version.

Why does a 'Syncing' message display next to my version's status?

JIRA Software regularly synchronizes issues with their related development data. However, to ensure that you are viewing the most recent data when you visit the Release page for the version, JIRA Software will perform

another check and synchronize any data that is not up to date.

Monitoring work in a Kanban project

The *Kanban board* is a board that was created using the "Kanban" preset (see [Creating a board](#)).

Kanban is based on the continuous delivery of work. Rather than plan iterations, the flow of work is constantly monitored to ensure that there are always tasks being worked on. This means that when tasks are completed, new tasks are pulled into work-in-progress.

Use the Kanban board if your team focuses on managing and constraining work-in-progress.

- Before you begin
- About the Kanban board
- Accessing the Kanban board
- What can I do in a Kanban board?
- Next steps

Before you begin

You need to [configure the columns](#) for your board. When configuring the columns, make sure to:

- Assign to each state (column) a distinct function in your development process, e.g. in progress, testing, awaiting merge, etc.
- Configure appropriate constraints for the columns.

Also, different functionalities in the Kanban board require different permissions. For example, to start a sprint, you need to be a project administrator for all projects that match the board's filter. See [Permissions overview](#) for more information.

About the Kanban board

An issue will only be visible in the Kanban board if:

- the issue is not a sub-task,
- the issue matches the board's saved filter,
- the issue's status maps to one of the board's columns (but not the 'Done' column), and
- there is at least a status being mapped to the right-most column, e.g. *if you have the columns Selected for Development, In Progress, and Done, ensure that you have a status mapped to In Progress at least. If you map all the statuses to the first column (Selected for Development), you will not be able to see any issues in the Kanban board.*

Accessing the Kanban board

Navigate to your desired board and click **Kanban board**.

Screenshot: the Kanban board (with an issue selected)

The screenshot shows a JIRA Kanban board with three columns: Backlog, In Progress, and Done. The Backlog column has 63 items, the In Progress column has 10, and the Done column has 25. A 'Release...' button is in the Done column. Below the columns, there are sections for 'Expedite' (3 issues) and 'Everything Else' (95 issues). The 'Expedite' section contains TIS-12 (Create 90 day plans for all departments in the Mars Office). The 'Everything Else' section contains TIS-9 (After 100,000 requests the SeeSpaceEZ server dies), TIS-8 (Requesting available flights is now taking > 5 seconds), TIS-110 (Afterburner revision VI automation), TIS-109 (Afterburner revision VI script), TIS-45 (Email non registered users to sign up with Teams In Space), TIS-57 (Buttons need to be red), and TIS-30 (Create Saturn Summer Sizzle Logo). A details panel on the right is open for TIS-12, showing its status as OPEN, component as Office Support, and reporter as Harvey Jennings.

What can I do in a Kanban board?

Task	Instructions
Add issues to the Kanban board	<p>Click Create in the header to open the 'Create Issue' dialog and create your issue. The issue will be added to the bottom of the Kanban board.</p> <p>Tip: tick the Create another checkbox in the 'Create Issue' dialog to keep it open, if you are creating multiple issues.</p>
Prioritize the Kanban board	Drag and drop an issue to rank it. You can also right-click the issue to open a menu that allows you to send it to the top or the bottom of the column.
View an issue's details	<p>Click the desired issue in the board. The issue's details will display in a panel on the right of the board. For information on how to configure the issue detail panel, see Configuring the issue detail view.</p> <p>If you want to select an issue rather than open the issue's details, use ctrl-left click or command-left click.</p>
Estimate stories	<p>Use the 'J' and 'K' keys to move through issues in the board and show the details on the right-hand side of the screen. Use the 'E' key to edit an issue, then update estimates or story points as you go.</p> <p>By default, the Story Points field is only available to issues of type 'Story' or 'Epic', see Configuring estimation and tracking for details.</p>
Identify the workload for specialists	The avatars for users (specialists) who have work assigned to them in a sprint are shown at the top of a sprint. Click ... (next to avatars) to view the sprint workload for assignees.
Create sub-tasks	<p>Select the desired issue, press the '.' key, and type 'create sub-task' to open the 'Create Sub-task' dialog. Create the sub-task as desired.</p> <p>Sub-tasks are useful for breaking a story (issue) down into implementable chunks.</p>
Flag an issue	Right-click an issue to open a menu, and select Add flag .
Filter issues	Click the desired Quick Filter (below the board name), e.g. Only My Issues . See Configuring Quick Filters for details.

Release issues	Click Release to release completed issues. When you release issues, these issues no longer appear in the Done column, and a release version is created, which you can view in the Releases tab of the project.
----------------	--

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Ranking an issue
- Estimating an issue
- Working with epics

Releasing a version

Releasing a version is the culmination of your team's hard work. It may be the result of many iterations of work across a number of months. As the iteration manager, it will be your job to ensure that all the work is complete for the release, and to coordinate the activities needed to release the version.

As described in [Configuring versions in a Scrum project](#) and [Configuring versions in a Kanban project](#), versions are handled differently in Scrum and Kanban projects. A version in a Scrum project is pre-planned and is released when the planned work is complete. In a Kanban project, a version can be released at any time — the version will contain all issues that are complete at that time.

The documentation in this section will help you release your version in JIRA Software.

Search the topics in 'Releasing a version':

Overview

Check the release status of a version

Before you release a version, you need to be sure that everything is ready — issues are complete, code is committed, reviewed and merged, builds are passing, etc. In JIRA Software, the Release Hub can provide you with the information you need for a release, in one place.

Learn more: [Checking the release status of a version](#)

Deploy the release

Deploying your release requires actions to be taken on a number of different systems. The version in JIRA Software needs to be released, build(s) need to be run to generate the artifact, the artifact needs to be deployed to the right environments, etc.

Learn more: [Deploying a release](#)

Create the release documentation

Once you have deployed your release, you may want to create documentation to accompany it. If you have JIRA Software connected to Confluence, you can generate a 'Change Log' report in Confluence for your stakeholders.

Learn more: [JIRA Report Blueprint](#)

Already released your version?

Next: [Reporting](#)

Project lifecycle

NEW PROJECT

BACKLOG

NEW VERSION

REPORTING

WORK

RELEASE

Checking the release status of a version

A version has many moving parts. As an iteration manager, you need to be sure that everything is ready before you can release — issues are complete, code is committed, reviewed and merged, builds are passing, etc. The Release Hub in JIRA Software can help you stay on top of everything when it's time to release a version. Each issue in JIRA Software also has a Development panel that helps you see the development status of an issue at a glance.

On this page:

- Before you begin
- Using the Release Hub
- Checking the development status of an issue
- Next steps

Before you begin

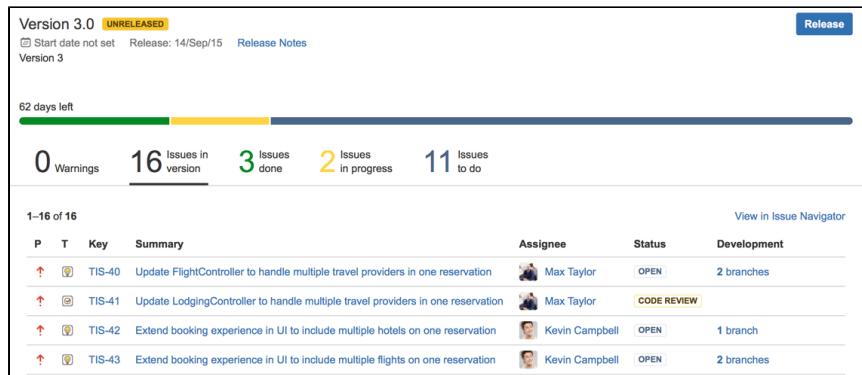
- If you are using a Kanban board, you cannot use the Release Hub to check the status of an unreleased version. The Release Hub on Kanban boards only shows released versions.

Using the Release Hub

As described in [Checking the progress of a version](#), the Release Hub is useful for seeing the status of a version at a point in time. It shows a summary and a breakdown of the issues in the version, grouped by status, so you can see if there's any remaining work for the release.

If you have JIRA Software connected to your development tools, the 'Warnings' tab can also help you spot issues that could cause problems for the release. These include issues that should be in the release but haven't been merged (open pull requests), issues that haven't had code reviews, etc. Rather than spend your time hunting this information down, you can help your team sort these problems out as soon as they occur.

1. Navigate to the project that your version is in.
2. Click **Releases** > your desired version.



Checking the development status of an issue

If you need to deep dive into individual issues, the Development panel on each issue will show you its development status. This includes commits, builds, branches, pull requests, and more. For more information, see [Viewing the development information for an issue](#).

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Configuring development tools

Deploying a release

It's time to turn your team's hard work into a software release. By this stage, you should be confident that your version is ready to be released — issues are complete, code is checked in, reviewed and merged, builds are passing, etc.

To deploy a release, you would typically release the version in JIRA Software, build the release, then deploy the release to the required environment.

Before you begin

- You must be a project administrator to do the JIRA Software tasks on this page.

On this page:

- Before you begin
- Release your version in JIRA Software
- Build the release
- Deploy the release
- Next steps

Release your version in JIRA Software

Before you release your version, check the release status of your version.

Scrum

In a Scrum project, a version is created when you start work on the project. Issues are assigned to the version throughout development. Releasing the version is just a matter of marking the version as released.

1. Navigate to the project that your version is in.
2. Click **Releases** > your desired version.
3. On the version page, click the **Release** button.
4. The release dialog will be displayed.
5. Enter the details for the release, then click the **Release** button.

Note, if there are unresolved issues, you can choose to ignore these issues and proceed with the release, or move them to a later version.

Releasing a version on a Scrum board marks the version as released. It doesn't automatically modify the issues in the version in any way.

Kanban

In a Kanban project, you create the version at the time of the release. The version will contain all issues that are 'Done' (i.e. in the right-most column) at that time.

1. Navigate to the project that your version is in.
2. Click **Kanban board**.
3. Click **Release...**
4. The release dialog will be displayed.
5. Enter the details for the release, then click the **Release** button.

Releasing a version on a Kanban board does the following:

- For all issues that are 'Done', sets the Fix Version to the version that you just created.
- Marks the version as released.

As a result, the issues should disappear from the board, as the default Work Sub-Filter for a Kanban board only shows issues that have no Fix Version, or issues with a Fix Version that is unreleased. Note, if an issue has multiple Fix Versions, it will only disappear from the board once all versions are released.

Build the release

The next step is to build the release. A release is essentially a snapshot of the artifact(s) and related metadata that will be deployed. A release is created from a single build.

If you have connected JIRA Software to Bamboo, you can automatically run a Bamboo build when you release a version. Otherwise, if you haven't connected JIRA Software to Bamboo, or you are using a different

build tool, you will need to run your builds independently.

Learn more: [Running a Bamboo build when releasing a version](#)

Deploy the release

Once you have built your artifacts, you need to deploy them to the required environment. You may do a few deployments to different environments (e.g. test, staging, etc) before releasing it to your customers.

If you are using Bamboo, deployment projects make it simple for you to deploy your artifacts to different environments. Otherwise, you will need to manually copy the files across environments.

Learn more: [Deployment projects](#)

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Configuring development tools](#)

Running a Bamboo build when releasing a version

Releasing a new version of software usually involves a number of tasks, such as releasing the version in JIRA Software, building and testing, merging code, creating tags, creating branches, labeling builds, etc. If you have connected JIRA Software to [Atlassian's Bamboo](#), you can trigger these tasks to run automatically at the release of a version in JIRA Software.

When you release a JIRA Software version, you will have the option of selecting a Bamboo plan and specifying which stages in the plan to run. Releasing the version will run the plan in Bamboo. If the Plan is successful, the version will be released on JIRA Software. Otherwise, the version will not be released.

On this page:

- [Before you begin](#)
- [Running a Bamboo build when releasing a version](#)
- [Next steps](#)

Before you begin

- Your JIRA administrator must have integrated JIRA Software with Bamboo. For instructions, see [Configuring development tools](#).
- Your JIRA administrator must have installed the latest [JIRA Bamboo plugin](#) to use the release management feature. For instructions on how to install a plugin, see [Managing add-ons](#).
- You must be a project administrator for the project that the version is in. See [Permissions overview](#) for more information.

Running a Bamboo build when releasing a version

1. Navigate to the project that your version is in.
2. Click **Releases**.
3. Click your desired version.
4. On the version page, click the **Release** button.
5. The release build dialog will be displayed.
6. Enter the build details for the release:

No Build	Choose this option if you <i>do not</i> want to run a Bamboo build, i.e. you only want to release the version in JIRA Software.
----------	---

New Build	Choose this option, if you want to run a Bamboo build that has not been started: * Using Plan — You can select any plan in the linked Bamboo instance that you have permission to view (unless your administrator has configured basic HTTP authentication). * Stages — You can select the stages that you want to run for this release. Note, you cannot skip stages. * Build Variables — You can override any global variables or plan variables with your own parameters. See Running a plan build manually in Bamboo .
Existing Build	Choose this option, if you want to run a Bamboo build that is in progress and has been paused at an optional stage: * Using Plan — You can select any plan in the linked Bamboo instance that you have permission to view (unless your administrator has configured basic HTTP authentication). * Stages — You can select the stages that you want to run for this release. Note, you cannot re-run stages that have been completed nor skip stages. * Build Variables — You can override any global variables or plan variables with your own parameters. See Running a plan build manually in Bamboo .

7. Click **Release**. The Bamboo build will run. If it is successful, the version will be released. If not, you can choose to run it again, or select a different plan.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Configuring development tools](#)

Reporting

Reporting is an activity that you will be doing throughout a project. JIRA Software has a range of reports that you can use to show information about your project, versions, epics, sprints, and issues.

The documentation in this section will help you configure and use the reports in JIRA Software.

Search the topics in 'Reporting':

Generating a report

1. Navigate to the desired board and click **Reports**. The last report viewed will be displayed.
2. Click **Switch report** to view a different report. The reports in this list are specific to agile development. See the 'Reports for Scrum projects' or 'Reports for Kanban projects' sections below for more details.
3. If you want to view reports that are not specific to agile development, select **All reports** from the **Switch report** drop-down, and view the reports that are not in the 'Agile' section. See the 'General reports for analyzing issues' section below for more details.

Reports for Scrum projects

Chart	Applies to	Purpose
Burndown Chart	Sprints	Tracks the total work remaining, and projects the likelihood of achieving the sprint goal. This helps your team manage its progress and respond accordingly.

Sprint Report	Sprints	Shows the work completed or pushed back to the backlog in each sprint. This helps you determine if your team is overcommitting or if there is scope creep.
Control Chart	Projects, versions, or sprints	Shows the cycle time for your product, version, or sprint. This helps you identify whether data from the current process can be used to determine future performance.
Cumulative Flow Diagram	Any period of time	Shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.
Epic Report	Epics	Shows the progress towards completing an epic over time. This helps you manage your team's progress by tracking the remaining incomplete and unestimated work.
Epic Burndown	Epics	Similar to the Epic Report, but optimized for Scrum teams that work in sprints. Tracks the projected number of sprints required to complete the epic. This helps you monitor whether the epic will release on time, so you can take action if work is falling behind.
Release Burndown	Versions	Similar to the Version Report, but optimized for Scrum teams that work in sprints. Tracks the projected release date for a version. This helps you monitor whether the version will release on time, so you can take action if work is falling behind.
Velocity Chart	Sprints	Tracks the amount of work completed from sprint to sprint. This helps you determine your team's velocity, and estimate the work your team can realistically achieve in future sprints.
Version Report	Versions	Tracks the projected release date for a version. This helps you monitor whether the version will release on time, so you can take action if work is falling behind.

Reports for Kanban projects

Chart	Applies to	Purpose
Control Chart	Projects, versions, or sprints	Shows the cycle time for your product, version, or sprint. This helps you identify whether data from the current process can be used to determine future performance.
Cumulative Flow Diagram	Any period of time	Shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.

General reports for analyzing issues

Chart	Purpose

Average Age Report	Shows the average age of unresolved issues for a project or filter. This helps you see whether your backlog is being kept up to date.
Created vs Resolved Issues Report	Maps created issues versus resolved issues over a period of time. This helps you understand whether your overall backlog is growing or shrinking. ▼ Notes... <ul style="list-style-type: none"> Viewing the chart — Areas in red show periods where more issues were created than resolved. Areas in green show periods where more were resolved than created.
Pie Chart Report	Shows a pie chart of issues for a project or filter grouped by a specified field. This helps you see the breakdown of a set of issues, at a glance. For example, you could create a chart to show issues grouped by Assignee for a particular version in a project (using a filter).
Recently Created Issues Report	Shows the number of issues created over a period of time for a project or filter, and how many were resolved. This helps you understand if your team is keeping up with incoming work. ▼ Notes... <ul style="list-style-type: none"> Viewing the chart — The green portion of the bar shows the created issues that are resolved. The red portion shows created but unresolved issues as yet.
Resolution Time Report	Shows the length of time taken to resolve a set of issues for a project or filter. This helps you identify trends and incidents that you can investigate further.
Single Level Group By Report	Shows issues grouped by a particular field for a filter. This helps you group search results by a field, and see the overall status of each group. For example, you could view the issues in a version of a project, grouped by Assignee.
Time Since Issues Report	For a date field and project or filter, maps the issues against the date that the field was set. This can help you track how many issues were created, updated, etc over a period of time.

Time Tracking Report *	<p>Shows time tracking information on issues for a particular version of a project.</p> <p>▼ Notes...</p> <p>The table in the report shows the issues within the version:</p> <ul style="list-style-type: none"> There are four time tracking fields as follows: <ul style="list-style-type: none"> Original Estimate - The original estimate of the total amount of time it would take to complete this issue. Estimated Time Remaining - The current estimate of the remaining amount of time it would take to complete this issue. Time Spent - The amount of time spent on the issue. This is the aggregate amount of time that has been logged against this issue. Accuracy - The accuracy of the original estimate compared to the current estimate for the issue. It is the difference between the sum of the Time Spent and Estimated Time Remaining fields, and the Original Estimate field. If sub-tasks are enabled, the ***column at the right of the field shows the aggregate time tracking information for each 'parent' issue (i.e. the sum of the issue's own values, plus those of its sub-tasks). The last line of the table shows the aggregate time tracking information for the whole version. <p>The report also includes two bar-graphs (above the table), which represent the aggregate time tracking information for the version:</p> <ul style="list-style-type: none"> The first bar-graph ('Progress') shows the percentage of completed issues (green) and incomplete issues (orange) in this version: <div style="border: 1px solid black; padding: 2px;"> Progress: 40% </div> <ul style="list-style-type: none"> The second bar-graph ('Accuracy' -blue) shows the accuracy of the original estimates. <p>The length of the Accuracy bar compared to the Progress bar indicates whether the issues in this version are ahead of or behind schedule. There are three cases:</p> <ol style="list-style-type: none"> <i>The issues are on schedule with the original estimate.</i> The Accuracy bar is completely blue and is the same length as the Progress bar above it. <div style="border: 1px solid black; padding: 2px;"> Progress: 40% Accuracy: 0% </div> <ol style="list-style-type: none"> <i>The issues are behind the original estimate (i.e. will take longer than originally estimated).</i> The Progress graph is longer than the Accuracy graph. The blue region represents the original estimated time, and the light-grey region is the amount of time by which issues are behind. <div style="border: 1px solid black; padding: 2px;"> Progress: 42% Accuracy: -4% </div> <ol style="list-style-type: none"> <i>The issues are ahead of the original estimate (i.e. will take less time than originally estimated).</i> The Accuracy graph is longer than the Progress graph. The blue bar represents the original estimated time, and the light-grey region represents the amount of time by which the original estimates were overestimated. <div style="border: 1px solid black; padding: 2px;"> Progress: 47% Accuracy: 8% </div>
User Workload Report *	<p>Shows how much work a user has been allocated, and how long it should take.</p> <p>For a specified user, you'll be able to see the number of unresolved issues assigned to the specified user, and the remaining workload, on a per-project basis.</p>

Version Workload Report *	Shows how much outstanding work there is (per user and per issue) before a given version is complete. For the specified version, you'll be able to see a list of unresolved issues assigned to each user, each user's workload, and a summary of the total remaining workload for the version.
Workload Pie Chart Report *	Shows the relative workload for assignees of all issues for a project or filter.

* Only available if your JIRA administrator has enabled time tracking.

Reports available in Confluence

If you have connected JIRA to Confluence, you can create the following reports in Confluence.

Chart	Purpose
Change Log	Displays a list of issues from JIRA. This list can be static or dynamic, automatically updating as the status of your issues change in JIRA.
Status Report	The Status Report displays the progress of a JIRA project and fix version in pie charts by status, priority, component, and issue type. The Status Report uses the JIRA Chart macro, and is dynamic.

Other reports

- Additional reports (e.g. Gantt Chart Report, Timesheet Report, JIRA SQL Plugin) are available for download from the [Atlassian Marketplace](#).
- JIRA administrators can also create new reports with the plugin API — see our [Tutorial - Creating a JIRA report](#). If you don't want to build a plugin yourself, [Atlassian Experts](#) are available for custom projects.
- Issue filters can be exported to Microsoft Excel, where they can be further manipulated into charts and reports. See [Working with search results](#).

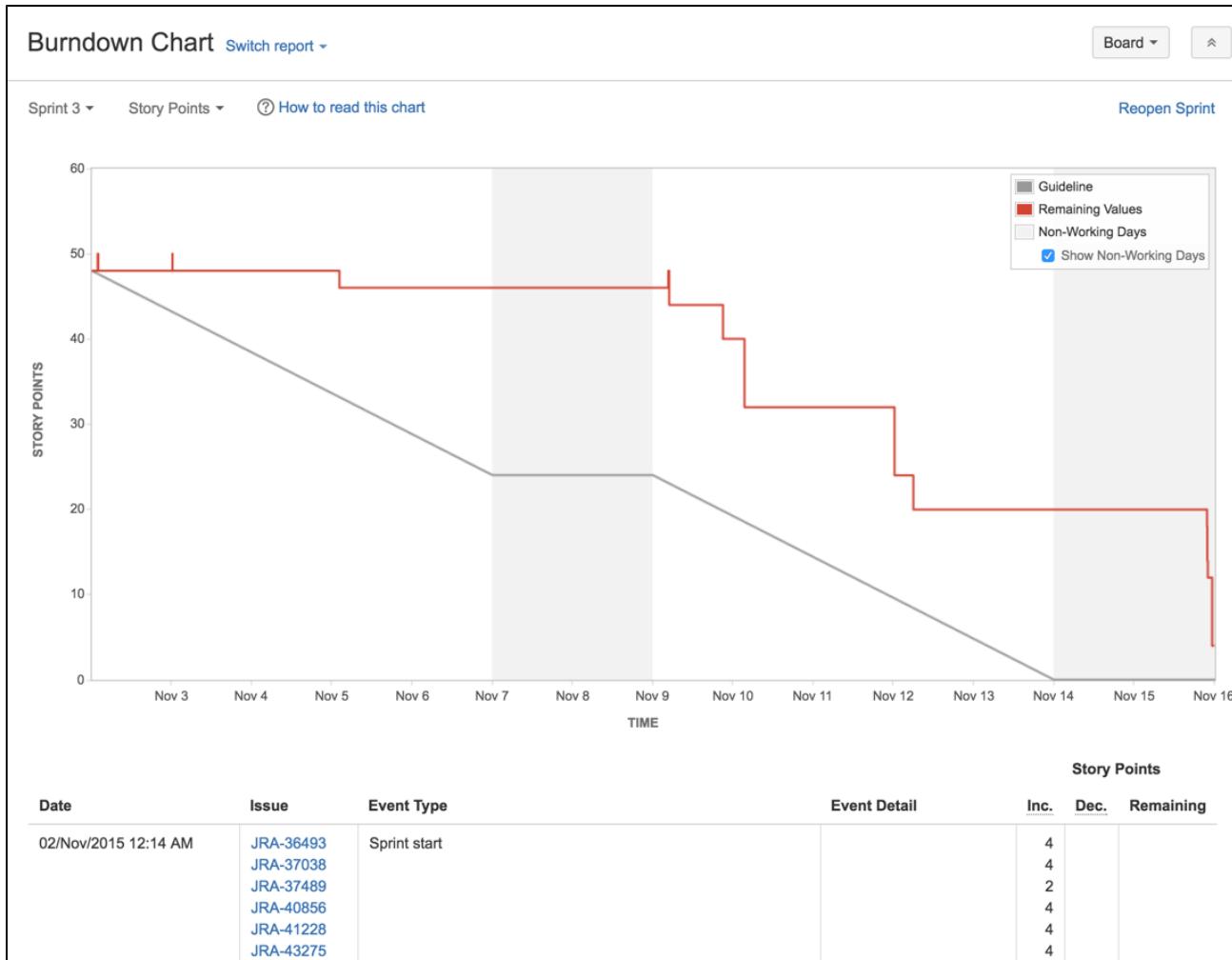
What next?

You are done! Now the cycle begins again. You will probably be planning a new version, even as you wrap up an existing version. You might even be ready to start a new project. Check out the previous topics below:

- [Starting a new project](#)
- [Building a backlog](#)
- [Planning a version](#)
- [Releasing a version](#)



Burndown Chart



Screenshot: Burndown Chart

A Burndown Chart shows the actual and estimated amount of work to be done in a sprint. The horizontal x-axis in a Burndown Chart indicates time, and the vertical y-axis indicates cards (issues).

Use a Burndown Chart to track the total work remaining, and to project the likelihood of achieving the sprint goal. By tracking the remaining work throughout the iteration, a team can manage its progress, and respond to trends accordingly. For example, if the Burndown Chart shows that the team may not likely reach the sprint goal, then the team can take the necessary actions to stay on track.

Before you begin

- The Burndown Chart only applies to Scrum boards.

Viewing the Burndown Chart

- Click **Boards** (in header) > select your desired board.
- Click **Reports**, then select **Burndown Chart**.
 - To choose a different sprint, click the sprint drop-down.
 - To choose a different estimate statistic, click the estimation statistic drop-down. This change will be saved for you, for when you next visit this chart.

Tip: Click **How to read this chart** at the top of the report to view a short description of the report.

Understanding the Burndown Chart

Before you start using the Burndown Chart, you should get to know how it works. The following information

On this page:

- Before you begin
- Viewing the Burndown Chart
- Understanding the Burndown Chart
- Next steps

will help you understand the key functionalities of the Burndown Chart:

- The Burndown Chart is board-specific – that is, it will only include issues that match your board's saved filter.
- The vertical axis represents the [estimation statistic](#) that you have configured for your board.
- The Burndown Chart is based on your board's column mapping. An issue is considered to be 'To Do' when it is in a status that has been mapped to the left-most column of your board. Similarly, an issue is considered to be 'Done' when it is in a status that has been mapped to the right-most column of your board. See [Configuring columns](#) for more information.
- If the grey 'Guideline' line does not show, the sprint may have been started before any issues were assigned to it, as described in this [KB article](#).

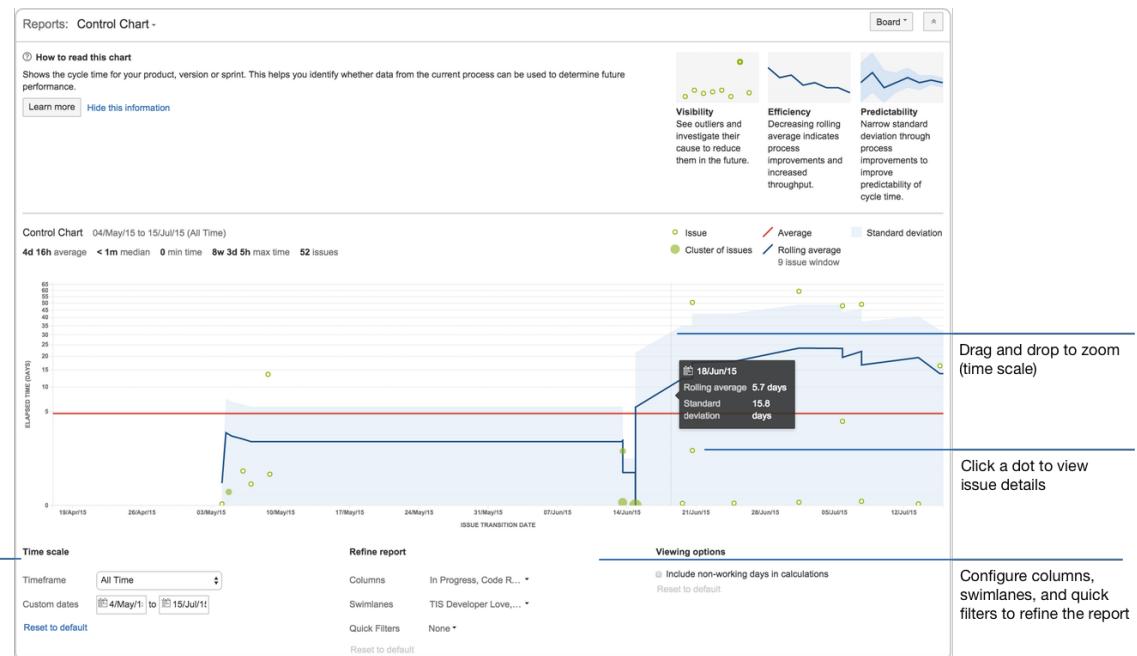
Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Gadgets for JIRA applications](#)
- [Configuring working days](#)
- [Configuring estimation and tracking](#)

Control Chart



Screenshot: Control Chart

About the Control Chart

The Control Chart shows the Cycle Time (or Lead Time) for your product, version, or sprint. It takes the time spent by each issue in a particular status (or statuses), and maps it over a specified period of time. The average, rolling average, and standard deviation for this data are shown.

A Control Chart helps you identify whether data from the current sprint can be used to determine future performance. The less variance in the cycle time of an issue, the higher the confidence in using the mean (or median) as an indication of future performance.

Here are some of the ways that you could use a Control Chart:

- Analyze your team's past performance in a retrospective,
- Measure the effect of a process change on your team's productivity,
- Provide external stakeholders with visibility of your team's

- performance, and
- For Kanban, use past performance to set targets for your team.

On this page:

- About the Control Chart
- Viewing the Control Chart
- Printing the Control Chart
- Understanding the Control Chart
- Tips and examples
- Known issues
- Next steps

Viewing the Control Chart

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Control Chart**.
3. Configure the chart as desired. The screenshot at the top of this page highlights the controls that you can use to configure the Control Chart.

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

 *If you are using Internet Explorer 8, the Control Chart will not work.*

Printing the Control Chart

To print the report, view the report and use the print functionality for your browser. The report will fit on either A4- or Letter-sized pages in both portrait and landscape modes (note, there is a known issue printing in landscape using Chrome).

Understanding the Control Chart

Before you start using the Control Chart, you should get to know how it works. The following questions and answers cover the key functionalities of the Control Chart:

What is cycle time and lead time?

Cycle time is the time spent working on an issue — typically, the time taken from when work begins on an issue to when work is completed, but it also includes any other time spent working on the issue. For example, if an issue is reopened, worked on, and completed again, then the time for this extra work is added to the cycle time.

Lead time is similar to cycle time, but is the time taken from when an issue is logged (not when work begins) until work is completed on that issue.

How is cycle time determined?

The statuses used to calculate cycle time depend on the workflow you're using for your project. You should configure the Control Chart to include the statuses that represent the time spent working on an issue. Note, the Control Chart will attempt to select these statuses automatically.

For example, if you are using the 'JIRA Software Development' workflow, you may consider work to have started on an issue when it transitions to 'In Progress', and work to have completed when it transitions from 'In Review' to 'Done'. You would show this on the Control Chart by selecting 'In Progress' and 'In Review' as the **Columns**, as this would show the time that issues have spent in those two statuses.

Tip: you can also configure the Control Chart to show lead time data instead of cycle time data. Just select the statuses that represent the time spent on an issue; from the time it is raised until work was completed.

How is rolling average calculated?

The rolling average (blue line on the chart) is issue-based, not time-based. For every issue shown on the chart, the rolling average (at that point in time) is calculated by taking the issue itself, X issues before the issue and X issues after the issue, then averaging their cycle times. 20% of the total issues displayed (always an odd number and a minimum of 5 issues) is used in the calculation.

For example, in the screenshot below, at the point of time where an issue (green dot) is shown, the rolling average is calculated as follows:

1. Take the issue plus four issues before and four issues after (nine issues total).
2. Average the cycle times for the nine issues.
3. Map the blue line to the calculated average.

If the **Timeframe** is reduced to 'Past two weeks', the number of issues used would reduce, as there are fewer total issues available to use for the calculations.



This method produces a steady rolling average line that shows outliers better (i.e. rolling average doesn't deviate as sharply towards outliers). The rolling average line is also easy to understand, as the inflections are related to the positions of issues.

i If you would like to know more about why the rolling average calculation is based on a percentage of the total issues, rather than a time period, see [Comparing different methods of calculating the rolling average on the Control Chart](#).

What does the blue shaded area represent?

The blue shaded area of the control chart represents the standard deviation — that is, the amount of variation of the actual data from the rolling average.

The standard deviation gives you an indication of the level of confidence that you can have in the data. For example, if there is a narrow blue band (low standard deviation), you can be confident that the cycle time of future issues will be close to the rolling average.

What do the dots on the chart represent?

As shown on the chart legend, each dot represents an issue or a group (cluster) of issues:

- The vertical placement of the dot represents the cycle time for the issue, i.e. the 'Elapsed Time'. For a cluster of issues, the dot is placed at the average cycle time for the issues.
- The horizontal placement indicates when the issue(s) transitioned out of the last status selected on the chart (in **Columns**). For example, if you are using the 'JIRA Software Development' workflow and have selected 'In Progress' and 'In Review' as the columns on the Control Chart, the dots will indicate when the issue transitioned out of the 'In Review' status.

Why does the scale of the Elapsed Time axis change when I change Timeframe?

If the maximum **Elapsed Time** value on the chart is less than 30 days, then a linear scale is used for the y-axis. If it is 30 days or greater, than a cube-root power scale is used.

When you change the **Timeframe**, you may include issues with an elapsed time of greater than 30 days when you previously did not, or vice versa. This will change the scale, as described above.

Linear scale for Elapsed Time



Cube-root power scale for Elapsed Time



Tips and examples

Learn how to tweak your Control Chart to show the data you need with the following examples:

▼ [Show me some tips...](#)

Tip 1: Remove unwanted outliers

The Control Chart can help you identify outliers. On closer examination, you may determine that certain outliers are invalid due to human error. For example, you may have a story that was started but stopped, then eventually dropped back to the backlog, but not returned to the 'To Do' status. The time that the issue spent 'In Progress' would incorrectly skew the data for your Control Chart.

To remove unwanted outliers from your Control Chart, add a label to each outlier issue (e.g. outlier) and create a Quick Filter with this JQL: `labels is EMPTY or labels not in (outlier)`. Configure your Control Chart to use this Quick Filter.

Example Control Chart with invalid outliers



*Example Control Chart with invalid outliers removed
(note the smaller scale for 'Elapsed Time')*



Tip 2: Remove triage casualties

In a Control Chart, you generally want to track the issues that are resolved as 'Fixed'. Issues that are triaged and resolved as a duplicate, answered, tracked elsewhere, etc can skew the data, bringing the average cycle time down considerably.

To remove triage casualties from your Control Chart, create a Quick Filter with this JQL: `resolution in (Fixed)`. Configure your Control Chart to use this Quick Filter.

Example Control Chart including issues where the resolution is not 'Fixed'



Example Control Chart excluding issues where the resolution is not 'Fixed' (note the higher average cycle time)



Tip 3: Exclude current work

The Control Chart shows data for issues that have been in a selected column, but are no longer in a selected column. This gives the cycle time (total elapsed time) for the issues. However, by default, this will include issues that are still moving across the board.

To view the data for completed work only in your Control Chart, create a Quick Filter with this JQL: `status in (Resolved, Closed)`. Configure your Control Chart to use this Quick Filter.

Example Control Chart including all issues



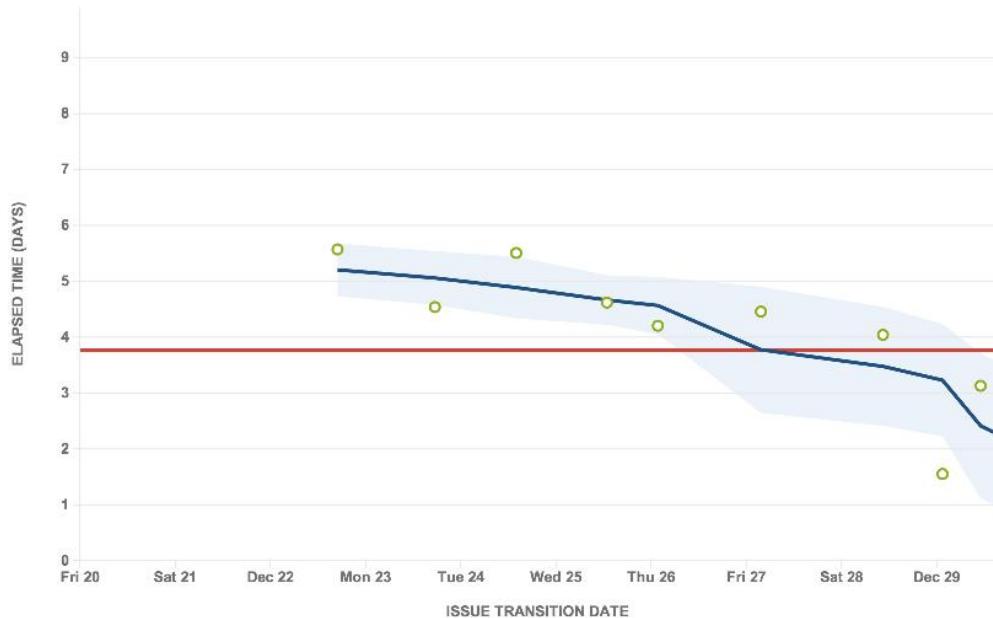
Example Control Chart including issues where the status is 'Resolved' or 'Closed' only



Learn how to interpret a Control Chart with the following examples:

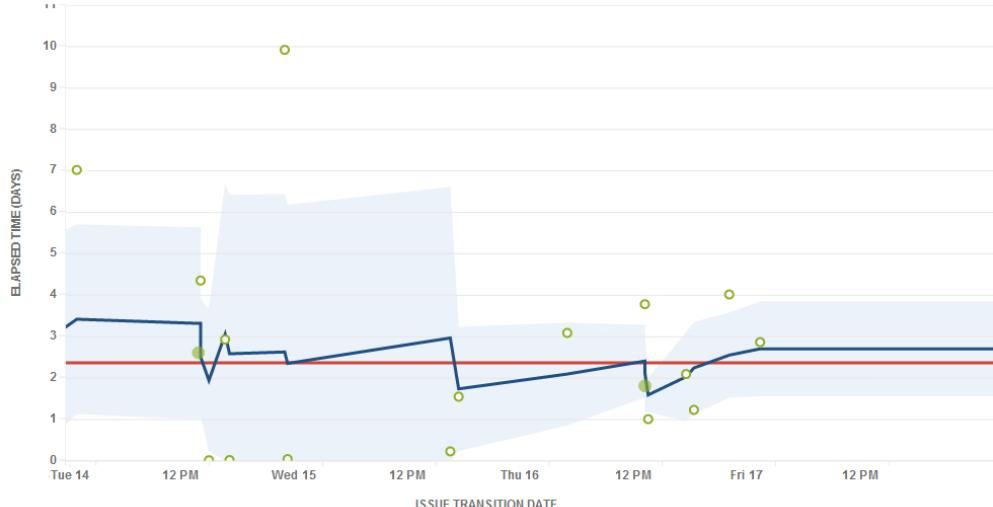
▼ Show me some examples...

Example 1:



- The productivity of the team is increasing: indicated by the downward trend of the rolling average.
- The cycle time of future issues are likely to be close to the rolling average (2 days or less): indicated by the low standard deviation (narrow blue shaded area).

Example 2:



- The team's productivity is pretty consistent: indicated by rolling average being close to the average.

- There are obvious outliers on Tue 14 and Wed 15 (7 days and 10 days elapsed time respectively, compared 2 days on average) that should be investigated.
- Data is becoming more predictable over time.

Known issues

If you encounter an issue that is not on this list, please [raise it in our issue tracker](#).

[▼ Click to view known issues...](#)

Key Summary T P Status

No issues found

Next steps

- i Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Configuring working days
- Configuring Quick Filters

Comparing different methods of calculating the rolling average on the Control Chart

The new Control Chart uses a different calculation for the rolling average, than the old Control Chart. On this page, we compare different methods for calculating the rolling average, and show how the issue-based calculation (Example 3) is better.

Consider how the rolling average (blue line) is mapped in the examples shown below. All three examples use the same data:

- Example 1: Window based on time, centered on each day
- Example 2: Window based on time, centered on each issue
- Example 3: Window based on a number of the total issues, centered on each issue

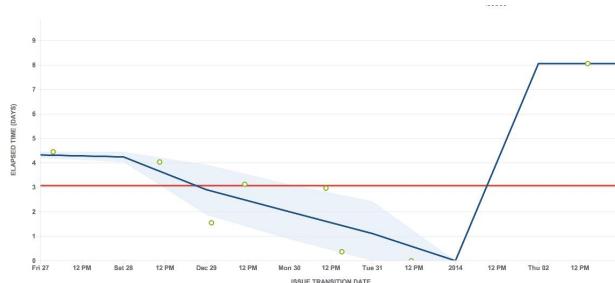
Example 1: Window based on time, centered on each day

In this example, the rolling average is calculated and mapped for **each day** on the chart. This calculation is used in the old Control Chart.

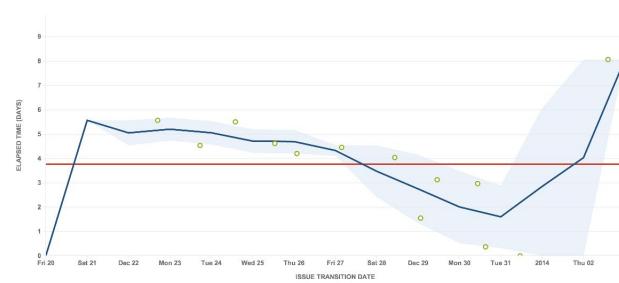
On each day, the average is calculated by doing the following:

- Determine a **window of time** (e.g. two days), based on the overall timeframe of the chart.
- Use the issues in the window to calculate the average cycle time.

One-week timeframe (one-day window)



Two-week timeframe (two-day window)



This approach has a number of problems. In these charts, you can see that the last issue is clearly an outlier. However, as it is too far apart from other issues, the rolling average line simply jumps up, particularly in the one-week timeframe. The rolling average line also bends at unexpected places on the chart.

The one-week timeframe also shows no standard deviation (light blue shaded area) around the outlier, although it is a bit better when zoomed out to a two-week timeframe. However, the two-week timeframe

shows other problems — the rolling average rises and even fluctuates before the very first issue.

Example 2: Window based on time, centered on each issue

In this example, the rolling average is calculated and mapped for **each issue** on the chart.

At each issue, the average is calculated by doing the following:

- Determine a **window of time** (e.g. two days), based on the overall timeframe of the chart.
- Use the issues in the window to calculate the average cycle time.

One-week timeframe (one-day window)



Two-week timeframe (two-day window)



This chart is the same as the one on the left, but is zoomed out to two weeks, i.e. includes an additional week prior to the original.

This approach is better than the approach in Example 1. By centering around the issues points, the variation in the moving average line is easier to understand — each completed issue causes a change. The moving average also starts at the first point, instead of moving up from zero.

However, note that the outlier is still not handled well. The rolling average still jumps up and shows no standard deviation.

Example 3: Window based on a number of the total issues, centered on each issue

In this example, the rolling average is calculated and mapped for **each issue** on the chart. This calculation is used in the new Control Chart and is described in more detail in [Control Chart](#).

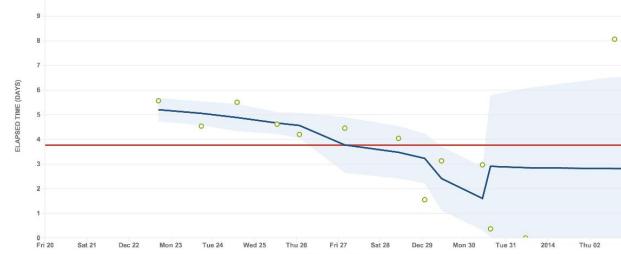
At each issue, the average is calculated by doing the following:

- Take a **set number of issues** (e.g. five issues), based on the total number of issues in the chart.
- Use the issues in the window to calculate the average cycle time.

One-week timeframe



Two-week timeframe



This approach improves upon both Example 1 and Example 2. The window is not time-based, but is based on the number of issues — in the charts above, it includes two issues ahead and two issues behind.

The rolling average starts at the first data point and the variations are easy to understand. In addition, it handles the outlier better. The rolling average is a more steady line towards the right of the chart, and clearly shows the last issue as an outlier with the appropriate standard deviation.

Cumulative Flow Diagram

Reports: Cumulative Flow Diagram ▾

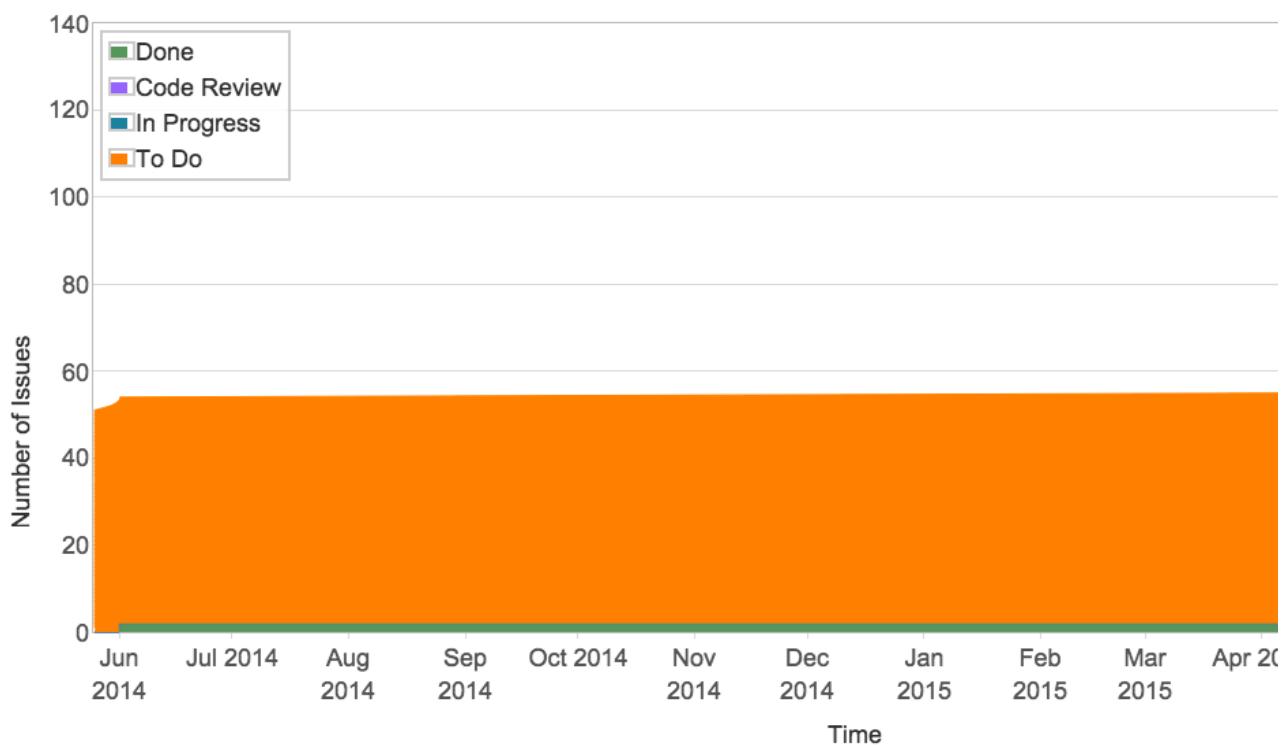
ⓘ How to read this chart

Shows the statuses of issues over time. This helps you identify potential bottlenecks that need to be investigated.

[Hide this information](#)

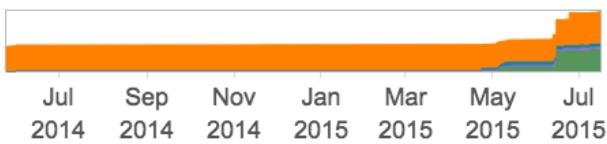
Cumulative Flow Diagram

25/May/14 to 16/Jul/15 (All Time) ▾ [Refine report](#) ▾



Overview

Click and drag cursor across chart or chart overview to select date range (double-click overview to reset).



Screenshot: Cumulative Flow Diagram

A Cumulative Flow Diagram (CFD) is an area chart that shows the various statuses of work items for an application, version, or sprint. The horizontal x-axis in a CFD indicates time, and the vertical y-axis indicates cards (issues). Each colored area of the chart equates to a workflow status (i.e. a column on your board).

The CFD can be useful for identifying bottlenecks. If your chart contains an area that is widening vertically over time, the column that equates to the widening area will generally be a bottleneck.

On this page:

- Viewing the Cumulative Flow Diagram
- Understanding the Cumulative Flow Diagram
- Next steps

Viewing the Cumulative Flow Diagram

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Cumulative Flow Diagram**.
 - To refine the data shown in the report, click **Refine report**, and select the desired filters.
 - To select a different timeframe, click the date range drop-down at the top of the chart.
 - To select a different date range, drag your cursor across the 'Overview' at the bottom of the chart.

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

Understanding the Cumulative Flow Diagram

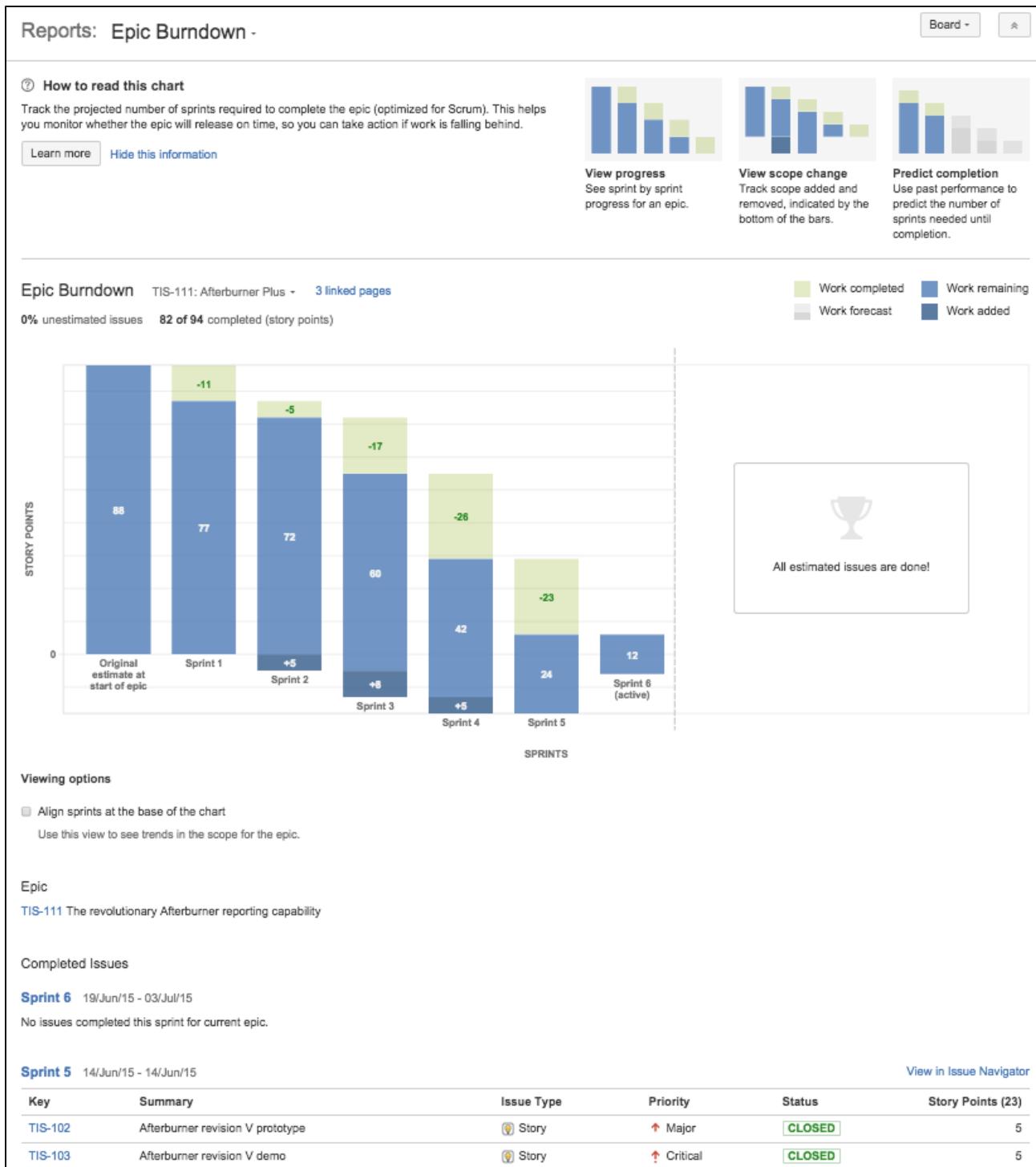
Before you start using the CFD, you should get to know how it works. The following information will help you understand the key functionalities of the CFD:

- The CFD is board-specific – that is, it will only include issues that match your board's saved filter.
- The CFD is based on your board's column mapping. An issue is considered to be 'To Do' when it is in a status that has been mapped to the left-most column of your board. Similarly, an issue is considered to be 'Done' when it is in a status that has been mapped to the right-most column of your board. See [Configuring columns](#) for more information.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Epic Burndown



Screenshot: Epic Burndown report (story points)

About the Epic Burndown report

The Epic Burndown report shows you how your team is progressing against the work for a epic. An epic is a large user story that can be broken down into a number of smaller stories. The report will show data based on the estimation statistic that your board is using.

Here are some of the ways that you could use an Epic Burndown report:

- See how quickly your team is working through the epic,
- See how work added and removed during the sprint has affected your team's overall progress, and
- Predict how many sprints it will take to complete the work for an epic, based on past sprints and changes during the sprints.

If you have used the Epic Report before, you will notice some similarities. However, the Epic Burndown is optimized for scrum teams that work in sprints — which makes tracking much easier.

On this page:

- About the Epic Burndown report
- Viewing the Epic Burndown report
- Printing the Epic Burndown report
- Understanding the Epic Burndown report
- Known issues
- Next steps

Viewing the Epic Burndown report

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Epic Burndown**.
3. Select the relevant epic from the Epic Burndown drop-down. You will be able to select from epics that are in projects configured for your board (via the board's filter).

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

! *If you are using Internet Explorer 8, the Epic Burndown will not work.*

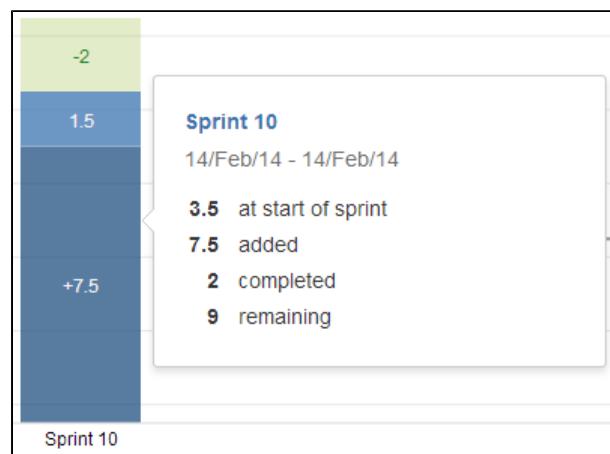
Printing the Epic Burndown report

To print the report, view the report and use the print functionality for your browser. The report will fit on either A4- or Letter-sized pages in both portrait and landscape modes (note, there is a known issue printing in landscape using Chrome).

Understanding the Epic Burndown report

Before you start using the Epic Burndown report, you should get to know how it works.

The sprint bar



- **Light green section** = work completed during the sprint.
Note, if a bar is completely light green, you won't be able to tell how much of the work completed was originally estimated or not. To find out this information, click the bar to view the details.
- **Light blue section** = work that is remaining in the epic, out of the total work estimated for the epic at the start of the sprint.
- **Dark blue section** = work that was added during the sprint, but not originally included (i.e. scope change).
- **Light green section + light blue section** = total work in the epic that was originally

estimated at the start of the sprint.

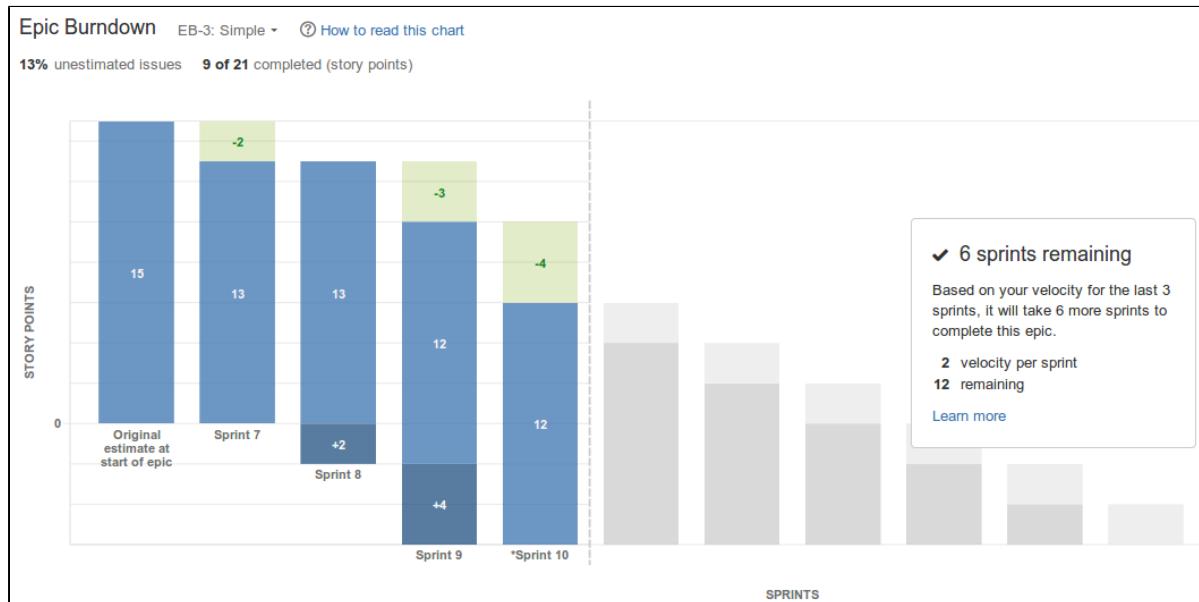
- **Light blue section + dark blue section =** total work in the epic that is remaining at the end of the sprint.
- **Bars with grey sections =** predicted sprints (see [below](#)).

Predicted sprints

Predicted sprints are calculated based on your team's velocity* (amount of work completed in the last three sprints), and the total work remaining for the epic. Scope change is not considered when calculating the velocity*, but is included in the total work remaining.

* *not the same as the velocity described in the [Velocity Chart](#)*

Consider the following example:



1. **Assessing the outstanding work:** 12 story points are remaining for the epic, at the start of the current sprint (sprint 10).
2. **Calculating the velocity:** 7 story points were completed in the last three sprints (sprint 8, sprint 9, and sprint 10). This averages out to a velocity of 2 story points per sprint, rounding to nearest story point.
3. **Predicting the remaining sprints:** At a velocity of 2 story points per sprint, it will take 6 more sprints to complete the work for the epic. That is, 6 sprints of 2 story points each.

Is my current sprint counted when calculating my team's velocity?

The current sprint is usually not counted when calculating the team's velocity. In the example above, the current sprint bar shows grey sections (like the bars for the predicted sprints) to represent this. The exception is when you have already *completed more work in the current sprint than the work that was predicted to be completed*. In this case, the current sprint (and the actual work completed) is used as one of the three sprints used to calculate the velocity. Also, the sprint bar will show green/blue sections, like the bars for completed sprints.

For example, in the chart above, if your team *had not* completed more than 2 story points in sprint 10, then the work completed in sprint 7, sprint 8, and sprint 9 would be used to calculate the velocity — rather than sprint 8, sprint 9, and sprint 10.

Other functionalities

The following questions and answers cover the other key functions of the Epic Burndown report:

What determines the first and last sprints shown on the chart?

- The **first sprint** shown is the one that contains the first issue (in the epic) that transitions out of the 'To Do' status, i.e. work is started on the epic.
- The **last sprint** shown is the one when all work is completed for the epic; or if work remains, then the predicted sprint when work will be finished.

The mapping of statuses to your board determines when an issue is considered 'To Do' or 'Done'. See [Configuring columns](#) for more information.

⌄ [How does the percentage of unestimated issues affect the report?](#)

The Epic Burndown report can only make predictions based on the estimated issues in your epic. This does not include issues that cannot be estimated (e.g. you have configured an issue type to not have the Story Points field). If you have a high percentage of unestimated issues, then the predictions in the report will not be reliable (the **% unestimated issues** label is colored red when the percentage is above 30%).

For example, if you have only estimated 10% of the issues in the epic, then the report predicts the completion of work for the epic based on the 10% of the total issues. In reality, your team probably has much more work left to complete.

⌄ [What changes affect the original estimate and what changes affect the scope \(work added\)?](#)

The following changes affect the original estimate of a sprint:

- An issue in an epic (before it started) is estimated (estimate is added)
- An issue in an epic (before it started) is re-estimated (estimate changes)

The following changes affect the scope of a sprint:

- An issue is added to an epic (after it was started) with an existing estimate
- An issue that was added to an epic (after it was started) is estimated (estimate is added)
- An issue that was added to an epic (after it was started) is re-estimated (estimate changes). Note, if the issue is re-estimated in a later sprint, the scope is retroactively adjusted in the sprint that the issue was originally added to.

⌄ [If work is completed outside of a sprint, how is it represented?](#)

Any change (burndown or scope) that occurs outside a sprint will be shown as part of the sprint with the latest start date before the change date.

⌄ [If a completed issue is reopened or added/removed from an epic, how is it represented?](#)

Issue completed in a sprint, then reopened:

- The issue will not be shown in the earlier sprint.

Issue completed in an epic, but removed from the epic afterwards:

- The scope will remain unchanged and the work completed is still shown.

Issue completed in another epic, but later included in the epic (shown on the report):

- The scope will remain unchanged.

Issue completed in a sprint, but only added to the epic afterwards:

- The issue will be shown on the report, as if it was always part of the epic.

⌄ [What if my issue is in an unmapped status?](#)

If your issue is in an unmapped status (i.e. status not mapped to a column), it will not be considered in the Epic Burndown report. That is, it won't be included in the sprint bars, the % unestimated issues, remaining story points, etc.

Known issues

If you encounter an issue that is not on this list, please [raise it in our issue tracker](#).

⌄ [Click to view known issues...](#)

Key Summary T P Status

No issues found

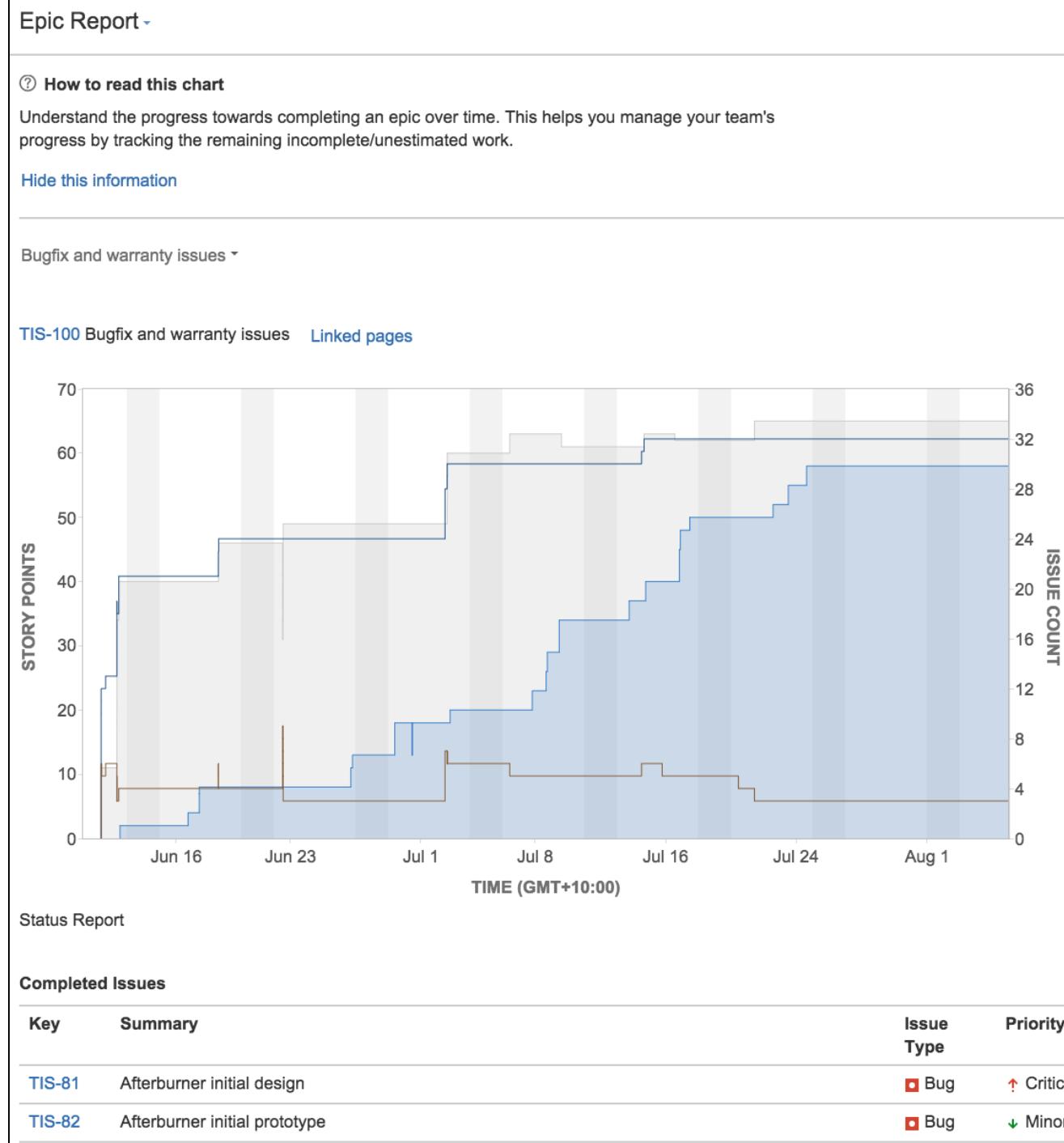
Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Configuring working days
- Configuring estimation and tracking
- Epic Report
- Release Burndown

Epic Report



Screenshot: Epic Report

The Epic Report shows a list of complete, incomplete, and unestimated issues in an epic. It is particularly useful in planning work for an epic that may extend over multiple sprints.

Use the Epic Report to understand the progress towards completing an epic over time, and to track the amount of remaining work that's incomplete or unestimated.

Before you begin

- This page only applies to Scrum boards.

On this page:

- Before you begin
- Viewing the Epic Report
- Understanding the Epic Report
- Next steps

Viewing the Epic Report

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Epic Report**.
3. Select the relevant epic from the epic drop-down.
 - Click **View in Issue Navigator** to jump to the **Issue Navigator** and see a list of all the epic's issues.

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

Understanding the Epic Report

Before you start using the Epic Report, you should get to know how it works. The following information will help you understand the key functionalities of the Epic Report:

- The Epic Report is based on your board's column mapping. An issue is considered to be 'To Do' when it is in a status that has been mapped to the left-most column of your board. Similarly, an issue is considered to be 'Done' when it is in a status that has been mapped to the right-most column of your board. See [Configuring columns](#) for more information.
- The graph will look different if you are using Issue Count as your Estimation Statistic (rather than Story Points, as shown in the screenshot above).

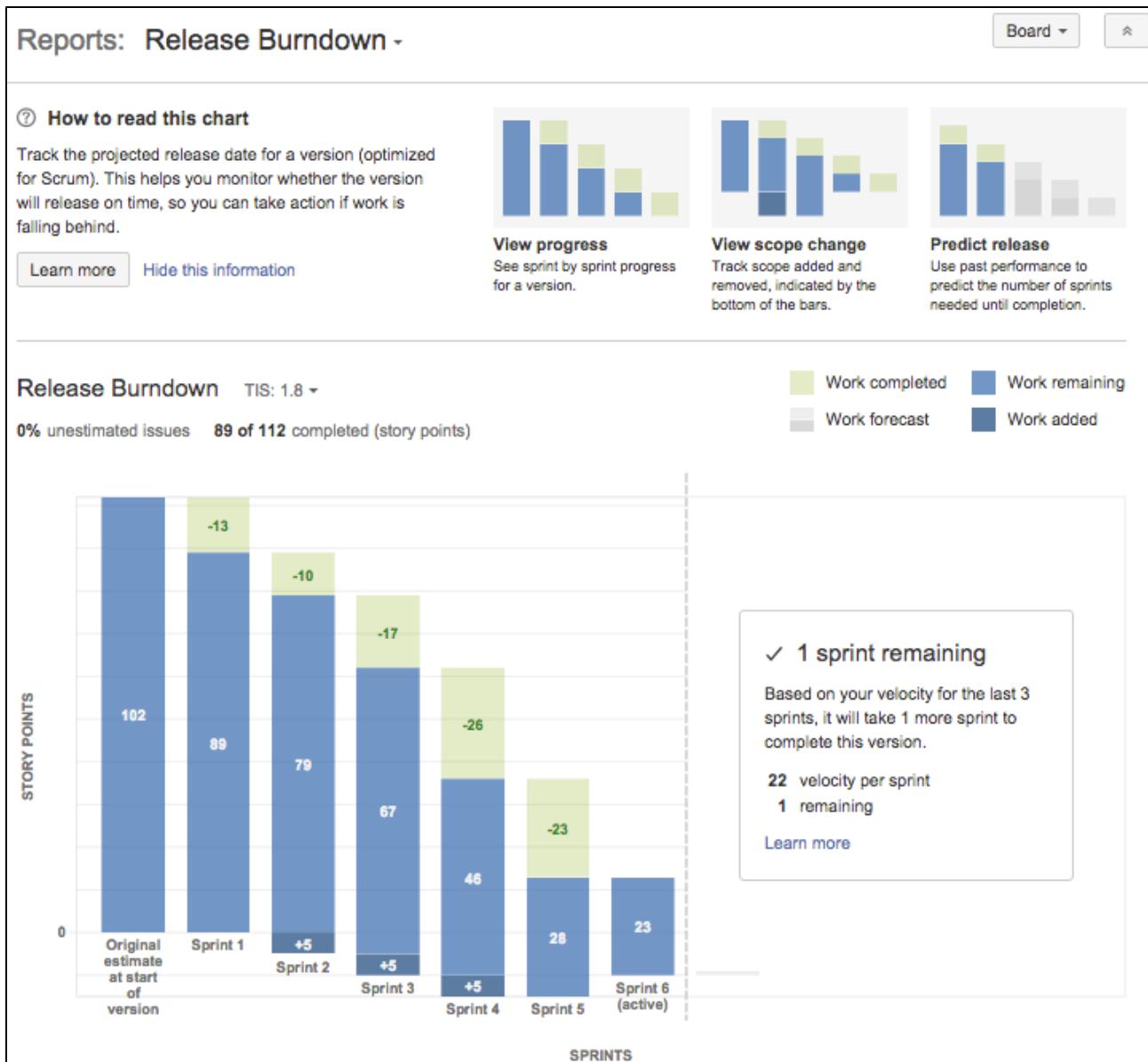
Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Working with epics
- Configuring estimation and tracking

Release Burndown



Screenshot: Release Burndown report (story points)

About the Release Burndown report

The Release Burndown report shows you how your team is progressing against the work for a release. In JIRA Software, there is no 'release' entity — a version is equivalent to a release (hence, the term 'version' will be used instead of 'release' in this document). The report will show data based on the estimation statistic that your board is using.

Here are some of the ways that you could use a Release Burndown report:

- See how quickly your team is working through the backlog,
- See how work added and removed during the sprint has affected your team's overall progress, and
- Predict how many sprints it will take to complete the work for a version, based on past sprints and changes during the sprints.

If you have used the Version Report before, you will notice some similarities. However, the Release Burndown report is optimized for scrum teams that work in sprints — which makes tracking much easier.

On this page:

- About the Release Burndown report
- Viewing the Release Burndown report
- Printing the Release Burndown report
- Understanding the Release Burndown report
- Known issues
- Next steps

Viewing the Release Burndown report

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Release Burndown**.
3. Select the relevant version from the Release Burndown drop-down. You will be able to choose from versions that are in projects configured for your board (via the board's filter).

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

 If you are using Internet Explorer 8, the Release Burndown will not work.

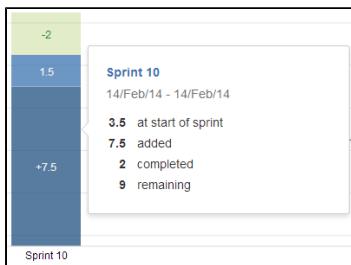
Printing the Release Burndown report

To print the report, view the report and use the print functionality for your browser. The report will fit on either A4- or Letter-sized pages in both portrait and landscape modes (note, there is a known issue printing in landscape using Chrome).

Understanding the Release Burndown report

Before you start using the Release Burndown report, you should get to know how it works.

The sprint bar



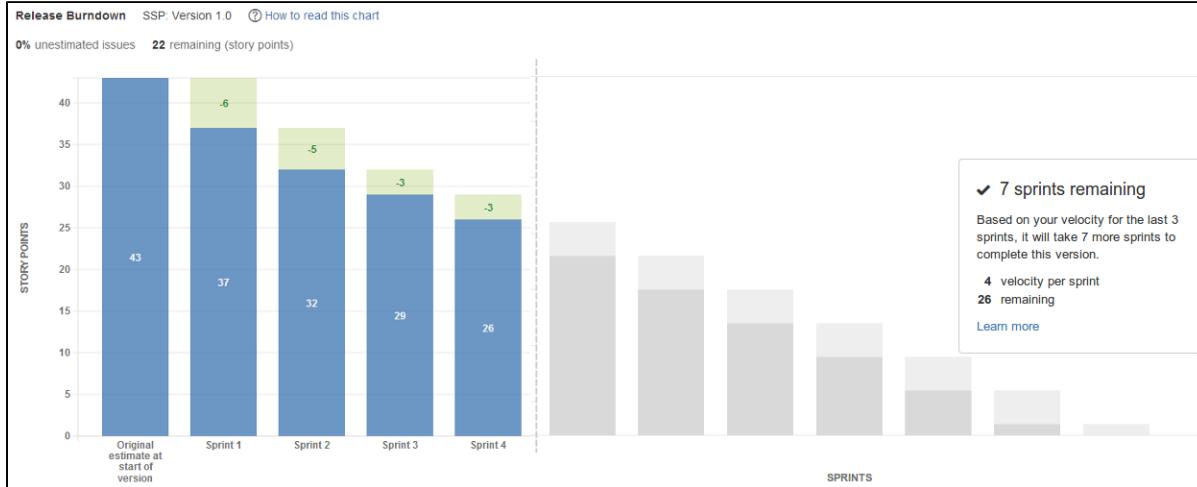
- **Light green section** = work completed during the sprint.
Note, if a bar is completely light green, you won't be able to tell how much of the work completed was originally estimated or not. To find out this information, click the bar to view the details.
- **Light blue section** = work that is remaining in the version, out of the total work estimated for the version at the start of the sprint.
- **Dark blue section** = work that was added during the sprint, but not originally included (i.e. scope change).
- **Light green section + light blue section** = total work in the version, that was originally estimated at the start of the sprint.
- **Light blue section + dark blue section** = total work in the version, that is remaining at the end of the sprint.
- **Bars with grey sections** = predicted sprints (see [below](#)).

Predicted sprints

Predicted sprints are calculated based on your team's velocity* (amount of work completed in the last three sprints), and the total work remaining in your backlog. Scope change is not considered when calculating the velocity*, but is included in the total work remaining.

* not the same as the velocity described in the [Velocity Chart](#).

Consider the following example:



- Assessing the outstanding work:** 26 story points are remaining for the version, at the start of the current sprint (sprint 5).

Note, the '22 remaining (story points)' label (at the top of every Release Burndown report) subtracts the 4 story points that are predicted to be completed in the current sprint.

- Calculating the velocity:** 11 story points were completed in the last three sprints (sprint 2, sprint 3, and sprint 4). This averages out to a velocity of 4 story points per sprint, rounding to nearest story point.
- Predicting the remaining sprints:** At a velocity of 4 story points per sprint, it will take 7 more sprints (including the current sprint) to complete the work for the version: 26 story points. That is, 6 sprints of 4 story points, plus one final sprint of 2 story points.

Is my current sprint counted when calculating my team's velocity?

The current sprint is usually not counted when calculating the team's velocity. In the example above, the current sprint bar shows grey sections (like the bars for the predicted sprints) to represent this. The exception is when you have already *completed more work in the current sprint than the work that was predicted to be completed*. In this case, the current sprint (and the actual work completed) is used as one of the three sprints used to calculate the velocity. Also, the sprint bar will show green/blue sections, like the bars for completed sprints.

For example, in the chart above, if your team had already completed more than 4 story points in sprint 5, then the work completed in sprint 3, sprint 4, and sprint 5 would be used to calculate the velocity, rather than sprint 2, sprint 3 and sprint 4.

Other functionalities

The following questions and answers cover the other key functions of the Release Burndown report:

Why don't the dates for the first/last sprints on my report match the Start date/Release date configured for my version?

The **Start date** and **Release date** configured for the version are shown on the bottom of the report as the **Planned start date** and **Planned end date**. However, these are *planned* dates, and they do not determine the first and last sprints shown on the report.

- The **first sprint** shown is the one that contains the first issue (in the version) that transitions out of the 'To Do' status, i.e. work is started on the version.
- The **last sprint** shown is the one when all work is completed for the version; or if work remains, then the predicted sprint when work will be finished.

The mapping of statuses to your board determines when an issue is considered 'To Do' or 'Done'. See [Configuring issue status](#)

nfiguring columns for more information.

▼ [How does the percentage of unestimated issues affect the report?](#)

The Release Burndown report can only make predictions based on the estimated issues in your version. If you have a high percentage of unestimated issues, then the predictions in the report will not be reliable (the **% unestimated issues** label is colored red when the percentage is above 30%, to help make you aware of this).

For example, if you have only estimated 10% of the issues in the version, then the report predicts the completion of work for the version based on the 10% of the total issues. In reality, your team probably has much more work left to complete.

▼ [What changes affect the original estimate and what changes affect the scope \(work added\)?](#)

The following changes affect the original estimate of a sprint:

- An issue in a version (before it started) is estimated (estimate is added)
- An issue in a version (before it started) is re-estimated (estimate changes)

The following changes affect the scope of a sprint:

- An issue is added to a version (after it was started) with an existing estimate
- An issue that was added to a version (after it was started) is estimated (estimate is added)
- An issue that was added to a version (after it was started) is re-estimated (estimate changes). Note, if the issue is re-estimated in a later sprint, the scope is retroactively adjusted in the sprint that the issue was originally added to.

▼ [If work is completed outside of a sprint, how is it represented?](#)

Any change (burndown or scope) that occurs outside a sprint will be shown as part of the sprint with the latest start date before the change date.

▼ [If a completed issue is reopened or added/removed from a version, how is it represented?](#)

Issue completed in a sprint, then reopened:

- The issue will not be shown in the earlier sprint.

Issue completed in a version, but removed from the version afterwards:

- The scope will remain unchanged and the work completed is still shown.

Issue completed in another version or without a version, but later included in the version (shown on the report):

- The scope will remain unchanged.

Issue completed in a sprint, but only added to the version afterwards:

- The issue will be shown on the report, as if it was always part of the version.

▼ [What if my issue is in an unmapped status?](#)

If your issue is in an unmapped status (i.e. status not mapped to a column), it will not be considered in the Release Burndown chart. That is, it won't be included in the sprint bars, the % unestimated issues, remaining story points, etc.

Known issues

If you encounter an issue that is not on this list, please [raise it in our issue tracker](#).

▼ [Click to view known issues...](#)

Key Summary T P Status

No issues found

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- Planning a version
- Configuring working days
- Configuring estimation and tracking
- Version Report
- Epic Burndown

Sprint Report

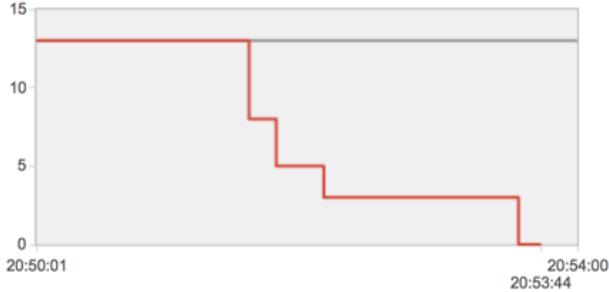
Reports: Sprint Report Board ▾ 

② How to read this chart
Understand the work completed or pushed back to the backlog in each sprint. This helps you determine if your team is overcommitting or if there is excessive scope creep.

[Hide this information](#)

Sprint Report Sprint 1 ▾

Closed Sprint, ended by [Alana Grant](#) 14/Jun/15 8:50 PM - 14/Jun/15 8:53 PM [Linked pages](#) [View Sprint 1 in Issue Navigator](#)



The chart shows a red step-down line representing work completed. It starts at a value of 13 at 20:50:01, remains flat until 20:53:44, then drops to 5. From 20:53:44 to 20:54:00, it drops again to 3, and finally to 0 at 20:54:00.

Status Report

Completed Issues [View in Issue Navigator](#)

Key	Summary	Issue Type	Priority	Status	Story Points (13)
TIS-81	Afterburner initial design	Story	Minor	CLOSED	5
TIS-82	Afterburner initial prototype	Story	Major	CLOSED	2

Screenshot: Sprint Report

The Sprint Report shows the list of issues in each sprint. It is useful for your Sprint Retrospective meetings, and also for mid-sprint progress checks.

Tip: If you have JIRA Software connected to Confluence, you can create a 'Retrospectives' page via the **Linked pages** link. See [Linking a Confluence page to a sprint](#) for details.

Before you begin

- This page only applies to Scrum boards.

On this page:

- Before you begin
- Viewing the Sprint Report
- Understanding the Sprint Report
- Next steps

Viewing the Sprint Report

1. Click **Boards** (in header) > select your desired board.

2. Click **Reports**, then select **Sprint Report**.
3. Select the relevant sprint from the sprint drop-down.

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

Understanding the Sprint Report

Before you start using the Sprint Report, you should get to know how it works. The following information will help you understand the key functionalities of the Sprint Report:

- The Sprint Report is board-specific – that is, it will only include issues that match your board's saved filter.
- Issues added after the sprint starts are indicated with an asterisk.
- An issue is considered to be 'To Do' when it is in a status that has been mapped to the left-most column of your board. Similarly, an issue is considered to be 'Done' when it is in a status that has been mapped to the right-most column of your board. See [Configuring columns](#) for more information.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Planning sprints](#)
- [Completing a sprint](#)
- [Configuring working days](#)

Velocity Chart

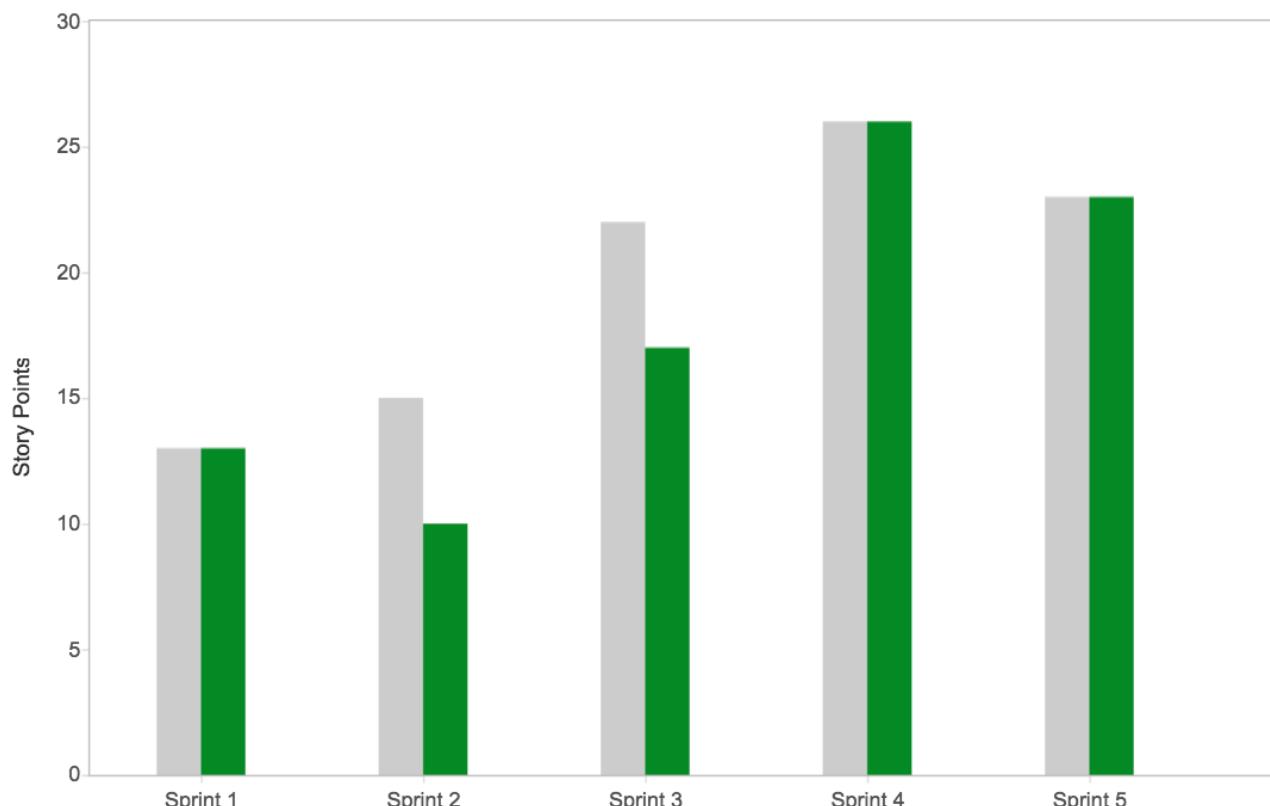
Reports: Velocity Chart -

② How to read this chart

Track the amount of work completed from sprint to sprint. This helps you determine your team's velocity and estimate the work your team can realistically achieve in future sprints.

[Hide this information](#)

Velocity Chart



Screenshot: Velocity Chart

The Velocity Chart shows the amount of value delivered in each sprint, enabling you to predict the amount of work the team can get done in future sprints. It is useful during your sprint planning meetings, to help you decide how much work you can feasibly commit to.

You can estimate your team's velocity based on the total Estimate (for all completed stories) for each recent sprint. This isn't an exact science — looking at several sprints will help you get a feel for the trend. For each sprint, the Velocity Chart shows the sum of the Estimates for complete and incomplete stories. Estimates can be based on story points, business value,

hours, issue count, or any numeric field of your choice (see [Configuring estimation and tracking](#)).

The velocity can be estimated as the average, over several recent sprints, of the sum of the estimates for the amount of work completed by a team per sprint — so in the chart above, the velocity = $(13 + 10 + 17 + 26 + 23) / 5 = 17.8$. A team's recent velocity can be useful in helping to predict how much work can be completed by the team in a future sprint.

On this page:

- [Viewing the Velocity Chart](#)
- [Understanding the Velocity Chart](#)
- [Next steps](#)

Viewing the Velocity Chart

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Velocity Chart**.
3. The Velocity Chart will be displayed, showing your last seven completed sprints.

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

Understanding the Velocity Chart

Before you start using the Velocity Chart, you should get to know how it works. The following information will help you understand the key functionalities of the Velocity Chart:

- The Velocity Chart is board-specific – that is, it will only include issues that match your board's saved filter.
- The Velocity Chart is based on your board's column mapping. An issue is considered to be 'To Do' when it is in a status that has been mapped to the left-most column of your board. Similarly, an issue is considered to be 'Done' when it is in a status that has been mapped to the right-most column of your board. See [Configuring columns](#) for more information.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Configuring estimation and tracking](#)

Version Report

Reports: Version Report

[⑦ How to read this chart](#)

Track the projected release date for a version. This helps you monitor whether the version will release on time, so you can take action if work is falling behind.

[Hide this information](#)

Version Report TIS: 3.0 ▾

Scheduled Release Date: 14/Sep/15

Release Date: 14/Sep/15

Today

Story Points

Percent

Time (GMT+10:00)

Status Report

Completed Issues

Key	Summary	Issue Type	Priority
TIS-23	Engage JetShuttle SpaceWays for short distance space travel	Travel Provider	Major
TIS-45	Email non registered users to sign up with Teams In Space	Story	Critical
TIS-20	Engage Saturn Shuttle Lines for group tours	Travel Provider	Major

Screenshot: Version Report

The Version Report shows your team's progress towards the completion of a version. The Version Report also shows you the predicted Release Date, based on your team's average rate of progress (velocity) since the start of the version, and the estimated amount of work remaining.

Before you begin

- This page only applies to Scrum boards.

On this page:

- Before you begin
- Viewing the Version Report
- Understanding the Version Report
- Next steps

Viewing the Version Report

1. Click **Boards** (in header) > select your desired board.
2. Click **Reports**, then select **Version Report**.
3. Select the relevant version from the Version Report drop-down.

*Tip: Click **How to read this chart** at the top of the report to view a short description of the report.*

Understanding the Version Report

Before you start using the Version Report, you should get to know how it works. The following information will help you understand the key functionalities of the Version Report:

- The Version Report is board-specific – that is, it will only include issues that match your board's saved filter.
- The Version Report will exclude issues of a 'sub-task type'.
- The Version Report shows 'Released' versions but not 'Archived' versions. For more about version status, see [Managing versions](#) (*JIRA Admin documentation*).
- The Version Report is based on your board's column mapping. An issue is considered to be 'To Do' when it is in a status that has been mapped to the left-most column of your board. Similarly, an issue is considered to be 'Done' when it is in a status that has been mapped to the right-most column of your board. See [Configuring columns](#) for more information.
- The horizontal axis starts on the version's Start Date; or if no Start Date is specified, the date on which an issue was first added to the version. The graph shows the state your version was in at any given point in time, in terms of your total and completed Story Points (or other Estimation Statistic of your choice), so that you can see how the scope may have changed, and how you are progressing towards completion of the estimated work.
- The graph shows you the following predictions:
 - the **Predicted Release Date** (*blue line*) – that is, the date at which you can expect all issues in your version to be complete, based on your average daily velocity and the amount of estimated work remaining.
 - the **Predicted Release Date (Optimistic)** (*shaded area to the left of the blue line*) – that is, the earliest date by which you might expect the version to be complete. (The 'optimistic' date is calculated by adding 10% to the average daily velocity.)
 - the **Predicted Release Date (Pessimistic)** (*shaded area to the right of the blue line*) – that is, the latest date by which you might expect the version to be complete. (The 'pessimistic' date is calculated by subtracting 10% from the average daily velocity.)
- 10% of the estimated work for the version will need to be complete before the predictions can be calculated.

Next steps

i Need help? If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

Read the following related topics:

- [Planning a version](#)
- [Configuring working days](#)
- [Configuring estimation and tracking](#)

Working as a team member in an agile project

If you are a team member working with JIRA Software, you are in the right place! These pages will be useful to you, if you are a developer, designer, QA person, technical writer, or anyone else that uses (but does not configure) JIRA Software. The topics in this chapter will help you use JIRA Software to manage your work.

In the wrong place? See the following chapters instead:

- If you are responsible for running an agile project with JIRA Software, see [Leading an agile project](#).
- If you are an administrator installing or configuring the JIRA Software server, see [Administering JIRA Software](#).

Where do I start?

- If this is the first time you're using JIRA Software, read our [Getting started as a JIRA Software user](#) before you start reading other topics.
- If you're already familiar with JIRA Software, use the search box below to find the topic you need.

Search the topics in 'Developing in an agile project':

Overview

The topics in this chapter cover the activities that you will need to do at any given time when you start using JIRA Software.

Note, you will be working with boards in an agile project, and it may be worthy to know how to use the project sidebar in JIRA Software. See [Using the project sidebar](#) for more information.

Searching for issues

While you're doing development work in JIRA Software, there will come a time when you'll have to search for issues. The issue you're currently working on may be related to an issue your team worked on before, and you want to find that issue. There are many ways for you to search for issues in JIRA Software.

[Learn more...](#)

Working with issues

There's so much that you can do in JIRA Software when you start working on your issues. You can create issues, estimate how long it takes to complete issues, collaborate with your team while working on issues, reference issues to and from your source repository, view the development information of issues, etc.

[Learn more...](#)

Configuring dashboards

You can add, configure, and customize dashboards in JIRA Software. With gadgets, you can choose the information that appears on your dashboard or wallboard — if you're using dashboards for that purpose. You can use the gadgets that help keep you and your team at your most productive.

[Learn more...](#)

Managing your user profile

You may want to personalize some details in your user profile as you go about your development work. You can edit your user details, change your avatar or your password, set your JIRA homepage, manage your email notifications, and view and manage your OAuth tokens, etc.

[Learn more...](#)

Searching for issues

Can't find the issue that you are looking for? This page will show you how to search for issues in JIRA. Any user can search for issues, although they will only see issue results from projects where they can view issues (i.e. 'Browse Project' permission).

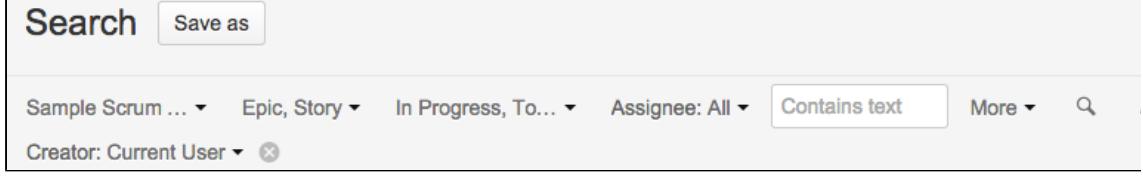
You'll find a step-by-step guide below that will show you how to run a search and use the search results. If you want more details on anything described on this page, see the related topics at the bottom of the page.

On this page:

- 1. Define your search criteria
- 2. Change your view of the search results
- 3. Working with the search results
- 4. Save your search
- Next steps

1. Define your search criteria

The first step in searching for issues is to define the criteria for your new search. You can define your search criteria in three different ways: using the quick search, using the basic search, or using the advanced search.

Quick search	<p>The quick search is the fastest way to define search criteria. However, it is less precise than other complex queries (e.g. project = JIRA AND status = Open AND priority = High). If your search criteria is not complex, for example, you know the project key and some key words for the issue title, then use the quick search.</p> <p>To use the quick search: Enter your search criteria in the search box in the header and press Enter.</p> <p><i>Tip: If you know the issue key or project key, enter it before other search terms, e.g. "JRA help library".</i></p> 
Basic search	<p>The basic search is more precise than the quick search, but easier to use than the advanced search. It has a user-friendly interface that lets you define complex queries, without needing to know how to use the syntax of searching).</p> <p>To use the basic search: Navigate to Issues (in header) > Search for issues, then enter your search criteria.</p> <p><i>Tip: If the advanced search is shown instead of the basic search, click Basic next to the search bar.</i></p> 

Advanced search	<p>The advanced search is the most powerful of the three search methods. You can specify criteria in the other searches (e.g. ORDER BY clause). However, you need to know how to construct structured JQL to use this feature.</p> <p>To use the advanced search: Navigate to Issues (in header) > Search for issues, then enter <code>project = SSP AND issuetype in (Epic, Story) AND status in ("In Progress", "To Do") AND creator in (currentUser())</code></p> <p><i>Tip: If the basic search is shown instead of the advanced search, click Advanced next to the icon.</i></p>
------------------------	--

2. Change your view of the search results

You have crafted the perfect search criteria and run the search. Your search results will be displayed in the issue navigator. The issue navigator allows you to change how the search results are displayed. For example, you may want to bring high priority issues to the top or hide certain fields.

- **Change the sort order:** Click the column name.
- **Show/hide columns:** Click **Columns** and choose the desired columns.

T	Key	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due	Fix Version/s
<input checked="" type="checkbox"/>	ANGRY-304	Red Angry Nerd is scary	Unassigned	Bartosz Gatz	↓ ↕	Open	Unresolved	21/Mar/13	21/Mar/13		
<input checked="" type="checkbox"/>	ANGRY-299	My red nerd is not working properly	Unassigned	Susan Griffin	↓ ↕	Open	Unresolved	15/Mar/13	15/Mar/13		
<input checked="" type="checkbox"/>	ANGRY-158	ANGRY-13 / give the red nerd a yellow hat	Unassigned	Rosie Jameson	↑ ↕	Open	Unresolved	16/Apr/12	10/Aug/12	1.2	
<input checked="" type="checkbox"/>	ANGRY-13	I don't like the red nerd	Unassigned	Edwin Wong	↑ ↕	In Progress	Unresolved	27/May/11	25/Feb/13	1.2	
<input checked="" type="checkbox"/>	ANGRY-306	Red Nerd should have his mouth open on impact	Unassigned	Bartosz Gatz	↓ ↕	Open	Unresolved	21/Mar/13	05/Apr/13	31/Mar/13	
<input checked="" type="checkbox"/>	ANGRY-35	Bug on Atlassian - Angry Nerds - Red bird should bounce not flip	Roy Krishna	Edwin Wong	↑ ↕	Resolved	Fixed	03/Jun/11	20/Jul/12	1.2	
<input checked="" type="checkbox"/>	ANGRY-79	Fix nerd's hair-styling	Bryan Rollins	Edwin Wong	↓ ↕	Closed	Fixed	12/Jul/11	20/Jan/12	1.3	
<input checked="" type="checkbox"/>	ANGRY-70	Some graphical glitches	Christina Bang	Edwin Wong	✗ ↕	In Progress	Unresolved	05/Jun/11	18/Sep/12	1.2	

3. Working with the search results

You've got the search results displaying the way that you want. Now you can work with the actual issues in the search results. The issue navigator lets you action individual issues, as well as the entire set of issues returned by your search.

Individual issues:

- **View the issue:** Click the issue key or name.
- **Action individual issues:** Click the cog icon next to the issue row and select an option.

All issues in the search results:

- **Export the search results to different formats, like Excel and XML:** Click **Export** and select the desired format.
- **Share the search results:** Click **Share**, then enter the recipient's details.
- **Create an RSS feed:** Click **Export > RSS (Issues)** or **RSS (Comments)**.
- **Bulk modify issues in search results:** Click **Tools** and select all **<n> issue(s)** under **Bulk Change**.

4. Save your search

If you frequently run the same search, you can save the search criteria as a filter. This saves you from having to manually redefine the search criteria every time. JIRA applications also include a number of predefined system filters for common queries, such as 'My Open Issues', 'Reported by Me', 'Recently Viewed', and 'All Issues'.

To save your search as a filter: On the search results page, click **Save as** and enter a name for the filter. Your new filter will be shown in the left panel with your other favorite filters, filters shared with you, and the system filters. To run a filter, just click it.

Next steps

Read the following related topics:

- Quick searching
- Basic searching
- Advanced searching
- Saving your search as a filter
- Working with search results

Basic searching

The basic search provides a user-friendly interface that lets you define complex queries, without needing to know how to use JQL (advanced searching).

- If you don't have complex search criteria, you may want to use [quick search](#) instead.
- If you are comfortable with the JIRA Query Language (JQL), you may want to use [advanced search](#) instead. This search is more powerful than than the basic search.

Screenshot: Basic search

The screenshot shows the JIRA basic search interface. On the left, there is a sidebar with filters like 'New filter', 'Find filters', 'My Open Issues', 'Reported by Me', 'Recently Viewed', 'All Issues', and 'FAVORITE FILTERS' (including '6.2-OD5 Docs' and 'Enterprise issues ass...'). The main area has a search bar with 'Search' and 'Save as' buttons, and dropdowns for 'Sample Scrum ...', 'Epic, Story ...', 'In Progress, To...', 'Assignee: All ...', 'Contains text', 'More', and 'Advanced'. Below the search bar is a 'Creator: Current User' dropdown. The results table shows 1-4 of 4 issues:

T	Key	Summary	Assignee	Reporter	P	Status	Created	Updated
	SSP-38	Estimates epic	Unassigned	Andrew Lui	▼	TO DO	03/Sep/14	23/Oct/14
	SSP-37	Filters epic	Unassigned	Andrew Lui	▼	TO DO	03/Sep/14	03/Sep/14
	SSP-36	Ranking epic	Unassigned	Andrew Lui	▼	TO DO	03/Sep/14	03/Sep/14
	SSP-35	A test story	Andrew Lui	Andrew Lui	▼	TO DO	05/Jun/14	03/Sep/14

At the bottom, it says '1-4 of 4'.

On this page:

- Basic searching
- Running a saved search
- Troubleshooting
- Next steps

Basic searching

1. Choose **Issues > Search for issues**.

- If there are existing search criteria, click the **New filter** button to reset the search criteria.
 - If the advanced search is shown instead of the basic search, click **Basic** (next to the icon).
- [Why can't I switch between basic and advanced search?](#)

In general, a query created using basic search will be able to be translated to advanced search, and back again. However, a query created using advanced search may not be able to be translated to basic search, particularly if:

- the query contains an OR operator (note you can have an IN operator and it will be translated, e.g. project in (A, B))
 - i.e. even though this query: (project = JRA OR project = CONF) is equivalent to this query: (project in (JRA, CONF)), only the second query will be translated.
- the query contains a NOT operator
- the query contains an EMPTY operator
- the query contains any of the comparison operators: !=, IS, IS NOT, >, >=, <, <=
- the query specifies a field and value that is related to a project (e.g. version, component, custom fields) and the project is not explicitly included in the query (e.g. fixVersion = "4.0", without the AND project=JRA). This is especially tricky with custom fields since they can be configured on a Project/Issue Type basis. The general rule of thumb is that if the query cannot be created in the basic search form, then it will not be able to be translated from advanced search to basic search.

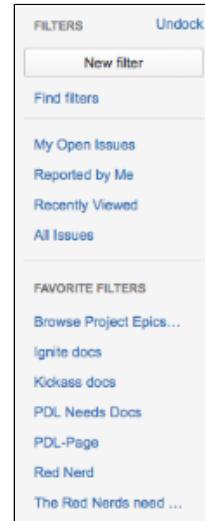
2. Enter the criteria for the search. You can search against specific fields and/or search for specific text.
 - If you are searching against a field and can't find the field you want, or the field is displaying greyed out text, see the [Troubleshooting section below](#).
 - If you are searching for text, you can use special characters and modifiers in your search text, such as wildcards and logical operators. See [Search syntax for text fields](#).
3. The search results will automatically update in the issue navigator, unless your administrator has disabled automatic updates of search results. If so, you will need to click the **Update** button on the field drop-down after every change.

Running a saved search

Saved searches (also known as [filters](#)) are shown in the left panel, when using basic search. If the left panel is not showing, hover your mouse over the left side of the screen to display it.

To run a filter, e.g. **My Open Issues**, simply click it. The search criteria for the basic search will be set, and the search results will be displayed.

Note, clicking the **Recently Viewed** filter will switch you to the advanced search, as the basic search cannot represent the ORDER BY clause in this filter.



Troubleshooting

▼ Why can't I find the field I want to choose?

Some fields are only valid for a particular *project/issue type context*. For these fields, you must select the applicable project/issue type. Otherwise, the field is not available for selection.

▼ Why are the field criteria displaying in grey text?

Some fields are only valid for a particular *project/issue type context*. If you choose a field in your search, then remove all projects/issue types that reference the field, then the field is invalid. The invalid field does not apply to your search and displays in grey text.

▼ Why is there a red exclamation mark in my field?

Some field values are only valid for a particular *project/issue type context*. For example, you may have configured a project to use a status *In QA Review* in its workflow. If you select this project and status in your search, then change the search to filter for a project that doesn't use *In QA Review*, the status will be invalid and ignored in the search.

Why don't my search results automatically update?

Your search results will always update automatically whenever any fields are changed, provided that your administrator has not disabled automatic updates of search results. Ask your administrator whether they have disabled automatic updates of search results.

Next steps

Read the following related topics:

- [Searching for issues](#)
- [Advanced searching](#)
- [Saving your search as a filter](#)
- [Working with search results](#) — find out how to use the issue navigator, export your search results, bulk modify issues, and share your search results.

Quick searching

Sometimes, you just want to be able to get to the particular issue that you are interested in. Other times, you can't remember what the issue was, but you remember that it was an open issue, assigned to you. Quick search can help you in these scenarios.

Quick searching

The **Quick Search** box is located at the top right of your screen. To use it, just starting typing what you are looking for.



On this page:

- [Quick searching](#)
- [Understanding quick searching](#)
- [Searching issues from your browser's search box](#)
- [Next steps](#)

Understanding quick searching

Read the following topics to learn how to get the most out of quick searching:
[Jumping to an issue](#) | [Smart querying](#) | [Free-text searching](#)

Jumping to an issue

If you type in the **key** of an issue, you will jump straight to that issue. For example, if you type in 'ABC-107' (or 'abc-107'), and press the **Enter** button, you will be redirected to the issue 'ABC-107'.

In many cases, you do not even need to type in the full key, but just the numerical part. If you are currently working on the 'ABC' project, and you type in '123', you will be redirected to 'ABC-123'.

Smart querying

Quick search also enables you to perform 'smart' searches with minimal typing. For example, to find all the open bugs in the 'TEST' project, you could simply type 'test open bugs' and quick search would locate them all for you.

Your search results will be displayed in the Issue Navigator, where you can view them in a variety of useful formats (Excel, XML, etc.).

The search terms that quick search recognizes are:

Search Term	Description	Examples
my	Find issues assigned to me.	my open bugs

r:	Find issues reported by you, another user or with no reporter, using the prefix <code>r:</code> followed by a specific reporter term, such as <code>me</code> , a username or <code>none</code> . <i>Note that there can be no spaces between "r:" and the specific reporter term.</i>	<code>r:me</code> — finds issues reported by you. <code>r:samuel</code> — finds issues reported by the user whose username is "samuel". <code>r:none</code> — finds issues with no reporter.
<project name> or <project key>	Find issues in a particular project.	<code>test project</code> <code>TST</code> <code>tst</code>
overdue	Find issues that were due before today.	<code>overdue</code>
created: updated: due:	Find issues with a particular Created, Updated, or Due Date using the prefixes <code>created:</code> , <code>updated:</code> , or <code>due:</code> , respectively. For the date range, you can use <code>today</code> , <code>tomorrow</code> , <code>yesterday</code> , a single date range (e.g. <code>'-1w'</code>), or two date ranges (e.g. <code>'-1w,1w'</code>). Note that date ranges cannot have spaces in them. Valid date/time abbreviations are: <code>'w'</code> (week), <code>'d'</code> (day), <code>'h'</code> (hour), <code>'m'</code> (minute).	<code>created:today</code> <code>created:yesterday</code> <code>updated:-1w</code> — finds issues updated in the last week. <code>due:1w</code> — finds issues due in the next week. <code>due:-1d,1w</code> — finds issues due from yesterday to next week. <code>created:-1w,-30m</code> — finds issues created from one week ago, to 30 minutes ago. <code>created:-1d</code> <code>updated:-4h</code> — finds issues created in the last day, updated in the last 4 hours.
<priority>	Find issues with a particular Priority.	<code>blocker</code> <code>major</code> <code>trivial</code>
<issue type>	Find issues with a particular Issue Type. Note that you can also use plurals.	<code>bug</code> <code>task</code> <code>bugs</code> <code>tasks</code>
<resolution>	Find issues with a particular Resolution.	<code>fixed</code> <code>duplicate</code> <code>cannot reproduce</code>
c:	Find issues with a particular Component(s). You can search across multiple components. <i>Note that there can be no spaces between "c:" and the component name.</i>	<code>c:security</code> — finds issues with a component whose name contains the word "security".

v:	<p>Find issues with a particular Affects Version(s). To find all issues belonging to a 'major' version, use the wildcard symbol '*'.</p> <p><i>Note that there can be no spaces between "v:" and the version name.</i></p>	<p>v:3.0 — finds issues that match the following versions (for example):</p> <ul style="list-style-type: none"> • 3.0 • 3.0 eap • 3.0 beta <p>...but will not match against the following versions (for example):</p> <ul style="list-style-type: none"> • 3.0.1 • 3.0.0.4 <p>That is, it will match against any version that contains the string you specify followed immediately by a space, but not against versions that do not contain a space immediately after the string you specify.</p>
ff:	Find issues with a particular Fix For Version(s). Same usage as v: (above).	
*	Wildcard symbol '*' . Can be used with v: and ff:.	<p>v:3.2* — finds any issue whose version number is (for example):</p> <ul style="list-style-type: none"> • 3.2 • 3.2-beta • 3.2.1 • 3.2.x

In Mozilla-based browsers, try creating a bookmark with URL `http://<your-JIRA-site>/secure/QuickSearch.jspa?searchString=%s` (substituting <your-JIRA-site> with your JIRA instance's URL) and keyword (such as 'j'). Now, typing 'j my open bugs' in the browser URL bar will search your JIRA instance for your open bugs. Or simply type your search term in the Quick Search box, then right-click on the Quick Search box (with your search term shown) and select "Add a Keyword for this search...".

Free-text searching

You can search for any word within the issue(s) you are looking for, provided the word is in one of the following fields:

- Summary
- Description
- Comments

You can combine free-text and keywords together, e.g. "my closed test tasks". You can also use wildcards, e.g. "win*8".

For more information on free-text searching, see [Search syntax for text fields](#).

Searching issues from your browser's search box

If you are using Firefox or Internet Explorer 8 (or later), you can add your JIRA instance as a search

engine/provider via the drop-down menu next to the browser's search box. Once you add your JIRA instance as a search engine/provider in your browser, you can use it at any time to conduct a Quick Search for issues in that JIRA instance.

OpenSearch

JIRA supports this browser search feature as part of the autodiscovery part of the [OpenSearch standard](#), by supplying an [OpenSearch description document](#). This is an XML file that describes the web interface provided by JIRA's search function. Any [client applications](#) that support OpenSearch will be able to add JIRA to their list of search engines.

Next steps

Read the following related topics:

- [Searching for issues](#)

Advanced searching

The advanced search allows you to build structured queries using the JIRA Query Language (JQL) to search for issues. You can specify criteria that cannot be defined in the quick or basic searches (e.g. ORDER BY clause).

- If you don't have complex search criteria, you may want to use [quick search](#) instead.
- If you are not comfortable with the JIRA Query Language (JQL), you may want to use [basic search](#) instead.

Note, JQL is not a database query language, even though it uses SQL-like syntax.

Screenshot: Advanced search

On this page:

- [Advanced searching](#)
- [Understanding advanced searching](#)
- [Reference](#)
- [Running a saved search](#)
- [Next steps](#)

The screenshot shows the JIRA search results for the query "Red Nerd". The results table has columns: T, Key, Summary, Assignee, Reporter, P, Status, Resolution, Created, Updated, Due, Fix Versions. There are 8 results listed, each with a small icon and some text. The first result is ANGRY-304: Red Angry Nerd is scary, assigned to Bartosz Gatz, reporter Susan Griffin, status Open, created 21/Mar/13, updated 21/Mar/13. Other results include ANGRY-158, ANGRY-13, ANGRY-309, ANGRY-35, ANGRY-79, and ANGRY-70.

Advanced searching

1. Navigate to **Issues** (in header) > **Search for issues**.

- If there are existing search criteria, click the **New filter** button to reset the search criteria.
- If the basic search is shown instead of the advanced search, click **Advanced** (next to the icon).

[▼ Why can't I switch between basic and advanced search?](#)

In general, a query created using basic search will be able to be translated to advanced search, and back again. However, a query created using advanced search may not be able to be translated to basic search, particularly if:

- the query contains an OR operator (note you can have an IN operator and it will be translated, e.g. project in (A, B))
 - i.e. even though this query: (project = JRA OR project = CONF) is equivalent to this query: (project in (JRA, CONF)), only the second query will be translated.
- the query contains a NOT operator
- the query contains an EMPTY operator
- the query contains any of the comparison operators: !=, IS, IS NOT, >, >=, <, <=
- the query specifies a field and value that is related to a project (e.g. version,

component, custom fields) and the project is not explicitly included in the query (e.g. `fixVersion = "4.0"`, without the `AND project=JRA`). This is especially tricky with custom fields since they can be configured on a Project/Issue Type basis. The general rule of thumb is that if the query cannot be created in the basic search form, then it will not be able to be translated from advanced search to basic search.

2. Enter your JQL query. As you type, JIRA will offer a list of "auto-complete" suggestions based on the context of your query. Note, auto-complete suggestions only include the first 15 matches, displayed alphabetically, so you may need to enter more text if you can't find a match.

▼ [Why aren't the auto-complete suggestions being shown?](#)

- Your administrator may have disabled the "JQL Auto-complete" feature for your JIRA instance.
- Auto-complete suggestions are not offered for function parameters.
- Auto-complete suggestions are not offered for all fields. Check the [fields](#) reference to see which fields support auto-complete.

3. Press Enter or click 

to run your query. Your search results will display in the issue navigator.

Understanding advanced searching

Read the following topics to learn how to get the most out of advanced searching:

[Constructing JQL queries](#) | [Setting the precedence of operators](#) | [Restricted words and characters](#) | [Performing text searches](#)

Constructing JQL queries

A simple query in JQL (also known as a 'clause') consists of a *field*, followed by an *operator*, followed by one or more *values* or *functions*. For example:

```
project = "TEST"
```

This query will find all issues in the "TEST" project. It uses the "project" *field*, the EQUALS *operator*, and the *value* "TEST".

A more complex query might look like this:

```
project = "TEST" AND assignee = currentUser()
```

This query will find all issues in the "TEST" project where the assignee is the currently logged in user. It uses the "project" *field*, the EQUALS *operator*, the *value* "TEST", the "AND" keyword and the "currentUser()" *function*.

For more information on fields, operators, keywords and functions, see the [Reference section](#) below.

Setting the precedence of operators

You can use parentheses in complex JQL statements to enforce the precedence of operators.

For example, if you want to find all resolved issues in the 'SysAdmin' project, as well as all issues (any status, any project) currently assigned to the system administrator (bobsmith), you can use parentheses to enforce the precedence of the boolean operators in your query, i.e.

```
(status=resolved AND project=SysAdmin) OR assignee=bobsmith
```

Note that if you do not use parentheses, the statement will be evaluated left-to-right.

You can also use parentheses to group clauses, so that you can apply the NOT operator to the group.

Restricted words and characters

Reserved characters

JQL has a list of reserved characters:

space (" ")	+	.	,	;	?		*	/	%	^	\$	#	@	[]
-------------	---	---	---	---	---	--	---	---	---	---	----	---	---	-----

If you wish to use these characters in queries, you need to:

- surround them with quote-marks (you can use either single quote-marks (') or double quote-marks ("));
- **and**, if you are searching a text field and the character is on the list of [reserved characters for text searches](#),
- precede them with two backslashes.

For example:

- `version = "[example]"`
- `summary ~ "\\\\[example\\]"`

Reserved words

JQL also has a list of reserved words. These words need to be surrounded by quote-marks (single or double) if you wish to use them in queries.

[Show me...](#)

"abort", "access", "add", "after", "alias", "all", "alter", "and", "any", "as", "asc", "audit", "avg", "before", "begin", "between", "boolean", "break", "by", "byte", "catch", "cf", "char", "character", "check", "checkpoint", "collate", "collation", "column", "commit", "connect", "continue", "count", "create", "current", "date", "decimal", "declare", "decrement", "default", "defaults", "define", "delete", "delimiter", "desc", "difference", "distinct", "divide", "do", "double", "drop", "else", "empty", "encoding", "end", "equals", "escape", "exclusive", "exec", "execute", "exists", "explain", "false", "fetch", "file", "field", "first", "float", "for", "from", "function", "go", "goto", "grant", "greater", "group", "having", "identified", "if", "immediate", "in", "increment", "index", "initial", "inner", "inout", "input", "insert", "int", "integer", "intersect", "intersection", "into", "is", "isempty", "isnull", "join", "last", "left", "less", "like", "limit", "lock", "long", "max", "min", "minus", "mode", "modify", "modulo", "more", "multiply", "next", "noaudit", "not", "notin", "nowait", "null", "number", "object", "of", "on", "option", "or", "order", "outer", "output", "power", "previous", "prior", "privileges", "public", "raise", "raw", "remainder", "rename", "resource", "return", "returns", "revoke", "right", "row", "rowid", "rownum", "rows", "select", "session", "set", "share", "size", "sqrt", "start", "strict", "string", "subtract", "sum", "synonym", "table", "then", "to", "trans", "transaction", "trigger", "true", "uid", "union", "unique", "update", "user", "validate", "values", "view", "when", "whenever", "where", "while", "with"

Note for JIRA administrators: this list is hard coded in the `JqlStringSupportImpl.java` file.

Performing text searches

You can use Lucene's text-searching features when performing searches on the following fields, using the CONTAINS operator:

Summary, Description, Environment, Comments, custom fields that use the "Free Text Searcher" (i.e. custom fields of the following built-in custom field types: Free Text Field, Text Field, Read-only Text Field).

For more information, see [Search syntax for text fields](#).

Reference

	Description	Reference
--	-------------	-----------

Fields	<p>A field in JQL is a word that represents a JIRA field (or a custom field that has already been defined in JIRA).</p>	<p>Fields reference page</p> <p>▼ Show list of fields</p> <ul style="list-style-type: none">• Affected version• Approvals• Assignee• Attachments• Category• Comment• Component• Created• Creator• Custom field• Customer Request Type• Description• Due• Environment• Epic link• Filter• Fix version• Issue key• Labels• Last viewed• Level• Original estimate• Parent• Priority• Project• Remaining estimate• Reporter• Request channel type• Request last activity time• Resolution• Resolved• Sprint• Status• Summary• Text• Time spent• Type• Updated• Voter• Votes• Watcher• Watchers• Work ratio
---------------	---	--

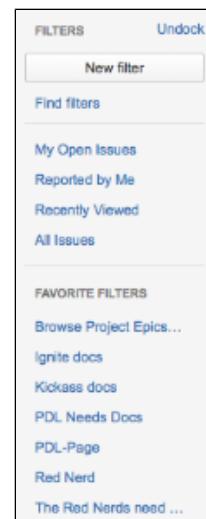
Operators	<p>An operator in JQL is one or more symbols or words that compare the value of a field on its left with one or more values (or functions) on its right, such that only true results are retrieved by the clause. Some operators may use the NOT keyword.</p>	<p>Operators reference page</p> <p>▼ Show list of operators</p> <ul style="list-style-type: none"> • EQUALS: = • NOT EQUALS: != • GREATER THAN: > • GREATER THAN EQUALS: >= • LESS THAN: < • LESS THAN EQUALS: <= • IN • NOT IN • CONTAINS: ~ • DOES NOT CONTAIN: !~ • IS • IS NOT • WAS • WAS IN • WAS NOT IN • WAS NOT • CHANGED
Keywords	<p>A keyword in JQL is a word or phrase that does (or is) any of the following:</p> <ul style="list-style-type: none"> • joins two or more clauses together to form a complex JQL query • alters the logic of one or more clauses • alters the logic of operators • has an explicit definition in a JQL query • performs a specific function that alters the results of a JQL query. 	<p>Keywords reference page</p> <p>▼ Show list of keywords</p> <ul style="list-style-type: none"> • AND • OR • NOT • EMPTY • NULL • ORDER BY

Functions	<p>A function in JQL appears as a word followed by parentheses, which may contain one or more explicit values or JIRA fields.</p> <p>A function performs a calculation on either specific JIRA data or the function's content in parentheses, such that only true results are retrieved by the function, and then again by the clause in which the function is used.</p>	<p>Functions reference page</p> <p>▼ Show list of functions</p> <ul style="list-style-type: none"> • approved() • approver() • cascadeOption() • closedSprints() • componentsLeadByUser() • currentLogin() • currentUser() • earliestUnreleasedVersion() • endOfDay() • endOfMonth() • endOfWeek() • endOfYear() • issueHistory() • issuesWithRemoteLinksByGlobalId() • lastLogin() • latestReleasedVersion() • linkedIssues() • membersOf() • myApproval() • myPending() • now() • openSprints() • pending() • pendingBy() • projectsLeadByUser() • projectsWhereUserHasPermission() • projectsWhereUserHasRole() • releasedVersions() • standardIssueTypes() • startOfDay() • startOfMonth() • startOfWeek() • startOfYear() • subtaskIssueTypes() • unreleasedVersions() • votedIssues() • watchedIssues()
------------------	--	---

Running a saved search

Saved searches (also known as [Saving your search as a filter](#)) are shown in the left panel, when using advanced search. If the left panel is not showing, hover your mouse over the left side of the screen to display it.

To run a filter, e.g. **My Open Issues**, simply click it. The JQL for the advanced search will be set, and the search results will be displayed.



Next steps

Read the following related topics:

- Searching for issues
- Basic searching
- Search syntax for text fields
- JQL: The most flexible way to search JIRA (on the Atlassian blog)
- Saving your search as a filter
- Working with search results— find out how to use the issue navigator, export your search results, bulk modify issues, and share your search results.

Advanced searching - fields reference

This page describes information about fields that are used for advanced searching. A field in JQL is a word that represents a JIRA field (or a custom field that has already been defined in your JIRA applications). In a clause, a field is followed by an [operator](#), which in turn is followed by one or more values (or [functions](#)). The operator compares the value of the field with one or more values or functions on the right, such that only true results are retrieved by the clause. Note: it is not possible to compare two fields in JQL.

Affected version

Search for issues that are assigned to a particular affects version(s). You can search by version name or version ID (i.e. the number that JIRA automatically allocates to a version). Note, it is better to search by version ID than by version name. Different projects may have versions with the same name. It is also possible for your JIRA administrator to change the name of a version, which could break any saved filters that rely on that name. Version IDs, however, are unique and cannot be changed.

Syntax	<code>affectedVersion</code>
Field Type	<code>VERSION</code>
Auto-complete	Yes
Supported operators	<code>=, !=, >, >=, <, <=</code> IS, IS NOT, IN, NOT IN <i>Note that the comparison operators (e.g. ">") use the version order that has been set up by your project administrator, not a numeric or alphabetic order.</i>
Unsupported operators	<code>~, !~</code> WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none"> • <code>releasedVersions()</code> • <code>latestReleasedVersion()</code> • <code>unreleasedVersions()</code> • <code>earliestUnreleasedVersion()</code>

List of Fields:

- Affected version
- Approvals
- Assignee
- Attachments
- Category
- Comment
- Component
- Created
- Creator
- Custom field
- Customer Request Type
- Description
- Due
- Environment
- Epic link
- Filter
- Fix version
- Issue key
- Labels
- Last viewed
- Level
- Original estimate
- Parent
- Priority
- Project
- Remaining estimate
- Reporter
- Request channel type
- Request last activity time
- Resolution
- Resolved
- Sprint
- Status
- Summary
- Text
- Time spent
- Type
- Updated
- Voter
- Votes
- Watcher
- Watchers
- Work ratio

Examples

- Find issues with an AffectedVersion of 3.14:
`affectedVersion = "3.14"`
Note that full-stops are reserved characters and need to be surrounded by quote-marks.
- Find issues with an AffectedVersion of "Big Ted":
`affectedVersion = "Big Ted"`
- Find issues with an AffectedVersion ID of 10350:
`affectedVersion = 10350`

[^ top of page](#)**Approvals**

Only applicable if JIRA Service Desk is installed and licensed, and you're using the Approvals functionality.

Search for issues that have been approved or require approval. This can be further refined by user.

Syntax	approvals
Field Type	USER
Auto-complete	No
Supported operators	=
Unsupported operators	<code>~, !=, !~, >, >=, <, <=</code> <code>IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN</code> <code>, CHANGED</code>
Supported functions	<ul style="list-style-type: none"> • <code>approved()</code> • <code>approver()</code> • <code>myApproval()</code> • <code>myPending()</code> • <code>pending()</code> • <code>pendingBy()</code>
Examples	<ul style="list-style-type: none"> • Find issues that require or required approval by John Smith: <code>approval = approver(jsmith)</code> • Find issues that require approval by John Smith: <code>approval = pendingBy(jsmith)</code> • Find issues that require approval by the current user: <code>approval = myPending()</code> • Find all issues that require approval: <code>approval = pending()</code>

[^ top of page](#)**Assignee**

Search for issues that are assigned to a particular user. You can search by the user's full name, ID, or email address.

Syntax	assignee
Alias	cf[CustomFieldID]
Field Type	USER
Auto-complete	Yes
Supported operators	=, != IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED <i>Note that the comparison operators (e.g. ">") use the version order that has been set up by your project administrator, not a numeric or alphabetic order.</i>
Unsupported operators	~, !~, >, >=, <, <=
Supported functions	When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none">• membersOf() When used with the EQUALS and NOT EQUALS operators, this field supports: <ul style="list-style-type: none">• currentUser()
Examples	<ul style="list-style-type: none">• Find issues that are assigned to John Smith: assignee = "John Smith" or assignee = jsmith• Find issues that are currently assigned, or were previously assigned, to John Smith: assignee WAS "John Smith" or assignee WAS jsmith• Find issues that are assigned by the user with email address "bob@mycompany.com": assignee = "bob@mycompany.com" <p><i>Note that full-stops and "@" symbols are reserved characters and need to be surrounded by quote-marks.</i></p>

[^ top of page](#)

Attachments

Search for issues that have or do not have attachments.

Syntax	attachments
Field Type	ATTACHMENT
Auto-complete	Yes
Supported operators	IS, IS NOT
Unsupported operators	=, !=, ~, !~, >, >=, <, <= IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	None

Examples	<ul style="list-style-type: none"> Search for issues that have attachments: attachments IS NOT EMPTY Search for issues that do not have attachments: attachments IS EMPTY
-----------------	---

[^ top of page](#)

Category

Search for issues that belong to projects in a particular category.

Syntax	category
Field Type	CATEGORY
Auto-complete	Yes
Supported operators	=, != IS, IS NOT, IN, NOT IN
Unsupported operators	~, !~, >, >=, <, <= WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues that belong to projects in the "Alphabet Projects" Category: category = "Alphabet Projects"

[^ top of page](#)

Comment

Search for issues that have a comment that contains particular text. [JIRA text-search syntax](#) can be used.

Syntax	comment
Field Type	TEXT
Auto-complete	No
Supported operators	~, !~
Unsupported operators	=, !=, >, >=, <, <= IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where a comment contains text that matches "My PC is quite old" (i.e. a "fuzzy" match): comment ~ "My PC is quite old" Find issues where a comment contains the exact phrase "My PC is quite old": comment ~ "\\"My PC is quite old\\\""

[^ top of page](#)

Component

Search for issues that belong to a particular component(s) of a project. You can search by component name or component ID (i.e. the number that JIRA automatically allocates to a component).

Note, it is safer to *search by component ID than by component name*. Different projects may have components with the same name, so searching by component name may return issues from multiple projects. It is also possible for your JIRA administrator to change the name of a component, which could break any saved filters that rely on that name. Component IDs, however, are unique and cannot be changed.

Syntax	component
Field Type	COMPONENT
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN
Unsupported operators	~, !~, >, >=, <, <= WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	When used with the IN and NOT IN operators, component supports: <ul style="list-style-type: none"> componentsLeadByUser()
Examples	<ul style="list-style-type: none"> Find issues in the "Comp1" or "Comp2" component: component in (Comp1, Comp2) Find issues in the "Comp1" and "Comp2" components: component in (Comp1) and component in (Comp2) or component = Comp1 and component = Comp2 Find issues in the component with ID 20500: component = 20500

[^ top of page](#)

Created

Search for issues that were created on, before, or after a particular date (or date range). Note that if a time-component is not specified, midnight will be assumed. Please note that the search results will be relative to your configured time zone (which is by default the JIRA server's time zone).

Use one of the following formats:

```
"YYYY/MM/dd HH:mm"  
"YYYY-MM-dd HH:mm"  
"YYYY/MM/dd"  
"YYYY-MM-dd"
```

Or use "w" (weeks), "d" (days), "h" (hours) or "m" (minutes) to specify a date relative to the current time. The default is "m" (minutes). Be sure to use quote-marks ("); if you omit the quote-marks, the number you supply will be interpreted as milliseconds after epoch (1970-1-1).

Syntax	created
Alias	createdDate
Field Type	DATE

Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	When used with the EQUALS , NOT EQUALS , GREATER THAN , GREATER THAN EQUALS , LESS THAN or LESS THAN EQUALS operators, this field supports: <ul style="list-style-type: none">• currentLogin()• lastLogin()• now()• startOfDay()• startOfWeek()• startOfMonth()• startOfYear()• endOfDay()• endOfWeek()• endOfMonth()• endOfYear()
Examples	<ul style="list-style-type: none">• Find all issues created before 12th December 2010: <code>created < "2010/12/12"</code>• Find all issues created on or before 12th December 2010: <code>created <= "2010/12/13"</code>• Find all issues created on 12th December 2010 before 2:00pm: <code>created > "2010/12/12" and created < "2010/12/12 14:00"</code>• Find issues created less than one day ago: <code>created > "-1d"</code>• Find issues created in January 2011: <code>created > "2011/01/01" and created < "2011/02/01"</code>• Find issues created on 15 January 2011: <code>created > "2011/01/15" and created < "2011/01/16"</code>

[^ top of page](#)

Creator

Search for issues that were created by a particular user. You can search by the user's full name, ID, or email address.

Syntax	creator
Field Type	USER
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN
Unsupported operators	~ , !~ , > , >= , < , <= CHANGED

Supported functions	<p>When used with the IN and NOT IN operators, this field supports:</p> <ul style="list-style-type: none"> • <code>membersOf()</code> <p>When used with the EQUALS and NOT EQUALS operators, this field supports:</p> <ul style="list-style-type: none"> • <code>currentUser()</code>
Examples	<ul style="list-style-type: none"> • Search for issues that were created by Jill Jones: <code>creator = "Jill Jones"</code> or <code>creator = "jjones"</code> • Search for issues that were created by the user with email address "bob@mycompany.com": <code>creator = "bob@mycompany.com"</code> <small>(Note that full-stops and "@" symbols are reserved characters, so the email address needs to be surrounded by quote-marks.)</small>

[^ top of page](#)

Custom field

Only applicable if your JIRA administrator has created one or more custom fields.

Search for issues where a particular custom field has a particular value. You can search by custom field name or custom field ID (i.e. the number that JIRA automatically allocates to an custom field).

Note, it is safer to search by custom field ID than by custom field name. It is possible for a custom field to have the same name as a built-in JIRA system field; in which case, JIRA will search for the system field (not your custom field). It is also possible for your JIRA administrator to change the name of a custom field, which could break any saved filters that rely on that name. Custom field IDs, however, are unique and cannot be changed.

Syntax	<code>CustomFieldName</code>
Alias	<code>cf[CustomFieldID]</code>
Field Type	<i>Depends on the custom field's configuration</i> <small>Note, JIRA text-search syntax can be used with custom fields of type 'Text'.</small>
Auto-complete	Yes, for custom fields of type picker, group picker, select, checkbox and radio button fields
Supported operators	<i>Different types of custom field support different operators.</i>
Supported operators: number and date fields	<code>= , != , > , >= , < . <=</code> <code>IS , IS NOT , IN , NOT IN</code>
Unsupported operators: number and date fields	<code>~ , !~</code> <code>WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>

Supported operators: picker, select, checkbox and radio button fields	= , != IS , IS NOT , IN , NOT IN
Unsupported operators: picker, select, checkbox and radio button fields	~, !~, > , >= , < . <= WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported operators: text fields	~, !~ IS , IS NOT
Unsupported operators: text fields	= , != , > , >= , < . <= IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Unsupported operators	~, !~, > , >= , < , <= WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	<i>Different types of custom fields support different functions.</i>
Supported functions: date/time fields	When used with the EQUALS , NOT EQUALS , GREATER THAN , GREATER THAN EQUALS , LESS THAN or LESS THAN EQUALS operators, this field supports: <ul style="list-style-type: none"> • currentLogin() • lastLogin() • now() • startOfDay() • startOfWeek() • startOfMonth() • startOfYear() • endOfDay() • endOfWeek() • endOfMonth() • endOfYear()
Supported functions: version picker fields	Version picker fields: When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none"> • releasedVersions() • latestReleasedVersion() • unreleasedVersions() • earliestUnreleasedVersion()
Examples	<ul style="list-style-type: none"> • Find issues where the value of the "Location" custom field is "New York": <code>location = "New York"</code> • Find issues where the value of the custom field with ID 10003 is "New York": <code>cf[10003] = "New York"</code> • Find issues where the value of the "Location" custom field is "London" or "Milan" or "Paris": <code>cf[10003] in ("London" , "Milan" , "Paris")</code> • Find issues where the "Location" custom field has no value: <code>location != empty</code>

[^ top of page](#)

Customer Request Type

Only applicable if JIRA Service Desk is installed and licensed.

Search for Issues matching a specific Customer Request Type in a service desk project. You can search for a Customer Request Type either by name or description as configured in the Request Type configuration screen.

Syntax	"Customer Request Type"
Field Type	Custom field
Auto-complete	Yes
Supported operators	= , != IN , NOT IN
Unsupported operators	~, !~, >, >=, <, <= IS , IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
	<p>Note that the Lucene value for Customer Request Type, is portal-key/request-type-key. While the portal key cannot be changed after a service desk portal is created, the project key can be changed. The Request Type key cannot be changed once the Request Type is created.</p>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where Customer Request Type is Request a new account in projects that the user has access to: "Customer Request Type" = "Request a new account" Find issues where the Customer Request Type is Request a new account in SimpleDesk project, where the right operand is a selected Lucene value from the auto-complete suggestion list. "Customer Request Type" = "sd/system-access" Find issues where Customer Request Type is either Request a new account or Get IT Help. "Customer Request Type" IN ("Request a new account", "Get IT Help")

[^ top of page](#)

Description

Search for issues where the description contains particular text. JIRA text-search syntax can be used.

Syntax	description
Field Type	TEXT
Auto-complete	No
Supported operators	~, !~ IS , IS NOT
Unsupported operators	= , != , > , >= , < , <= IN , NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED

Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where the description contains text that matches "Please see screenshot" (i.e. a "fuzzy" match): description ~ "Please see screenshot" Find issues where the description contains the exact phrase "Please see screenshot": description ~ "\"Please see screenshot\""

[^ top of page](#)

Due

Search for issues that were due on, before, or after a particular date (or date range). Note that the due date relates to the *date* only (not to the time).

Use one of the following formats:

"YYYY/MM/dd"
"YYYY-MM-dd"

Or use "w" (weeks) or "d" (days) to specify a date relative to the current date. Be sure to use quote-marks ("").

Syntax	due
Alias	dueDate
Field Type	DATE
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	<p>When used with the EQUALS, NOT EQUALS, GREATER THAN, GREATER THAN EQUALS, LESS THAN or LESS THAN EQUALS operators, this field supports:</p> <ul style="list-style-type: none"> currentLogin() lastLogin() now() startOfDay() startOfWeek() startOfMonth() startOfYear() endOfDay() endOfWeek() endOfMonth() endOfYear()

Examples	<ul style="list-style-type: none"> Find all issues due before 31st December 2010: due < "2010/12/31" Find all issues due on or before 31st December 2010: due <= "2011/01/01" Find all issues due tomorrow: due = "1d" Find all issues due in January 2011: due >= "2011/01/01" and due <= "2011/01/31" Find all issues due on 15 January 2011: due = "2011/01/15"
-----------------	---

[^ top of page](#)

Environment

Search for issues where the environment contains particular text. [JIRA text-search syntax](#) can be used.

Syntax	environment
Field Type	TEXT
Auto-complete	No
Supported operators	<code>~ , !~</code> <code>IS , IS NOT</code>
Unsupported operators	<code>= , != , > , >= , < , <=</code> <code>IN , NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where the environment contains text that matches "Third floor" (i.e. a "fuzzy" match): environment ~ "Third floor" Find issues where the environment contains the exact phrase "Third floor": environment ~ "\\"Third floor\\\""

[^ top of page](#)

Epic link

Search for issues that belong to a particular epic. The search is based on either the epic's name, issue key, or issue ID (i.e. the number that JIRA automatically allocates to an issue).

Syntax	"epic link"
Field Type	Epic Link Relationship
Auto-complete	No
Supported operators	<code>= , !=</code> <code>IS , IS NOT, IN , NOT IN</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <=</code> <code>WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED</code>

Supported functions	When used with the IN or NOT IN operators, epic link supports:
	<ul style="list-style-type: none"> • <code>issueHistory()</code> • <code>linkedIssues()</code> • <code>votedIssues()</code> • <code>watchedIssues()</code>
Examples	<ul style="list-style-type: none"> • Find issues that belong to epic "Jupiter", where "Jupiter has the issue key ANERDS-31: <code>"epic link" = ANERDS-31</code> or <code>"epic link" = Jupiter</code>

[^ top of page](#)

Filter

You can use a saved filter to narrow your search. You can search by filter name or filter ID (i.e. the number that JIRA automatically allocates to a saved filter).

Note:

- It is safer to search by filter ID than by filter name. It is possible for a filter name to be changed, which could break a saved filter that invokes another filter by name. Filter IDs, however, are unique and cannot be changed.
- An unnamed link statement in your typed query will override an ORDER BY statement in the saved filter.
- You cannot run or save a filter that would cause an infinite loop (i.e. you cannot reference a saved filter if it eventually references your current filter).

Syntax	<code>filter</code>
Aliases	<code>request , savedFilter , searchRequest</code>
Field Type	Filter
Auto-complete	Yes
Supported operators	<code>= , !=</code> <code>IN , NOT IN</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <=</code> <code>IS , IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> • Search the results of the filter "My Saved Filter" (which has an ID of 12000) for issues assigned to the user <code>jsmith</code>: <code>filter = "My Saved Filter" and assignee = jsmith</code> or <code>filter = 12000 and assignee = jsmith</code>

[^ top of page](#)

Fix version

Search for issues that are assigned to a particular fix version. You can search by version name or version ID (i.e. the number that JIRA automatically allocates to a version).

Note, it is safer to search by version ID than by version name. Different projects may have versions with the

same name, so searching by version name may return issues from multiple projects. It is also possible for your JIRA administrator to change the name of a version, which could break any saved filters that rely on that name. Version IDs, however, are unique and cannot be changed.

Syntax	<code>fixVersion</code>
Field Type	VERSION
Auto-complete	Yes
Supported operators	= , != , > , >= , < , <= IS , IS NOT, IN , NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED <i>Note that the comparison operators (e.g. ">") use the version order that has been set up by your project administrator, not a numeric or alphabetic order.</i>
Unsupported operators	~, !~
Supported functions	When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none">• <code>releasedVersions()</code>• <code>latestReleasedVersion()</code>• <code>unreleasedVersions()</code>• <code>earliestUnreleasedVersion()</code>
Examples	<ul style="list-style-type: none">• Find issues with a Fix Version of 3.14 or 4.2: <code>fixVersion in ("3.14", "4.2")</code> <i>(Note that full-stops are reserved characters, so they need to be surrounded by quote-marks.)</i>• Find issues with a Fix Version of "Little Ted": <code>fixVersion = "Little Ted"</code>• Find issues with a Fix Version ID of 10001: <code>fixVersion = 10001</code>

[^ top of page](#)

Issue key

Search for issues with a particular issue key or issue ID (i.e. the number that JIRA automatically allocates to an issue).

Syntax	<code>issueKey</code>
Aliases	<code>id , issue , key</code>
Field Type	ISSUE
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT, IN , NOT IN
Unsupported operators	~, !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED

Supported functions	When used with the IN or NOT IN operators, issueKey supports:
	<ul style="list-style-type: none"> • <code>issueHistory()</code> • <code>linkedIssues()</code> • <code>votedIssues()</code> • <code>watchedIssues()</code>
Examples	<ul style="list-style-type: none"> • Find the issue with key "ABC-123": <code>issueKey = ABC-123</code>

[^ top of page](#)

Labels

Search for issues tagged with a label or list of labels. You can also search for issues without any labels to easily identify which issues need to be tagged so they show up in the relevant sprints, queues or reports.

Syntax	<code>labels</code>
Field Type	<code>LABEL</code>
Auto-complete	Yes
Supported operators	<code>=, !=, IS, IS NOT, IN, NOT IN</code> <i>We recommend using IS or IS NOT to search for a single label, and IN or NOT IN to search for a list of labels.</i>
Unsupported operators	<code>~, !~, , , >, >=, <, <=</code> <code>WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> • Find issues with an existing label: <code>labels = "x"</code> • Find issues without a specified label, including issues without a label: <code>labels not in ("x") or labels is EMPTY</code>

Last viewed

Search for issues that were last viewed on, before, or after a particular date (or date range). Note that if a time-component is not specified, midnight will be assumed. Please note that the search results will be relative to your configured time zone (which is by default the JIRA server's time zone).

Use one of the following formats:

```
"YYYY/MM/dd HH:mm"
"YYYY-MM-dd HH:mm"
"YYYY/MM/dd"
"YYYY-MM-dd"
```

Or use "`w`" (weeks), "`d`" (days), "`h`" (hours) or "`m`" (minutes) to specify a date relative to the current time. The default is "`m`" (minutes). Be sure to use quote-marks (""); if you omit the quote-marks, the number you supply will be interpreted as milliseconds after epoch (1970-1-1).

Syntax	<code>lastViewed</code>
Field Type	<code>DATE</code>

Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	When used with the EQUALS , NOT EQUALS , GREATER THAN , GREATER THAN EQUALS , LESS THAN or LESS THAN EQUALS operators, this field supports: <ul style="list-style-type: none">• currentLogin()• lastLogin()• now()• startOfDay()• startOfWeek()• startOfMonth()• startOfYear()• endOfDay()• endOfWeek()• endOfMonth()• endOfYear()
Examples	<ul style="list-style-type: none">• Find all issues last viewed before 12th December 2010: lastViewed < "2010/12/12"• Find all issues last viewed on or before 12th December 2010: lastViewed <= "2010/12/13"• Find all issues last viewed on 12th December 2010 before 2:00pm: lastViewed > "2010/12/12" and created < "2010/12/12 14:00"• Find issues last viewed less than one day ago: lastViewed > "-1d"• Find issues last viewed in January 2011: lastViewed > "2011/01/01" and created < "2011/02/01"• Find issues last viewed on 15 January 2011: lastViewed > "2011/01/15" and created < "2011/01/16"

[^ top of page](#)

Level

Only available if issue level security has been enabled by your JIRA administrator.

Search for issues with a particular security level. You can search by issue level security name or issue level security ID (i.e. the number that JIRA automatically allocates to an issue level security).

Note, it is safer to search by security level ID than by security level name. It is possible for your JIRA administrator to change the name of a security level, which could break any saved filter that rely on that name. Security level IDs, however, are unique and cannot be changed.

Syntax	level
Field Type	SECURITY LEVEL
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN
Unsupported operators	> , >= , < , <= , ~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED

Supported functions	None
Examples	<ul style="list-style-type: none"> Search for issues with a security level of "Really High" or "level1": level in ("Really High", level1) Search for issues with a security level ID of 123: level = 123

[^ top of page](#)

Original estimate

Only available if time-tracking has been enabled by your JIRA administrator.

Search for issues where the original estimate is set to a particular value (i.e. a number, not a date or date range). Use "w", "d", "h" and "m" to specify weeks, days, hours, or minutes.

Syntax	originalEstimate
Alias	timeOriginalEstimate
Field Type	DURATION
Auto-complete	No
Supported operators	=, !=, >, >=, <, <=, IS, IS NOT, IN, NOT IN
Unsupported operators	~, !~ WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with an original estimate of 1 hour: originalEstimate = 1h Find issues with an original estimate of more than 2 days: originalEstimate > 2d

[^ top of page](#)

Parent

Only available if sub-tasks have been enabled by your JIRA administrator.

Search for all sub-tasks of a particular issue. You can search by issue key or by issue ID (i.e. the number that JIRA automatically allocates to an Issue).

Syntax	parent
Field Type	ISSUE
Auto-complete	No
Supported operators	=, != IN, NOT IN
Unsupported operators	>, >=, <, <=, ~, !~ IS, IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	None

Examples

- Find issues that are sub-tasks of issue TEST-1234:
parent = TEST-1234

[^ top of page](#)

Priority

Search for issues with a particular priority. You can search by priority name or priority ID (i.e. the number that JIRA automatically allocates to a priority).

Note, it is safer to search by priority ID than by priority name. It is possible for your JIRA administrator to change the name of a priority, which could break any saved filter that rely on that name. Priority IDs, however, are unique and cannot be changed.

Syntax	priority
Field Type	PRIORITY
Auto-complete	Yes
Supported operators	= , != , > , >= , < , <= IS , IS NOT, IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Unsupported operators	~ , !~
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with a priority of "High": priority = High Find issues with a priority ID of 10000: priority = 10000

[^ top of page](#)

Project

Search for issues that belong to a particular project. You can search by project name, by project key or by project ID (i.e. the number that JIRA automatically allocates to a project). In the rare case where there is a project whose project key is the same as another project's name, then the project key takes preference and hides results from the second project.

Syntax	project
Field Type	PROJECT
Auto-complete	Yes
Supported operators	= , != IS , IS NOT, IN , NOT IN
Unsupported operators	> , >= , < , <= , ~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED

Supported functions	When used with the IN and NOT IN operators, project supports:
	<ul style="list-style-type: none"> • <code>projectsLeadByUser()</code> • <code>projectsWhereUserHasPermission()</code> • <code>projectsWhereUserHasRole()</code>
Examples	<ul style="list-style-type: none"> • Find issues that belong to the Project that has the name "ABC Project": <code>project = "ABC Project"</code> • Find issues that belong to the project that has the key "ABC": <code>project = "ABC"</code> • Find issues that belong to the project that has the ID "1234": <code>project = 1234</code>

[^ top of page](#)

Remaining estimate

Only available if time-tracking has been enabled by your JIRA administrator.

Search for issues where the remaining estimate is set to a particular value (i.e. a number, not a date or date range). Use "w", "d", "h" and "m" to specify weeks, days, hours, or minutes.

Syntax	<code>remainingEstimate</code>
Alias	<code>timeEstimate</code>
Field Type	DURATION
Auto-complete	No
Supported operators	= , != , > , >= , < , <= , IS , IS NOT , IN , NOT IN
Unsupported operators	~, !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> • Find issues with a remaining estimate of more than 4 hours: <code>remainingEstimate > 4h</code>

[^ top of page](#)

Reporter

Search for issues that were reported by a particular user. This may be the same as the creator, but can be distinct. You can search by the user's full name, ID, or email address.

Syntax	<code>reporter</code>
Field Type	USER
Auto-complete	Yes
Supported operators	= , != , IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED

Unsupported operators	<code>~ , !~ , > , >= , < , <=</code>
Supported functions	<p>When used with the IN and NOT IN operators, this field supports:</p> <ul style="list-style-type: none"> • <code>membersOf()</code> <p>When used with the EQUALS and NOT EQUALS operators, this field supports:</p> <ul style="list-style-type: none"> • <code>currentUser()</code>
Examples	<ul style="list-style-type: none"> • Search for issues that were reported by Jill Jones: <code>reporter = "Jill Jones"</code> or <code>reporter = jjones</code> • Search for issues that were reported by the user with email address "bob@mycompany.com": <code>reporter = "bob@mycompany.com"</code> <small>(Note that full-stops and "@" symbols are reserved <i>characters</i>, so the email address needs to be surrounded by quote-marks.)</small>

[^ top of page](#)

Request channel type

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that were requested through a specific channel (e.g. issues submitted via email or through a Service Desk portal).

Syntax	<code>request-channel-type</code>
Field Type	TEXT
Auto-complete	Yes
Supported operators	<code>= , !=</code> <code>IS, IS NOT, IN, NOT IN</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <=</code> <code>WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	<p>When used with the IN and NOT IN operators, this field supports:</p> <ul style="list-style-type: none"> • <code>email</code>: requests submitted via email • <code>jira</code>: requests created using JIRA • <code>portal</code>: requests created using a Service Desk portal • <code>api</code>: requests created using a REST API
Examples	<ul style="list-style-type: none"> • Find issues where the request channel was email: <code>request-channel-type = email</code> • Find issues where the request channel was something other than a service desk portal: <code>request-channel-type != portal</code>

[^ top of page](#)

Request last activity time

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that were created on, before, or after a particular date (or date range). Note that if a time-component is not specified, midnight will be assumed. Please note that the search results will be relative to your configured time zone (which is by default the JIRA server's time zone).

Use one of the following formats:

```
"YYYY/MM/dd HH:mm"
"YYYY-MM-dd HH:mm"
"YYYY/MM/dd"
"YYYY-MM-dd"
```

Or use "w" (weeks), "d" (days), "h" (hours) or "m" (minutes) to specify a date relative to the current time. The default is "m" (minutes). Be sure to use quote-marks (""); if you omit the quote-marks, the number you supply will be interpreted as milliseconds after epoch (1970-1-1).

Syntax	request-last-activity-time
Field Type	DATE
Auto-complete	Yes
Supported operators	=, !=, >, >=, <, <= IS, IS NOT, IN, NOT IN
Unsupported operators	~, !~ WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	When used with the EQUALS, NOT EQUALS, GREATER THAN, GREATER THAN EQUALS, LESS THAN or LESS THAN EQUALS operators, this field supports: <ul style="list-style-type: none"> • currentLogin() • lastLogin() • now() • startOfDay() • startOfWeek() • startOfMonth() • startOfYear() • endOfDay() • endOfWeek() • endOfMonth() • endOfYear()
Examples	<ul style="list-style-type: none"> • Find all issues last acted on before 23rd May 2016: request-last-activity-time < "2016/05/23" • Find all issues last acted on or before 23rd May 2016: request-last-activity-time <= "2016/05/23" • Find all issues created on 23rd May 2016 and last acted on before 2:00pm that day: created > "2016/05/23" AND request-last-activity-time < "2016/05/23 14:00" • Find issues last acted on less than one day ago: request-last-activity-time > "-1d" • Find issues last acted on in January 2016: request-last-activity-time > "2016/01/01" and request-last-activity-time < "2016/02/01"

[^ top of page](#)

Resolution

Search for issues that have a particular resolution. You can search by resolution name or resolution ID (i.e. the number that JIRA automatically allocates to a resolution).

Note, it is safer to search by resolution ID than by resolution name. It is possible for your JIRA administrator to change the name of a resolution, which could break any saved filter that rely on that name. Resolution IDs, however, are unique and cannot be changed.

Syntax	resolution
Field Type	RESOLUTION
Auto-complete	Yes
Supported operators	= , != , > , >= , < , <= IS , IS NOT, IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Unsupported operators	~ , !~
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with a resolution of "Cannot Reproduce" or "Won't Fix": resolution in ("Cannot Reproduce", "Won't Fix") Find issues with a resolution ID of 5: resolution = 5 Find issues that do not have a resolution: resolution = unresolved

[^ top of page](#)

Resolved

Search for issues that were resolved on, before, or after a particular date (or date range). Note that if a time-component is not specified, midnight will be assumed. Please note that the search results will be relative to your configured time zone (which is by default the JIRA server's time zone).

Use one of the following formats:

```
"YYYY/MM/dd HH:mm"
"YYYY-MM-dd HH:mm"
"YYYY/MM/dd"
"YYYY-MM-dd"
```

Or use "w" (weeks), "d" (days), "h" (hours) or "m" (minutes) to specify a date relative to the current time. The default is "m" (minutes). Be sure to use quote-marks ("); if you omit the quote-marks, the number you supply will be interpreted as milliseconds after epoch (1970-1-1).

Syntax	resolved
Alias	resolutionDate
Field Type	DATE
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT, IN , NOT IN

Unsupported operators	<code>~, !~</code> <code>WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	When used with the EQUALS, NOT EQUALS, GREATER THAN, GREATER THAN EQUALS, LESS THAN or LESS THAN EQUALS operators, this field supports: <ul style="list-style-type: none"> • <code>currentLogin()</code> • <code>lastLogin()</code> • <code>now()</code> • <code>startOfDay()</code> • <code>startOfWeek()</code> • <code>startOfMonth()</code> • <code>startOfYear()</code> • <code>endOfDay()</code> • <code>endOfWeek()</code> • <code>endOfMonth()</code> • <code>endOfYear()</code>
Examples	<ul style="list-style-type: none"> • Find all issues that were resolved before 31st December 2010: <code>resolved <= "2010/12/31"</code> • Find all issues that were resolved before 2.00pm on 31st December 2010: <code>resolved < "2010/12/31 14:00"</code> • Find all issues that were resolved on or before 31st December 2010: <code>resolved <= "2011/01/01"</code> • Find issues that were resolved in January 2011: <code>resolved > "2011/01/01" and resolved < "2011/02/01"</code> • Find issues that were resolved on 15 January 2011: <code>resolved > "2011/01/15" and resolved < "2011/01/16"</code> • Find issues that were resolved in the last hour: <code>resolved > -1h</code>

[^ top of page](#)

Sprint

Search for issues that are assigned to a particular sprint. This works for active sprints and future sprints. The search is based on either the sprint name or the sprint ID (i.e. the number that JIRA automatically allocates to a sprint).

If you have multiple sprints with similar (or identical) names, you can simply search by using the sprint name — or even just part of it. The possible matches will be shown in the autocomplete drop-down, with the sprint dates shown to help you distinguish between them. (The sprint ID will also be shown, in brackets).

Syntax	<code>sprint</code>
Field Type	NUMBER
Auto-complete	Yes
Supported operators	<code>=, !=</code> <code>IS, IS NOT, IN, NOT IN</code>
Unsupported operators	<code>~, !~, >, >=, <, <=</code> <code>WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	<ul style="list-style-type: none"> • <code>openSprints()</code> • <code>closedSprints()</code>

Examples	<ul style="list-style-type: none"> Find issues that belong to sprint 999: sprint = 999 Find issues that belong to sprint "February 1": sprint = "February 1" Find issues that belong to either "February 1", "February 2" or "February 3": sprint in ("February 1", "February 2", "February 3") Find issues that are assigned to a sprint: sprint is not empty
-----------------	--

[^ top of page](#)

Status

Search for issues that have a particular status. You can search by status name or status ID (i.e. the number that JIRA automatically allocates to a status).

Note:

- It is safer to search by status ID than status name. It is possible for your JIRA administrator to change the name of a status, which could break any saved filter that rely on that name. Status IDs, however, are unique and cannot be changed.
- The WAS, WAS NOT, WAS IN and WAS NOT IN operators can only be used with the name, not the ID.

Syntax	status
Field Type	STATUS
Auto-complete	Yes
Supported operators	= , != IS , IS NOT, IN , NOT IN , WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Unsupported operators	~, !~, > , >= , < , <=
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with a status of "Open": status = Open Find issues with a status ID of 1: status = 1 Find issues that currently have, or previously had, a status of "Open": status WAS Open

[^ top of page](#)

Summary

Search for issues where the summary contains particular text. [JIRA text-search syntax](#) can be used.

Syntax	summary
Field Type	TEXT
Auto-complete	No

Supported operators	<code>~, !~</code> <code>IS, IS NOT</code>
Unsupported operators	<code>=, !=, >, >=, <, <=</code> <code>IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where the summary contains text that matches "Error saving file" (i.e. a "fuzzy" match): <code>summary ~ "Error saving file"</code> Find issues where the summary contains the exact phrase "Error saving file": <code>summary ~ "\\" Error saving file \\ "</code>

[^ top of page](#)

Text

This is a "master-field" that allows you to search all text fields, i.e.:

- Summary
- Description
- Environment
- Comments
- custom fields that use the "free text searcher"; this includes custom fields of the following built-in custom field types:
 - Free text field (unlimited text)
 - Text field (< 255 characters)
 - Read-only text field

Notes:

- The `text` master-field can only be used with the CONTAINS operator ("~" and "! ~").
- [JIRA text-search syntax](#) can be used with these fields.

Syntax	<code>text</code>
Field Type	TEXT
Auto-complete	No
Supported operators	<code>~</code>
Unsupported operators	<code>=, !=, !~, >, >=, <, <=</code> <code>IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where a text field matches the word "Fred": <code>text ~ "Fred"</code> or <code>text ~ Fred</code> Find all issues where a text field contains the exact phrase "full screen": <code>text ~ "\\" full screen \\ "</code>

[^ top of page](#)

Time spent

Only available if time-tracking has been enabled by your JIRA administrator.

Search for issues where the time spent is set to a particular value (i.e. a number, not a date or date range). Use "w", "d", "h" and "m" to specify weeks, days, hours, or minutes.

Syntax	timeSpent
Field Type	DURATION
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues where the time spent is more than 5 days: <code>timeSpent > 5d</code>

[^ top of page](#)

Type

Search for issues that have a particular issue type. You can search by issue type name or issue type ID (i.e. the number that JIRA automatically allocates to an issue type).

Note, it is safer to search by type ID than type name. It is possible for your JIRA administrator to change the name of a type, which could break any saved filter that rely on that name. Type IDs, however, are unique and cannot be changed.

Syntax	type
Alias	issueType
Field Type	ISSUE_TYPE
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ , > , >= , < , <= WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues with an issue type of "Bug": <code>type = Bug</code> Find issues with an issue type of "Bug" or "Improvement": <code>issueType in (Bug, Improvement)</code> Find issues with an issue type ID of 2: <code>issueType = 2</code>

[^ top of page](#)

Updated

Search for issues that were last updated on, before, or after a particular date (or date range). Note that if a time-component is not specified, midnight will be assumed. Please note that the search results will be relative to your configured time zone (which is by default the JIRA server's time zone).

Use one of the following formats:

```
"YYYY/MM/dd HH:mm"
"YYYY-MM-dd HH:mm"
"YYYY/MM/dd"
"YYYY-MM-dd"
```

Or use "w" (weeks), "d" (days), "h" (hours) or "m" (minutes) to specify a date relative to the current time. The default is "m" (minutes). Be sure to use quote-marks ("); if you omit the quote-marks, the number you supply will be interpreted as milliseconds after epoch (1970-1-1).

Syntax	updated
Alias	updatedDate
Field Type	DATE
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	When used with the EQUALS , NOT EQUALS , GREATER THAN , GREATER THAN EQUALS , LESS THAN or LESS THAN EQUALS operators, this field supports: <ul style="list-style-type: none"> • currentLogin() • lastLogin() • now() • startOfDay() • startOfWeek() • startOfMonth() • startOfYear() • endOfDay() • endOfWeek() • endOfMonth() • endOfYear()
Examples	<ul style="list-style-type: none"> • Find issues that were last updated before 12th December 2010: updated < "2010/12/12" • Find issues that were last updated on or before 12th December 2010: updated < "2010/12/13" • Find all issues that were last updated before 2.00pm on 31st December 2010: updated < "2010/12/31 14:00" • Find issues that were last updated more than two weeks ago: updated < "-2w" • Find issues that were last updated on 15 January 2011: updated > "2011/01/15" and updated < "2011/01/16" • Find issues that were last updated in January 2011: updated > "2011/01/01" and updated < "2011/02/01"

[^ top of page](#)

Voter

Search for issues for which a particular user has voted. You can search by the user's full name, ID, or email address. Note that you can only find issues for which you have the "View Voters and Watchers" permission, unless you are searching for your own votes. See also [votedIssues](#).

Syntax	voter
Field Type	USER
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ , > , >= , < , <= WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none">• membersOf() When used with the EQUALS and NOT EQUALS operators, this field supports: <ul style="list-style-type: none">• currentUser()
Examples	<ul style="list-style-type: none">• Search for issues that you have voted for: <code>voter = currentUser()</code>• Search for issues that the user "jsmith" has voted for: <code>voter = "jsmith"</code>• Search for issues for which a member of the group "jira-administrators" has voted: <code>voter in membersOf("jira-administrators")</code>

[^ top of page](#)

Votes

Search for issues with a specified number of votes.

Syntax	votes
Field Type	NUMBER
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IN , NOT IN
Unsupported operators	~ , !~ IS , IS NOT , WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none">• Find all issues that have 12 or more votes: <code>votes >= 12</code>

[^ top of page](#)

Watcher

Search for issues that a particular user is watching. You can search by the user's full name, ID, or email address. Note that you can only find issues for which you have the "View Voters and Watchers" permission, unless you are searching for issues where you are the watcher. See also [watchedIssues](#).

Syntax	watcher
Field Type	USER
Auto-complete	Yes
Supported operators	= , != IS , IS NOT , IN , NOT IN
Unsupported operators	~ , !~ , > , >= , < , <= WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none">• membersOf() When used with the EQUALS and NOT EQUALS operators, this field supports: <ul style="list-style-type: none">• currentUser()
Examples	<ul style="list-style-type: none">• Search for issues that you are watching: <code>watcher = currentUser()</code>• Search for issues that the user "jsmith" is watching: <code>watcher = "jsmith"</code>• Search for issues that are being watched by a member of the group "jira-administrators": <code>watcher in membersOf("jira-administrators")</code>

[^ top of page](#)

Watchers

Search for issues with a specified number of watchers.

Syntax	watchers
Field Type	NUMBER
Auto-complete	No
Supported operators	= , != , > , >= , < , <= IN , NOT IN
Unsupported operators	~ , !~ IS , IS NOT , WAS, WAS IN, WAS NOT, WAS NOT IN , CHANGED
Supported functions	When used with the IN and NOT IN operators, this field supports: <ul style="list-style-type: none">• membersOf() When used with the EQUALS and NOT EQUALS operators, this field supports: <ul style="list-style-type: none">• currentUser()

Examples	<ul style="list-style-type: none"> Find all issues that are being watched by more than 3 people: watchers > 3
-----------------	---

[^ top of page](#)

Work ratio

Only available if time-tracking has been enabled by your JIRA administrator.

Search for issues where the work ratio has a particular value. Work ratio is calculated as follows: **workRatio = timeSpent / originalEstimate) x 100**

Syntax	workRatio
Field Type	NUMBER
Auto-complete	No
Supported operators	=, !=, >, >=, <, <=, IS, IS NOT, IN, NOT IN
Unsupported operators	~, !~ WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Supported functions	None
Examples	<ul style="list-style-type: none"> Find issues on which more than 75% of the original estimate has been spent: workRatio > 75

[^ top of page](#)

Advanced searching - keywords reference

This page describes information about keywords that are used for advanced searching. See [Advanced searching](#).

List of Keywords:

- AND
- OR
- NOT
- EMPTY
- NULL
- ORDER BY

A keyword in JQL is a word or phrase that does (or is) any of the following:

- joins two or more clauses together to form a complex JQL query
- alters the logic of one or more clauses
- alters the logic of operators
- has an explicit definition in a JQL query
- performs a specific function that alters the results of a JQL query

AND

Used to combine multiple clauses, allowing you to refine your search.

Note that you can use parentheses to control the order in which clauses are executed.

Examples

- Find all open issues in the "New office" project:

```
project = "New office" and status = "open"
```

- Find all open, urgent issues that are assigned to jsmith:

```
status = open and priority = urgent and assignee = jsmith
```

- Find all issues in a particular project that are not assigned to jsmith:

```
project = JRA and assignee != jsmith
```

- Find all issues for a specific release which consists of different version numbers across several projects:

```
project in (JRA,CONF) and fixVersion = "3.14"
```

- Find all issues where neither the Reporter nor the Assignee is Jack, Jill or John:

```
reporter not in (Jack,Jill,John) and assignee not in  
(Jack,Jill,John)
```

[^ top of page](#)

OR

Used to combine multiple clauses, allowing you to expand your search.

Note that you can use parentheses to control the order in which clauses are executed.

(Note: also see [IN](#), which can be a more convenient way to search for multiple values of a field.)
Examples

- Find all issues that were created by either jsmith or jbrown:

```
reporter = jsmith or reporter = jbrown
```

- Find all issues that are overdue or where no due date is set:

```
duedate < now() or duedate is empty
```

[^ top of page](#)

NOT

Used to negate individual clauses or a complex JQL query (a query made up of more than one clause) using parentheses, allowing you to refine your search.

(Note: also see [NOT EQUALS](#) ("!="), [DOES NOT CONTAIN](#) ("!~"), [NOT IN](#) and [IS NOT](#).)
Examples

- Find all issues that are assigned to any user except jsmith:

```
not assignee = jsmith
```

- Find all issues that were not created by either jsmith or jbrown:

```
not (reporter = jsmith or reporter = jbrown)
```

[^ top of page](#)

EMPTY

Used to search for issues where a given field does not have a value. See also [NULL](#).

Note that EMPTY can only be used with fields that support the **IS** and **IS NOT** operators. To see a field's supported operators, check the individual [field](#) reference.

Examples

- Find all issues without a DueDate:

```
duedate = empty
```

or

```
duedate is empty
```

[^ top of page](#)

NULL

Used to search for issues where a given field does not have a value. See also [EMPTY](#).

Note that NULL can only be used with fields that support the **IS** and **IS NOT** operators. To see a field's supported operators, check the individual [field](#) reference.

Examples

- Find all issues without a DueDate:

```
duedate = null
```

or

```
duedate is null
```

[^ top of page](#)

ORDER BY

Used to specify the fields by whose values the search results will be sorted.

By default, the field's own sorting order will be used. You can override this by specifying ascending order ("as c") or descending order ("desc").

Examples

- Find all issues without a DueDate, sorted by CreationDate:

```
duedate = empty order by created
```

- Find all issues without a DueDate, sorted by CreationDate, then by Priority (highest to lowest):

```
duedate = empty order by created, priority desc
```

- Find all issues without a DueDate, sorted by CreationDate, then by Priority (lowest to highest):

```
duedate = empty order by created, priority asc
```

Ordering by **Components** or **Versions** will list the returned issues first by **Project**, and only then by the field's natural order (see [JRA-31113](#)).

[^ top of page](#)

Advanced searching - operators reference

This page describes information about operators that are used for advanced searching.

An operator in JQL is one or more symbols or words, which compares the value of a [field](#) on its left with one or more values (or [functions](#)) on its right, such that only true results are retrieved by the clause. Some operators may use the [NOT](#) keyword.

EQUALS: =

The "=" operator is used to search for issues where the value of the specified field exactly matches the specified value. (Note: cannot be used with text fields; see the [CONTAINS](#) operator instead.)

To find issues where the value of a specified field exactly matches *multiple* values, use multiple "=" statements with the [AND](#) operator.

Examples

- Find all issues that were created by jsmith:

```
reporter = jsmith
```

- Find all issues that were created by John Smith:

```
reporter = "John Smith"
```

List of Operators:

- EQUALS: =
- NOT EQUALS: !=
- GREATER THAN: >
- GREATER THAN EQUALS: >=
- LESS THAN: <
- LESS THAN EQUALS: <=
- IN
- NOT IN
- CONTAINS: ~
- DOES NOT CONTAIN: !=~
- IS
- IS NOT
- WAS
- WAS IN
- WAS NOT IN
- WAS NOT
- CHANGED

[^ top of page](#)

NOT EQUALS: !=

The "!=" operator is used to search for issues where the value of the specified field does not match the specified value. (Note: cannot be used with text fields; see the [DOES NOT MATCH](#) ("!~") operator instead.)

Note that typing `field != value` is the same as typing `NOT field = value`, and that `field != EMPTY` is the same as `field IS_NOT EMPTY`.

The "!=" operator will not match a field that has no value (i.e. a field that is empty). For example, `component != fred` will only match issues that have a component **and** the component is not "fred". To find issues that have a component other than "fred" **or have no component**, you would need to type: `component != fred OR component IS_EMPTY`.

Examples

- Find all issues that are assigned to any user except jsmith:

```
not assignee = jsmith
```

or:

```
assignee != jsmith
```

- Find all issues that are not assigned to jsmith:

```
assignee != jsmith or assignee is empty
```

- Find all issues that were reported by me but are not assigned to me:

```
reporter = currentUser() and assignee != currentUser()
```

- Find all issues where the Reporter or Assignee is anyone except John Smith:

```
assignee != "John Smith" or reporter != "John Smith"
```

- Find all issues that are not unassigned:

```
assignee is not empty
```

or

```
assignee != null
```

[^top of page](#)

GREATER THAN: >

The ">" operator is used to search for issues where the value of the specified field is greater than the specified value.

Note that the ">" operator can only be used with fields that support ordering (e.g. date fields and version fields), and cannot be used with text fields. To see a field's supported operators, check the individual field reference.

Examples

- Find all issues with more than 4 votes:

```
votes > 4
```

- Find all overdue issues:

```
duedate < now() and resolution is empty
```

- Find all issues where priority is higher than "Normal":

```
priority > normal
```

[^top of page](#)

GREATER THAN EQUALS: >=

The ">=" operator is used to search for issues where the value of the specified field is greater than or equal to the specified value.

Note that the ">=" operator can only be used with fields that support ordering (e.g. date fields and version fields), and cannot be used with text fields. To see a field's supported operators, check the individual field reference.

Examples

- Find all issues with 4 or more votes:

```
votes >= 4
```

- Find all issues due on or after 31/12/2008:

```
duedate >= "2008/12/31"
```

- Find all issues created in the last five days:

```
created >= "-5d"
```

[^top of page](#)

LESS THAN: <

The "<" operator is used to search for issues where the value of the specified field is less than the specified value.

Note that the "<" operator can only be used with fields which support ordering (e.g. date fields and version fields), and cannot be used with text fields. To see a field's supported operators, check the individual field reference.

Examples

- Find all issues with less than 4 votes:

```
votes < 4
```

[^top of page](#)

LESS THAN EQUALS: <=

The "<=" operator is used to search for issues where the value of the specified field is less than or equal to than the specified value.

Note that the "<=" operator can only be used with fields which support ordering (e.g. date fields and version fields), and cannot be used with text fields. To see a field's supported operators, check the individual field reference.

Examples

- Find all issues with 4 or fewer votes:

```
votes <= 4
```

- Find all issues that have not been updated in the past month (30 days):

```
updated <= "-4w 2d"
```

[^top of page](#)

IN

The "IN" operator is used to search for issues where the value of the specified field is one of multiple specified values. The values are specified as a comma-delimited list, surrounded by parentheses.

Using "IN" is equivalent to using multiple [EQUALS](#) (=) statements, but is shorter and more convenient. That is, typing reporter IN (tom, jane, harry) is the same as typing reporter = "tom" [OR](#) reporter = "jane" [OR](#) reporter = "harry".

Examples

- Find all issues that were created by either jsmith or jbrown or jjones:

```
reporter in (jsmith,jbrown,jjones)
```

- Find all issues where the Reporter or Assignee is either Jack or Jill:

```
reporter in (Jack,Jill) or assignee in (Jack,Jill)
```

- Find all issues in version 3.14 or version 4.2:

```
affectedVersion in ("3.14", "4.2")
```

[^top of page](#)

NOT IN

The "NOT IN" operator is used to search for issues where the value of the specified field is not one of multiple specified values.

Using "NOT IN" is equivalent to using multiple [NOT_EQUALS](#) (!=) statements, but is shorter and more convenient. That is, typing reporter NOT IN (tom, jane, harry) is the same as typing reporter != "tom" [AND](#) reporter != "jane" [AND](#) reporter != "harry".

The "NOT IN" operator will not match a field that has no value (i.e. a field that is empty). For example, assignee not in (jack,jill) will only match issues that have an assignee **and** the assignee is not "jack" or "jill". To find issues that are assigned to someone other than "jack" or "jill" **or are unassigned**, you would need to type: assignee not in (jack,jill) [or](#) assignee is empty.

Examples

- Find all issues where the Assignee is someone other than Jack, Jill, or John:

```
assignee not in (Jack,Jill,John)
```

- Find all issues where the Assignee is not Jack, Jill, or John:

```
assignee not in (Jack,Jill,John) or assignee is empty
```

- Find all issues where the FixVersion is not 'A', 'B', 'C', or 'D':

```
FixVersion not in (A, B, C, D)
```

- Find all issues where the FixVersion is not 'A', 'B', 'C', or 'D', or has not been specified:

```
FixVersion not in (A, B, C, D) or FixVersion is empty
```

[^top of page](#)

CONTAINS: ~

The "`~`" operator is used to search for issues where the value of the specified field matches the specified value (either an exact match or a "fuzzy" match — see examples below). For use with text fields only, i.e.:

- Summary
- Description
- Environment
- Comments
- custom fields that use the "Free Text Searcher"; this includes custom fields of the following built-in Custom Field Types
 - Free Text Field (unlimited text)
 - Text Field (< 255 characters)
 - Read-only Text Field

The JQL field "text" as in `text ~ "some words"` searches an issue's Summary, Description, Environment, Comments. It also searches all text custom fields. If you have many text custom fields you can improve performance of your queries by searching on specific fields, e.g.

`Summary ~ "some words" OR Description ~ "some words"`

Note: when using the "`~`" operator, the value on the right-hand side of the operator can be specified using [JIR A text-search syntax](#).

Examples

- Find all issues where the Summary contains the word "win" (or simple derivatives of that word, such as "wins"):

```
summary ~ win
```

- Find all issues where the Summary contains a wild-card match for the word "win":

```
summary ~ "win*"
```

- Find all issues where the Summary contains the word "issue" and the word "collector":

```
summary ~ "issue collector"
```

- Find all issues where the Summary contains the exact phrase "full screen" (see [Search syntax for text fields](#) for details on how to escape quote-marks and other special characters):

```
summary ~ "\"full screen\""
```

[^top of page](#)

DOES NOT CONTAIN: !~

The "`!~`" operator is used to search for issues where the value of the specified field is not a "fuzzy" match for the specified value. For use with text fields only, i.e.:

- Summary
- Description
- Environment
- Comments
- custom fields that use the "Free Text Searcher"; this includes custom fields of the following built-in Custom Field Types
 - Free Text Field (unlimited text)
 - Text Field (< 255 characters)
 - Read-only Text Field

The JQL field "text" as in `text ~ "some words"` searches an issue's Summary, Description, Environment, Comments. It also searches all text custom fields. If you have many text custom fields you can improve performance of your queries by searching on specific fields, e.g.

`Summary ~ "some words" OR Description ~ "some words"`

Note: when using the "`!~`" operator, the value on the right-hand side of the operator can be specified using [JIRA text-search syntax](#).

Examples

- Find all issues where the Summary does not contain the word "run" (or derivatives of that word, such as "running" or "ran"):

```
summary !~ run
```

[^top of page](#)

IS

The "`IS`" operator can only be used with [EMPTY](#) or [NULL](#). That is, it is used to search for issues where the specified field has no value.

Note that not all fields are compatible with this operator; see the individual field reference for details.

Examples

- Find all issues that have no Fix Version:

```
fixVersion is empty
```

or

```
fixVersion is null
```

[^top of page](#)

IS NOT

The "IS NOT" operator can only be used with **EMPTY** or **NULL**. That is, it is used to search for issues where the specified field has a value.

Note that not all fields are compatible with this operator; see the individual field reference for details.

Examples

- Find all issues that have one or more votes:

```
votes is not empty
```

or

```
votes is not null
```

[^top of page](#)

WAS

The "WAS" operator is used to find issues that currently have or previously had the specified value for the specified field.

This operator has the following optional predicates:

- AFTER "date"
- BEFORE "date"
- BY "username"
- DURING ("date1", "date2")
- ON "date"

This operator will match the value name (e.g. "Resolved"), which was configured in your system *at the time that the field was changed*. This operator will also match the value ID associated with that value name too — that is, it will match "4" as well as "Resolved".

(Note: This operator can be used with the Assignee, Fix Version, Priority, Reporter, Resolution, and Status fields only.)

Examples

- Find issues that currently have or previously had a status of 'In Progress':

```
status WAS "In Progress"
```

- Find issues that were resolved by Joe Smith before 2nd February:

```
status WAS "Resolved" BY jsmith BEFORE "2011/02/02"
```

- Find issues that were resolved by Joe Smith during 2010:

```
status WAS "Resolved" BY jsmith DURING  
("2010/01/01", "2011/01/01")
```

[^top of page](#)

WAS IN

The "WAS IN" operator is used to find issues that currently have or previously had any of multiple specified values for the specified field. The values are specified as a comma-delimited list, surrounded by parentheses.

Using "WAS IN" is equivalent to using multiple [WAS](#) statements, but is shorter and more convenient. That is, typing `status WAS IN ('Resolved', 'Closed')` is the same as typing `status WAS "Resolved" OR status WAS "Closed"`.

This operator has the following optional predicates:

- AFTER "date"
- BEFORE "date"
- BY "username"
- DURING ("date1", "date2")
- ON "date"

This operator will match the value name (e.g. "Resolved"), which was configured in your system *at the time that the field was changed*. This operator will also match the value ID associated with that value name too — that is, it will match "4" as well as "Resolved".

(Note: This operator can be used with the Assignee, Fix Version, Priority, Reporter, Resolution, and Status fields only.)

Examples

- Find all issues that currently have, or previously had, a status of 'Resolved' or 'In Progress':

```
status WAS IN ("Resolved", "In Progress")
```

[^top of page](#)

WAS NOT IN

The "WAS NOT IN" operator is used to search for issues where the value of the specified field has never been one of multiple specified values.

Using "WAS NOT IN" is equivalent to using multiple [WAS_NOT](#) statements, but is shorter and more convenient. That is, typing `status WAS NOT IN ("Resolved", "In Progress")` is the same as typing `status WAS NOT "Resolved" AND status WAS NOT "In Progress"`.

This operator has the following optional predicates:

- AFTER "date"
- BEFORE "date"
- BY "username"
- DURING ("date1", "date2")
- ON "date"

This operator will match the value name (e.g. "Resolved"), which was configured in your system *at the time that the field was changed*. This operator will also match the value ID associated with that value name, too — that is, it will match "4" as well as "Resolved".

(Note: This operator can be used with the Assignee, Fix Version, Priority, Reporter, Resolution, and Status fields only.)

Examples

- Find issues that have never had a status of 'Resolved' or 'In Progress':

```
status WAS NOT IN ("Resolved", "In Progress")
```

- Find issues that did not have a status of 'Resolved' or 'In Progress' before 2nd February:

```
status WAS NOT IN ("Resolved", "In Progress") BEFORE "2011/02/02"
```

[^top of page](#)

WAS NOT

The "WAS NOT" operator is used to find issues that have never had the specified value for the specified field.

This operator has the following optional predicates:

- AFTER "date"
- BEFORE "date"
- BY "username"
- DURING ("date1", "date2")
- ON "date"

This operator will match the value name (e.g. "Resolved"), which was configured in your system *at the time that the field was changed*. This operator will also match the value ID associated with that value name too — that is, it will match "4" as well as "Resolved".

(Note: This operator can be used with the Assignee, Fix Version, Priority, Reporter, Resolution, and Status fields only.)

Examples

- Find issues that do not have, and have never had a status of 'In Progress':

```
status WAS NOT "In Progress"
```

- Find issues that did not have a status of 'In Progress' before 2nd February:

```
status WAS NOT "In Progress" BEFORE "2011/02/02"
```

[^top of page](#)

CHANGED

The "CHANGED" operator is used to find issues that have a value that had changed for the specified field.

This operator has the following optional predicates:

- AFTER "date"
- BEFORE "date"
- BY "username"
- DURING ("date1", "date2")
- ON "date"
- FROM "oldvalue"
- TO "newvalue"

(Note: This operator can be used with the Assignee, Fix Version, Priority, Reporter, Resolution, and Status fields only.)

Examples

- Find issues whose assignee had changed:

```
assignee CHANGED
```

- Find issues whose status had changed from 'In Progress' back to 'Open':

```
status CHANGED FROM "In Progress" TO "Open"
```

- Find issues whose priority was changed by user 'freddo' after the start and before the end of the current week.

```
priority CHANGED BY freddo BEFORE endOfWeek() AFTER
startOfWeek()
```

[^top of page](#)

Advanced searching - functions reference

This page describes information about functions that are used for advanced searching.

A function in JQL appears as a word followed by parentheses, which may contain one or more explicit values or JIRA fields. In a clause, a function is preceded by an **operator**, which in turn is preceded by a **field**. A function performs a calculation on either specific JIRA data or the function's content in parentheses, such that only true results are retrieved by the function, and then again by the clause in which the function is used.

Unless specified in the search query, note that JQL searches do not return empty fields in results. To include empty fields (e.g. unassigned issues) when searching for issues that are not assigned to the current user, you would enter (assignee != currentUser() OR assignee is EMPTY) to include unassigned issues in the list of results.

approved()

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that required approval and have a final decision of approved.

List of functions:

- [approved\(\)](#)
- [approver\(\)](#)
- [cascadeOption\(\)](#)
- [closedSprints\(\)](#)
- [componentsLeadByUser\(\)](#)
- [currentLogin\(\)](#)
- [currentUser\(\)](#)
- [earliestUnreleasedVersion\(\)](#)
- [endOfDay\(\)](#)
- [endOfMonth\(\)](#)
- [endOfWeek\(\)](#)
- [endOfYear\(\)](#)
- [issueHistory\(\)](#)
- [issuesWithRemoteLinksByGlobalId\(\)](#)
- [lastLogin\(\)](#)
- [latestReleasedVersion\(\)](#)
- [linkedIssues\(\)](#)
- [membersOf\(\)](#)
- [myApproval\(\)](#)
- [myPending\(\)](#)
- [now\(\)](#)
- [openSprints\(\)](#)
- [pending\(\)](#)
- [pendingBy\(\)](#)
- [projectsLeadByUser\(\)](#)
- [projectsWhereUserHasPermission\(\)](#)
- [projectsWhereUserHasRole\(\)](#)
- [releasedVersions\(\)](#)
- [standardIssueTypes\(\)](#)

Syntax	approved()
Supported fields	Custom fields of type Approval
Supported operators	=
Unsupported operators	~, !=, !~, >, >=, <, <=, IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED

Examples

- Find all issues that are approved:
approval = approved()

[^ top of page](#)

- startOfDay()
- startOfMonth()
- startOfWeek()
- startOfYear()
- subtaskIssueTypes()
- unreleasedVersions()
- votedIssues()
- watchedIssues()

approver()

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that require or required approval by the listed user/s. This uses an OR operator, and you must specify the username/s.

Syntax	approver(user,user)
Supported fields	Custom fields of type Approval
Supported operators	=
Unsupported operators	<code>~, !=, !~, >, >=, <, <=</code> <code>IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Examples	<ul style="list-style-type: none"> Find issues that require or required approval by John Smith: approval = approver(jsmith) Find issues that require or required approval by John Smith or Sarah Khan: approval = approver(jsmith,skhan)

[^ top of page](#)**cascadeOption()**

Search for issues that match the selected values of a 'cascading select' custom field.

The *parentOption* parameter matches against the first tier of options in the cascading select field. The *childOption* parameter matches against the second tier of options in the cascading select field, and is optional.

The keyword "none" can be used to search for issues where either or both of the options have no value.

Syntax	cascadeOption(parentOption) cascadeOption(parentOption,childOption)
Supported fields	Custom fields of type 'Cascading Select'
Supported operators	<code>IN, NOT IN</code>
Unsupported operators	<code>=, !=, ~, !~, >, >=, <, <=</code> <code>IS, IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>

Examples <ul style="list-style-type: none"> Find issues where a custom field ("Location") has the value "USA" for the first tier and "New York" for the second tier: <code>location in cascadeOption("USA", "New York")</code> Find issues where a custom field ("Location") has the value "USA" for the first tier and any value (or no value) for the second tier: <code>location in cascadeOption("USA")</code> Find issues where a custom field ("Location") has the value "USA" for the first tier and no value for the second tier: <code>location in cascadeOption("USA", none)</code> Find issues where a custom field ("Location") has no value for the first tier and no value for the second tier: <code>location in cascadeOption(none)</code> Find issues where a custom field ("Referrer") has the value "none" for the first tier and "none" for the second tier: <code>referrer in cascadeOption("\\"none\\\"", "\\"none\\\"")</code> Find issues where a custom field ("Referrer") has the value "none" for the first tier and no value for the second tier: <code>referrer in cascadeOption("\\"none\\\"", none)</code>
--

[^ top of page](#)

`closedSprints()`

Search for issues that are assigned to a completed Sprint. Note, it is possible for an issue to belong to both a completed Sprint(s) and an incomplete Sprint(s). See also [openSprints\(\)](#).

Syntax	<code>closedSprints()</code>
Supported fields	Sprint
Supported operators	<code>IN</code> , <code>NOT IN</code>
Unsupported operators	<code>=</code> , <code>!=</code> , <code>~</code> , <code>!~</code> , <code>></code> , <code>>=</code> , <code><</code> , <code><=</code> <code>IS</code> , <code>IS NOT</code> , <code>WAS</code> , <code>WAS IN</code> , <code>WAS NOT</code> , <code>WAS NOT IN</code> , <code>CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that are assigned to a completed sprint: <code>sprint in closedSprints()</code>

[^ top of page](#)

`componentsLeadByUser()`

Find issues in components that are led by a specific user. You can optionally specify a user, or if the user is omitted, the current user (i.e. you) will be used. Note that if you are not logged in to JIRA, a user must be specified.

Syntax	<code>componentsLeadByUser()</code> <code>componentsLeadByUser(username)</code>
Supported fields	Component
Supported operators	<code>IN</code> , <code>NOT IN</code>
Unsupported operators	<code>=</code> , <code>!=</code> , <code>~</code> , <code>!~</code> , <code>></code> , <code>>=</code> , <code><</code> , <code><=</code> <code>IS</code> , <code>IS NOT</code> , <code>WAS</code> , <code>WAS IN</code> , <code>WAS NOT</code> , <code>WAS NOT IN</code> , <code>CHANGED</code>

Examples	<ul style="list-style-type: none"> Find open issues in components that are led by you: <code>component in componentsLeadByUser() AND status = Open</code> Find open issues in components that are led by Bill: <code>component in componentsLeadByUser(bill) AND status = Open</code>
-----------------	---

[^ top of page](#)

currentLogin()

Perform searches based on the time at which the current user's session began. See also [lastLogin](#).

Syntax	<code>currentLogin()</code>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	<code>= , != , > , >= , < , <=</code> <code>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</code> <small>* Only in predicate</small>
Unsupported operators	<code>~ , !~ IS , IS NOT , IN , NOT IN</code>
Examples	<ul style="list-style-type: none"> Find issues that have been created during my current session: <code>created > currentLogin()</code>

[^ top of page](#)

currentUser()

Perform searches based on the currently logged-in user. Note, this function can only be used by logged-in users. So if you are creating a saved filter that you expect to be used by anonymous users, do not use this function.

Syntax	<code>currentUser()</code>
Supported fields	Assignee, Reporter, Voter, Watcher, custom fields of type User
Supported operators	<code>= , !=</code>
Unsupported operators	<code>~ , !~ , > , >= , < , <= IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find issues that are assigned to me: <code>assignee = currentUser()</code> Find issues that were reported to me but are not assigned to me: <code>reporter = currentUser() AND (assignee != currentUser() OR assignee is EMPTY)</code>

[^ top of page](#)

earliestUnreleasedVersion()

Perform searches based on the earliest unreleased version (i.e. next version that is due to be released) of a specified project. See also [unreleasedVersions](#). Note, the "earliest" is determined by the ordering assigned to the versions, not by actual Version Due Dates.

Syntax	<code>earliestUnreleasedVersion(project)</code>
Supported fields	AffectedVersion, FixVersion, custom fields of type Version
Supported operators	<code>IN , NOT IN</code>
Unsupported operators	<code>= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find issues whose FixVersion is the earliest unreleased version of the ABC project: <code>fixVersion = earliestUnreleasedVersion(ABC)</code> Find issues that relate to the earliest unreleased version of the ABC project: <code>affectedVersion = earliestUnreleasedVersion(ABC) or fixVersion = earliestUnreleasedVersion(ABC)</code>

[^ top of page](#)

`endOfDay()`

Perform searches based on the end of the current day. See also [endOfWeek](#), [endOfMonth](#), and [endOfYear](#); and [startOfDay](#), [startOfWeek](#), [startOfMonth](#), and [startOfYear](#).

Syntax	<code>endOfDay()</code> <code>endOfDay("inc")</code> <i>where inc is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>endOfDay("+1")</code> is the same as <code>endOfDay("+1d")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</i>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	<code>= , != , > , >= , < , <=</code> <code>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</code> <i>* Only in predicate</i>
Unsupported operators	<code>~ , !~ IS , IS NOT , IN , NOT IN</code>
Examples	<ul style="list-style-type: none"> Find issues due by the end of today: <code>due < endOfDay()</code> Find issues due by the end of tomorrow: <code>due < endOfDay("+1")</code>

[^ top of page](#)

`endOfMonth()`

Perform searches based on the end of the current month. See also [endOfDay](#), [endOfWeek](#), and [endOfYear](#); and [startOfDay](#), [startOfWeek](#), [startOfMonth](#), and [startOfYear](#).

Syntax	<pre>endOfMonth()</pre> <pre>endOfMonth("inc")</pre> <p>where <i>inc</i> is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>endOfMonth("+1")</code> is the same as <code>endOfMonth("+1M")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</p>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	$=$, $!=$, $>$, \geq , $<$, \leq WAS^* , WAS IN^* , WAS NOT^* , WAS NOT IN^* , CHANGED^* <i>* Only in predicate</i>
Unsupported operators	\sim , !~ IS , IS NOT , IN , NOT IN
Examples	<ul style="list-style-type: none"> Find issues due by the end of this month: <code>due < endOfMonth()</code> Find issues due by the end of next month: <code>due < endOfMonth(" +1 ")</code> Find issues due by the 15th of next month: <code>due < endOfMonth(" +15d ")</code>

[^ top of page](#)

endOfWeek()

Perform searches based on the end of the current week. See also `endOfDay`, `endOfMonth`, and `endOfYear`; and `startOfDay`, `startOfWeek`, `startOfMonth`, and `startOfYear`.

For the `endOfWeek()` function, the result depends upon your locale. For example, in Europe, the first day of the week is generally considered to be Monday, while in the USA, it is considered to be Sunday.

Syntax	<pre>endOfWeek()</pre> <pre>endOfWeek("inc")</pre> <p>where <i>inc</i> is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>endOfWeek("+1")</code> is the same as <code>endOfWeek("+1w")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</p>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	$=$, $!=$, $>$, \geq , $<$, \leq WAS^* , WAS IN^* , WAS NOT^* , WAS NOT IN^* , CHANGED^* <i>* Only in predicate</i>
Unsupported operators	\sim , !~ IS , IS NOT , IN , NOT IN
Examples	<ul style="list-style-type: none"> Find issues due by the end of this week: <code>due < endOfWeek()</code> Find issues due by the end of next week: <code>due < endOfWeek(" +1 ")</code>

[^ top of page](#)

[endOfYear\(\)](#)

Perform searches based on the end of the current year. See also [startOfDay](#), [startOfWeek](#), and [startOfMonth](#); and [endOfDay](#), [endOfWeek](#), [endOfMonth](#), and [endOfYear](#).

Syntax	<pre>endOfYear() endOfYear("inc")</pre> <p>where <i>inc</i> is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>endOfYear("+1")</code> is the same as <code>endOfYear("+1y")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</p>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	=, !=, >, >=, <, <=, WAS*, WAS IN*, WAS NOT*, WAS NOT IN*, CHANGED* * Only in predicate
Unsupported operators	~, !~ IS, IS NOT, IN, NOT IN
Examples	<ul style="list-style-type: none"> Find issues due by the end of this year: <code>due < endOfYear()</code> Find issues due by the end of March next year: <code>due < endOfYear(" +3M")</code>

[^ top of page](#)

[issueHistory\(\)](#)

Find issues that you have recently viewed, i.e. issues that are in the 'Recent Issues' section of the 'Issues' drop-down menu.

Note:

- `issueHistory()` returns up to 50 issues, whereas the 'Recent Issues' drop-down returns only 5.
- if you are not logged in to JIRA, only issues from your current browser session will be included.

Syntax	<code>issueHistory()</code>
Supported fields	Issue
Supported operators	IN, NOT IN
Unsupported operators	=, !=, ~, !~, >, >=, <, <= IS, IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Examples	<ul style="list-style-type: none"> Find issues which I have recently viewed, that are assigned to me: <code>issue in issueHistory() AND assignee = currentUser()</code>

[^ top of page](#)

[issuesWithRemoteLinksByGlobalId\(\)](#)

Perform searches based on issues that are associated with remote links that have any of the specified global ids.

Note:

- This function accepts 1 to 100 globalIds. Specifying 0 or more than 100 globalIds will result in errors.

Syntax	<code>issuesWithRemoteLinksByGlobalId()</code>
Supported fields	<code>Issue</code>
Supported operators	<code>IN , NOT IN</code>
Unsupported operators	<code>= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>
Examples	<ul style="list-style-type: none"> • Find issues that are linked to remote links that have globalId "abc": <code>issue in issuesWithRemoteLinksByGlobalId(abc)</code> • Find issues that are linked to remote links that have either globalId "abc" or "def": <code>issue in issuesWithRemoteLinksByGlobalId(abc, def)</code>

[^ top of page](#)

`lastLogin()`

Perform searches based on the time at which the current user's previous session began. See also [currentLogin](#).

Syntax	<code>lastLogin()</code>
Supported fields	<code>Created, Due, Resolved, Updated, custom fields of type Date/Time</code>
Supported operators	<code>= , != , > , >= , < , <=</code> <code>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</code> <i>* only in predicate</i>
Unsupported operators	<code>~ , !~ IS , IS NOT , IN , NOT IN</code>
Examples	<ul style="list-style-type: none"> • Find issues that have been created during my last session: <code>created > lastLogin()</code>

[^ top of page](#)

`latestReleasedVersion()`

Perform searches based on the latest released version (i.e. the most recent version that has been released) of a specified project. See also [releasedVersions\(\)](#). Note, the "latest" is determined by the ordering assigned to the versions, not by actual Version Due Dates.

Syntax	<code>latestReleasedVersion(project)</code>
Supported fields	<code>AffectedVersion, FixVersion, custom fields of type Version</code>
Supported operators	<code>IN , NOT IN</code>
Unsupported operators	<code>= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>

Examples	<ul style="list-style-type: none"> Find issues whose FixVersion is the latest released version of the ABC project: fixVersion = latestReleasedVersion(ABC) Find issues that relate to the latest released version of the ABC project: affectedVersion = latestReleasedVersion(ABC) or fixVersion = latestReleasedVersion(ABC)
-----------------	---

[^ top of page](#)**linkedIssues()**

Perform searches based on issues that are linked to a specified issue. You can optionally restrict the search to links of a particular type. Note that LinkType is case-sensitive.

Syntax	linkedIssues(issueKey) linkedIssues(issueKey,linkType)
Supported fields	Issue
Supported operators	IN , NOT IN
Unsupported operators	= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Examples	<ul style="list-style-type: none"> Find issues that are linked to a particular issue: issue in linkedIssues(ABC-123) Find issues that are linked to a particular issue via a particular type of link: issue in linkedIssues(ABC-123,"is duplicated by")

[^ top of page](#)**membersOf()**

Perform searches based on the members of a particular group.

Syntax	membersOf(Group)
Supported fields	Assignee, Reporter, Voter, Watcher, custom fields of type User
Supported operators	IN , NOT IN
Unsupported operators	= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Examples	<ul style="list-style-type: none"> Find issues where the Assignee is a member of the group "jira-administrators": assignee in membersOf("jira-administrators") Search through multiple groups and a specific user: reporter in membersOf("jira-administrators") or reporter in membersOf("jira-core-users") or reporter=jsmith Search for a particular group, but exclude a particular member or members: assignee in membersOf(QA) and assignee not in ("John Smith","Jill Jones") Exclude members of a particular group: assignee not in membersOf(QA)

[^ top of page](#)

myApproval()

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that require approval or have required approval by the current user.

Syntax	myApproval()
Supported fields	Custom fields of type Approval
Supported operators	=
Unsupported operators	<code>~ , != , !~ , > , >= , < , <=</code> <code>IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN</code> <code>, CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that require or have required my approval approval = myApproval()

[^ top of page](#)

myPending()

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that require approval by the current user.

Syntax	approved()
Supported fields	Custom fields of type Approval
Supported operators	=
Unsupported operators	<code>~ , != , !~ , > , >= , < , <=</code> <code>IS , IS NOT , IN , NOT IN , WAS , WAS IN , WAS NOT , WAS NOT IN</code> <code>, CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that require my approval approval = myPending()

[^ top of page](#)

now()

Perform searches based on the current time.

Syntax	now()
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	<code>= , != , > , >= , < , <=</code> <code>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</code> <small>* Only in predicate</small>

Unsupported operators	<code>~, !~ IS, IS NOT, IN, NOT IN</code>
Examples	<ul style="list-style-type: none"> Find issues that are overdue: <code>duedate < now() and status not in (closed, resolved)</code>

[^ top of page](#)

openSprints()

Search for issues that are assigned to a Sprint that has not yet been completed. Note, it is possible for an issue to belong to both a completed Sprint(s) and an incomplete Sprint(s). See also [closedSprints\(\)](#).

Syntax	<code>openSprints()</code>
Supported fields	Sprint
Supported operators	<code>IN, NOT IN</code>
Unsupported operators	<code>=, !=, ~, !~, >, >=, <, <=</code> <code>IS, IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that are assigned to a sprint that has not yet been completed: <code>sprint in openSprints()</code>

[^ top of page](#)

pending()

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that require approval.

Syntax	<code>pending()</code>
Supported fields	Custom fields of type Approval
Supported operators	<code>=</code>
Unsupported operators	<code>~, !=, !~, >, >=, <, <=</code> <code>IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED</code>
Examples	<ul style="list-style-type: none"> Find all issues that require approval: <code>approval = pending()</code>

[^ top of page](#)

pendingBy()

Only applicable if JIRA Service Desk is installed and licensed.

Search for issues that require approval by the listed user/s. This uses an OR operator, and you must specify the username/s.

Syntax	<code>pendingBy(user1,user2)</code>
---------------	-------------------------------------

Supported fields	Custom fields of type Approval
Supported operators	=
Unsupported operators	=, !=, ~, !~, >, >=, <, <=, IS, IS NOT, IN, NOT IN, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Examples	<ul style="list-style-type: none"> Find issues that require approval by John Smith: <code>approval = pending(jsmith)</code> Find issues that require by John Smith or Sarah Khan: <code>approval = pending(jsmith,skhan)</code>

[^ top of page](#)

projectsLeadByUser()

Find issues in projects that are led by a specific user. You can optionally specify a user, or if the user is omitted, the current user will be used. Note that if you are not logged in to JIRA, a user must be specified.

Syntax	<code>projectsLeadByUser()</code> <code>projectsLeadByUser(username)</code>
Supported fields	Project
Supported operators	IN, NOT IN
Unsupported operators	=, !=, ~, !~, >, >=, <, <=, IS, IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED
Examples	<ul style="list-style-type: none"> Find open issues in projects that are led by you: <code>project in projectsLeadByUser() AND status = Open</code> Find open issues in projects that are led by Bill: <code>project in projectsLeadByUser(bill) AND status = Open</code>

[^ top of page](#)

projectsWhereUserHasPermission()

Find issues in projects where you have a specific permission. Note, this function operates at the project level. This means that if a permission (e.g. "Edit Issues") is granted to the reporter of issues in a project, then you may see some issues returned where you are not the reporter, and therefore don't have the permission specified. Also note, this function is only available if you are logged in to JIRA.

Syntax	<code>projectsWhereUserHasPermission(permission)</code> For the <code>permission</code> parameter, you can specify any of the permissions described on .
Supported fields	Project
Supported operators	IN, NOT IN
Unsupported operators	=, !=, ~, !~, >, >=, <, <=, IS, IS NOT, WAS, WAS IN, WAS NOT, WAS NOT IN, CHANGED

Examples	<ul style="list-style-type: none"> Find open issues in projects where you have the "Resolve Issues" permission: <code>project in projectsWhereUserHasPermission("Resolve Issues") AND status = Open</code>
-----------------	--

[^ top of page](#)

`projectsWhereUserHasRole()`

Find issues in projects where you have a specific role. Note, this function is only available if you are logged in to JIRA.

Syntax	<code>projectsWhereUserHasRole(roletype)</code>
Supported fields	Project
Supported operators	<code>IN , NOT IN</code>
Unsupported operators	<code>= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find open issues in projects where you have the "Developers" role: <code>project in projectsWhereUserHasRole("Developers") AND status = Open</code>

[^ top of page](#)

`releasedVersions()`

Perform searches based on the released versions (i.e. versions that your JIRA administrator has released) of a specified project. You can also search on the released versions of all projects, by omitting the `project` parameter. See also `latestReleasedVersion()`.

Syntax	<code>releasedVersions()</code> <code>releasedVersions(project)</code>
Supported fields	AffectedVersion, FixVersion, custom fields of type Version
Supported operators	<code>IN , NOT IN</code>
Unsupported operators	<code>= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find issues whose FixVersion is a released version of the ABC project: <code>fixVersion in releasedVersions(ABC)</code> Find issues that relate to released versions of the ABC project: <code>(affectedVersion in releasedVersions(ABC)) or (fixVersion in releasedVersions(ABC))</code>

[^ top of page](#)

`standardIssueTypes()`

Perform searches based on "standard" Issue Types, that is, search for issues that are not sub-tasks. See also `subtaskIssueTypes()`.

Syntax	<code>standardIssueTypes()</code>
Supported fields	Type
Supported operators	<code>IN</code> , <code>NOT IN</code>
Unsupported operators	<code>=</code> , <code>!=</code> , <code>~</code> , <code>!~</code> , <code>></code> , <code>>=</code> , <code><</code> , <code><=</code> <code>IS</code> , <code>IS NOT</code> , <code>WAS</code> , <code>WAS IN</code> , <code>WAS NOT</code> , <code>WAS NOT IN</code> , <code>CHANGED</code>
Examples	<ul style="list-style-type: none"> Find issues that are not subtasks (i.e. issues whose Issue Type is a standard issue type, not a subtask issue type): <code>issuetype in standardIssueTypes()</code>

[^ top of page](#)

startOfDay()

Perform searches based on the start of the current day. See also [startOfWeek](#), [startOfMonth](#), and [startOfYear](#); and [endOfDay](#), [endOfWeek](#), [endOfMonth](#), and [endOfYear](#).

Syntax	<code>startOfDay()</code> <code>startOfDay("inc")</code> <i>where inc is an optional increment of (+/-)nn(Y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>startOfDay("+1")</code> is the same as <code>startOfDay("+1d")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</i>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	<code>=</code> , <code>!=</code> , <code>></code> , <code>>=</code> , <code><</code> , <code><=</code> <code>WAS*</code> , <code>WAS IN*</code> , <code>WAS NOT*</code> , <code>WAS NOT IN*</code> , <code>CHANGED*</code> <i>* Only in predicate</i>
Unsupported operators	<code>~</code> , <code>!~</code> <code>IS</code> , <code>IS NOT</code> , <code>IN</code> , <code>NOT IN</code>
Examples	<ul style="list-style-type: none"> Find new issues created since the start of today: <code>created > startOfDay()</code> Find new issues created since the start of yesterday: <code>created > startOfDay("-1")</code> Find new issues created in the last three days: <code>created > startOfDay("-3d")</code>

[^ top of page](#)

startOfMonth()

Perform searches based on the start of the current month. See also [startOfDay](#), [startOfWeek](#), and [startOfYear](#); and [endOfDay](#), [endOfWeek](#), [endOfMonth](#), and [endOfYear](#).

Syntax	<pre>startOfMonth()</pre> <pre>startOfMonth("inc")</pre> <p>where <i>inc</i> is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>startOfMonth("+1")</code> is the same as <code>startOfMonth("+1M")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</p>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	$=$, $!=$, $>$, \geq , $<$, \leq WAS^* , WAS IN^* , WAS NOT^* , WAS NOT IN^* , CHANGED^* <i>* Only in predicate</i>
Unsupported operators	\sim , !~ IS , IS NOT , IN , NOT IN
Examples	<ul style="list-style-type: none"> Find new issues created since the start of this month: <code>created > startOfMonth()</code> Find new issues created since the start of last month: <code>created > startOfMonth("-1")</code> Find new issues created since the 15th of this month: <code>created > startOfMonth("+14d")</code>

[^ top of page](#)

startOfWeek()

Perform searches based on the start of the current week. See also [startOfDay](#), [startOfMonth](#), and [startOfYear](#); and [endOfDay](#), [endOfWeek](#), [endOfMonth](#), and [endOfYear](#). For the `startOfWeek()` function, the result depends upon your locale. For example, in Europe, the first day of the week is generally considered to be Monday, while in the USA, it is considered to be Sunday.

Syntax	<pre>startOfWeek()</pre> <pre>startOfWeek("inc")</pre> <p>where <i>inc</i> is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>startOfWeek("+1")</code> is the same as <code>startOfWeek("+1w")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</p>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	$=$, $!=$, $>$, \geq , $<$, \leq WAS^* , WAS IN^* , WAS NOT^* , WAS NOT IN^* , CHANGED^* <i>* Only in predicate</i>
Unsupported operators	\sim , !~ IS , IS NOT , IN , NOT IN
Examples	<ul style="list-style-type: none"> Find new issues since the start of this week: <code>created > startOfWeek()</code> Find new issues since the start of last week: <code>created > startOfWeek("-1")</code>

[^ top of page](#)

startOfYear()

Perform searches based on the start of the current year. See also `startOfDay`, `startOfWeek` and `startOfMonth`; and `endOfDay`, `endOfWeek`, `endOfMonth` and `endOfYear`.

Syntax	<code>startOfYear()</code> <code>startOfYear("inc")</code> <i>where inc is an optional increment of (+/-)nn(y/M/w/d/h/m). If the time unit qualifier is omitted, it defaults to the natural period of the function, e.g. <code>endOfYear("+1")</code> is the same as <code>endOfYear("+1y")</code>. If the plus/minus (+/-) sign is omitted, plus is assumed.</i>
Supported fields	Created, Due, Resolved, Updated, custom fields of type Date/Time
Supported operators	<code>= , != , > , >= , < , <=</code> <code>WAS* , WAS IN* , WAS NOT* , WAS NOT IN* , CHANGED*</code> <i>* Only in predicate</i>
Unsupported operators	<code>~ , !~ IS , IS NOT , IN , NOT IN</code>
Examples	<ul style="list-style-type: none"> Find new issues since the start of this year: <code>created > startOfYear()</code> Find new issues since the start of last year: <code>created > startOfYear("-1")</code>

[^ top of page](#)

`subtaskIssueTypes()`

Perform searches based on issues that are sub-tasks. See also `standardIssueTypes()`.

Syntax	<code>subtaskIssueTypes()</code>
Supported fields	Type
Supported operators	<code>IN , NOT IN</code>
Unsupported operators	<code>= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED</code>
Examples	<ul style="list-style-type: none"> Find issues that are subtasks (i.e. issues whose Issue Type is a subtask issue type): <code>issuetype in subtaskIssueTypes()</code>

[^ top of page](#)

`unreleasedVersions()`

Perform searches based on the unreleased versions (i.e. versions that your JIRA administrator has not yet released) of a specified project. You can also search on the unreleased versions of all projects, by omitting the `project` parameter. See also `earliestUnreleasedVersion()`.

Syntax	<code>unreleasedVersions()</code> <code>unreleasedVersions(project)</code>
---------------	---

Supported fields	AffectedVersion, FixVersion, custom fields of type Version
Supported operators	IN , NOT IN
Unsupported operators	= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Examples	<ul style="list-style-type: none"> Find issues whose FixVersion is an unreleased version of the ABC project: fixVersion in unreleasedVersions(ABC) Find issues that relate to unreleased versions of the ABC project: affectedVersion in unreleasedVersions(ABC)

[^ top of page](#)

votedIssues()

Perform searches based on issues for which you have voted. Also, see the Voter field. Note, this function can only be used by logged-in users.

Syntax	votedIssues()
Supported fields	Issue
Supported operators	IN , NOT IN
Unsupported operators	= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Examples	<ul style="list-style-type: none"> Find issues that you have voted for: issue in votedIssues()

[^ top of page](#)

watchedIssues()

Perform searches based on issues that you are watching. Also, see the Watcher field. Note that this function can only be used by logged-in users.

Syntax	watchedIssues()
Supported fields	Issue
Supported operators	IN , NOT IN
Unsupported operators	= , != , ~ , !~ , > , >= , < , <= IS , IS NOT , WAS , WAS IN , WAS NOT , WAS NOT IN , CHANGED
Examples	<ul style="list-style-type: none"> Find issues that you are watching: issue in watchedIssues()

[^ top of page](#)

Search syntax for text fields

This page provides information on the syntax for searching text fields, which

can be done in the quick search, basic search, and advanced search.

Text searches can be done in the advanced search when the [CONTAINS \(~\) operator](#) is used, e.g. `summary~"windows*`. It can also be done in quick search and basic search when searching on supported fields.

Acknowledgments: JIRA uses Apache Lucene for text indexing, which provides a rich query language. Much of the information on this page is derived from the [Query Parser Syntax](#) page of the Lucene documentation.

On this page:

- [Query terms](#)
- [Term modifiers](#)
- [Boosting a term: ^](#)
- [Boolean operators](#)
- [Grouping](#)
- [Escaping special characters: \ or \\](#)
- [Reserved words](#)
- [Word stemming](#)
- [Limitations](#)
- [Next steps](#)

Query terms

A query is broken up into **terms** and **operators**. There are two types of terms: **Single Terms** and **Phrases**.

A **Single Term** is a single word, such as "test" or "hello".

A **Phrase** is a group of words surrounded by double quotes, such as "hello dolly".

Multiple terms can be combined together with Boolean operators to form a more complex query (see below). If you combine multiple terms without specifying any Boolean operators, they will be joined using AND operators.

Note: All query terms in JIRA are not case sensitive.

Term modifiers

JIRA supports modifying query terms to provide a wide range of searching options.

[Wildcard searches: ? and *](#) | [Fuzzy searches: ~](#) | [Proximity searches](#)

Wildcard searches: ? and *

JIRA supports single and multiple character wildcard searches.

To perform a single character wildcard search, use the "?" symbol.

To perform a multiple character wildcard search, use the "*" symbol.

Wildcard characters need to be enclosed in quote-marks, as they are reserved characters in advanced search. Use quotations, e.g. `summary ~ "cha?k` and `che*`"

The single character wildcard search looks for terms that match that with the single character replaced. For example, to search for "text" or "test", you can use the search:

te?t

Multiple character wildcard searches looks for 0 or more characters. For example, to search for Windows, Win95, or WindowsNT, you can use the search:

```
win*
```

You can also use the wildcard searches in the middle of a term. For example, to search for Win95 or Windo ws95, you can use the search:

```
wi*95
```

You cannot use a * or ? symbol as the first character of a search. The feature request for this is [JRA-6218](#).

Fuzzy searches: ~

JIRA supports fuzzy searches. To do a fuzzy search, use the tilde, "~", symbol at the end of a single word term. For example, to search for a term similar in spelling to "roam", use the fuzzy search:

```
roam~
```

This search will find terms like foam and roams.

Note: Terms found by the fuzzy search will automatically get a boost factor of 0.2.

Proximity searches

JIRA supports finding words that are within a specific distance away. To do a proximity search, use the tilde, "~", symbol at the end of a phrase. For example, to search for "atlassian" and "jira" within 10 words of each other in a document, use the search:

```
"atlassian jira"~10
```

Boosting a term: ^

JIRA provides the relevance level of matching documents based on the terms found. To boost a term, use the caret, "^", symbol with a boost factor (a number) at the end of the term you are searching. The higher the boost factor, the more relevant the term will be.

Boosting allows you to control the relevance of a document by boosting its term. For example, if you are searching for

```
atlassian jira
```

and you want the term "atlassian" to be more relevant, boost it using the ^ symbol along with the boost factor next to the term. You would type:

```
atlassian^4 jira
```

This will make documents with the term atlassian appear more relevant. You can also boost Phrase Terms, as in the example:

```
"atlassian jira" ^4 querying
```

By default, the boost factor is 1. Although, the boost factor must be positive, it can be less than 1 (i.e. 0.2).

Boolean operators

Boolean operators allow terms to be combined through logic operators. JIRA supports AND, "+", OR, NOT and "-" as Boolean operators.

Boolean operators must be ALL CAPS.

OR | AND | Required term: + | NOT | Excluded term: -

OR

The OR operator is the default conjunction operator. This means that if there is no Boolean operator between two terms, the OR operator is used. The OR operator links two terms, and finds a matching document if either of the terms exist in a document. This is equivalent to a union using sets. The symbol || can be used in place of the word OR.

To search for documents that contain either "atlassian jira" or just "confluence", use the query:

```
"atlassian jira" || confluence
```

or

```
"atlassian jira" OR confluence
```

AND

The AND operator matches documents where both terms exist anywhere in the text of a single document. This is equivalent to an intersection using sets. The symbol && can be used in place of the word AND.

To search for documents that contain "atlassian jira" and "issue tracking", use the query:

```
"atlassian jira" AND "issue tracking"
```

Required term: +

The "+" or required operator requires that the term after the "+" symbol exists somewhere in the field of a single document.

To search for documents that must contain "jira" and may contain "atlassian", use the query:

```
+jira atlassian
```

NOT

The NOT operator excludes documents that contain the term after NOT. This is equivalent to a difference using sets. The symbol ! can be used in place of the word NOT.

To search for documents that contain "atlassian jira" but not "japan", use the query:

```
"atlassian jira" NOT "japan"
```

Note: The NOT operator cannot be used with just one term. For example, the following search will return no results:

```
NOT "atlassian jira"
```

Usage of the **NOT** operator over multiple fields may return results that include the specified excluded term. This is due to the fact that the search query is executed over each field in turn, and the result set for each field is combined to form the final result set. Hence, an issue that matches the search query based on one field, but fails based on another field will be included in the search result set.

Excluded term: -

The "-" or prohibit operator excludes documents that contain the term after the "-" symbol.

To search for documents that contain "atlassian jira" but not "japan", use the query:

```
"atlassian jira" -japan
```

Grouping

JIRA supports using parentheses to group clauses to form sub queries. This can be very useful if you want to control the boolean logic for a query.

To search for bugs and either atlassian or jira, use the query:

```
bugs AND (atlassian OR jira)
```

This eliminates any confusion and makes sure that bugs must exist, and either term atlassian or jira may exist.

Do not use the grouping character '(' at the start of a search query, as this will result in an error. For example, "(atlassian OR jira) AND bugs" will not work.

Escaping special characters: \ or \\

JIRA supports the ability to search issues for special characters by escaping them in your query syntax. The current list of such characters is:

```
+ - & | ! ( ) { } [ ] ^ ~ * ? \ :
```

To escape these characters, type a backslash character '\' before the special character (or if using advanced searching, type two backslashes '\\\' before the special character).

For example, to search for (1+1) in either a basic or quick search, use the query:

```
\(1\+1\)
```

and to search for [example] in the summary of an advanced search (in JIRA Query Language or JQL), use the query:

```
summary ~ "\\\[example\\]"
```

Please note: If you are using advanced searching, see [Reserved characters](#) for more information about how these characters and others are escaped in JIRA Query Language.

Reserved words

To keep the search index size and search performance optimal in JIRA, the following English *reserved words* (also known as 'stop words') are ignored from the search index and hence, JIRA's text search features:

```
"a", "and", "are", "as", "at", "be", "but", "by", "for", "if", "in", "into",
"is", "it", "no", "not", "of", "on", "or", "s", "such", "t", "that", "the",
"their", "then", "there", "these", "they", "this", "to", "was", "will", "with"
```

Be aware that this can sometimes lead to unexpected results. For example, suppose one issue contains the text phrase "VSX will crash" and another issue contains the phrase "VSX will not crash". A text search for "VSX will crash" will return both of these issues. This is because the words `will` and `not` are part of the reserved words list.

i Your JIRA administrator can make JIRA index these reserved words (so that JIRA will find issues based on the presence of these words) by changing the **Indexing Language** to **Other** (under **Administration > System > General Configuration**).

Word stemming

Since JIRA cannot search for issues containing parts of words (see [below](#)), word 'stemming' allows you to retrieve issues from a search based on the 'root' (or 'stem') forms of words instead of requiring an exact match with specific forms of these words. The number of issues retrieved from a search based on a stemmed word is typically larger, since any other issues containing words that are stemmed back to the same root will also be retrieved in the search results.

For example, if you search for issues using the query term 'customize' on the Summary field, JIRA stems this word to its root form 'custom', and will retrieve all issues whose Summary field also contains any word that can be stemmed back to 'custom'. Hence, the following query:

```
summary ~ "customize"
```

will retrieve issues whose Summary field contains the following words:

- customized
- customizing
- customs
- customer
- etc.

i Please Note:

- Your JIRA administrator can disable word stemming (so that JIRA will find issues based on exact matches with words) by changing the **Indexing Language** to **Other** (under **Administration > System > General Configuration**).
- Word stemming applies to *all* JIRA fields (as well as text fields).
- When JIRA indexes its fields, any words that are 'stemmed' are stored in JIRA's search index in root form only.

Limitations

Please note that the following limitations apply to JIRA's search:

Whole words only

JIRA cannot search for issues containing parts of words but on whole words only. The exception to this are words which are [stemmed](#).

This limitation can also be overcome using [fuzzy searches](#).

Next steps

Read the following related topics:

- [Searching for issues](#)
- [Quick searching](#)
- [Basic searching](#)
- [Advanced searching](#)

Saving your search as a filter

JIRA's powerful [issue search](#) functionality is enhanced by the ability to save searches, called *filters* in JIRA, for later use. You can do the following with JIRA filters:

- Share and email search results with your colleagues, as well as people outside of your organization
- Create lists of [favorite filters](#)
- Have search results [emailed to you](#) according to your preferred schedule
- View and export the search results in various formats (RSS, Excel, etc)
- Display the search results in a report format
- Display the search results in a [dashboard gadget](#)

On this page:

- [Saving a search as a filter](#)
- [Running a filter](#)
- [Managing your existing filters](#)
- [Managing other user's shared filters](#)
- [Next steps](#)

Screenshot: Issue filter results in detail view (click to view full size image)

Red Nerd [Save as](#) [Details](#) ★

Project: All ▾ Type: All ▾ Status: All ▾ Assignee: All ▾ **Red Nerd** More ▾ [Advanced](#)

Order by ▾

- [ANGRY-304 Red Angry Nerd is scary](#)
- [ANGRY-299 My red nerd is not working properly](#)
- [ANGRY-158 give the red nerd a yellow hat](#)
- [ANGRY-13 I don't like the red nerd](#)
- [ANGRY-306 Red Nerd should have his mouth o...](#)
- [ANGRY-35 Bug on Atlassian - Angry Nerds - R...](#)
- [ANGRY-79 Fix nerd's hair-styling](#)
- [ANGRY-70 Some graphical glitches](#)

Angry Nerds / ANGRY-304 ▾ Display ▾ 1 of 8 ▾

[Edit](#) [Comment](#) [Assign](#) [More ▾](#) [Resolve Issue](#) [Close Issue](#) [Share](#)

Details

Type:	Bug	Status:	Open (View Workflow)
Priority:	Minor	Resolution:	Unresolved
Component/s:	None	Fix Version/s:	None
Labels:	None		
Monkey:	Cheeky Monkey		

Description
[Click to add description](#)

Attachments +

People

Assignee:	Unassigned
Reporter:	Bartosz Galz
Votes:	0 Vote for this issue
Watchers:	1 Start watching this issue

Dates

Created:	21/Mar/13 3:37 PM
Updated:	21/Mar/13 3:39 PM
Scheduled:	21/Mar/13
Deployment Date:	

Collaborators +

Agile
[View on Board](#)

Saving a search as a filter

1. Define and run your search.
2. Click the **Save as** link above the search results. The **Save Filter** dialog is displayed.
3. Enter a name for the new filter and click **Submit**. Your filter is created.

Your new filter will be added to your favorite filters and shared, according to the sharing preference in your user profile. If you haven't specified a preference, then the global default will be applied, which is 'Private' unless changed by your JIRA administrator.

Running a filter

1. Choose **Issues > Search for issues**.
2. Choose any filter from the list on the left:
 - System filter — **My Open Issues, Reported by Me, Recently Viewed, All Issues**
 - Favorite filters (listed alphabetically)
 - **Find filters** lets you search for any filter that's been shared, which you can then subscribe to (adding it to your **Favorite Filters**).
3. After selecting a filter, the search results are displayed. The search criteria for the filter are also displayed and can be changed.
*Note, if you run the **Recently Viewed** system filter, this will switch you to the advanced search, as the basic search cannot represent the ORDER BY clause in this filter.*

Managing your existing filters

Click **Issues > Manage filters** to manage your filters.

The screenshot shows the 'Manage Filters' page with the 'My' tab selected. On the left, there's a sidebar with 'Favourite', 'My', 'Popular', and 'Search' tabs. The main area is titled 'My Filters' and contains a list of saved filters. Each filter entry includes the filter name, 'Shared With' status (either 'Shared with all users' or 'Private filter'), and 'Subscriptions' status ('None - Subscribe'). There are also edit and delete icons next to each filter entry. At the bottom of the page, there are links for 'Powered by Atlassian | Terms of Use | Answers'.

Name	Shared With	Subscriptions
★ Browse Project Epics for ADG	• Shared with all users	None - Subscribe
★ Ignite docs	• Private filter	None - Subscribe
★ JIRA OnDemand Feature Tracking	• Shared with all users	None - Subscribe
★ Kickass docs	• Private filter	None - Subscribe
★ PDL Needs Docs	• Shared with all users	None - Subscribe
★ PDL-Page	• Private filter	None - Subscribe
★ Red Nerd	• Private filter	None - Subscribe
★ The Red Nerds need modifications	• Shared with all users	None - Subscribe

The **Manage Filters** page allows you to view and configure filters that you have created, as well as work with filters that other users have shared with you. See the following topics for more information:

- Searching for a filter
- Updating a filter
- Deleting a filter
- Cloning a filter
- Adding a filter as a favorite
- Sharing a filter
- Defining a filter-specific column order
- Subscribing to a filter

Searching for a filter

You can find and run any filters that you have created or that have been shared by other users.

1. Click the **Search** tab on the 'Manage Filters' page.
2. Enter your search criteria and click **Search** to run the search.
3. Your search results are displayed on the same page. Click the name of any issue filter to run it.

*Tip: If the filter has been added as a favorite by many users, you may also be able locate it on the **Popular** tab of the **Manage Filters** page.*

Updating a filter

You can update the name, description, sharing, favorite of any filters that you have created. If you want to edit a filter that was shared with you, either [clone](#) (aka copy) the shared filter, or ask your JIRA administrator to [change the filter's ownership](#).

Update the filter's details:

1. Click the **My** tab on the 'Manage Filters' page.
2. Locate the filter you wish to update, click the **cog icon > Edit**.
3. The **Edit Current Filter** page displays, where you can update the filter details as required.
4. Click **Save** to save your changes.

Update the filter's search criteria:

1. Click the **My** tab on the 'Manage Filters' page.
2. Locate the filter you want to update and run it.
3. Update the search criteria as desired, and rerun the query to ensure the update is valid. You will see the word *Edited* displayed next to your filter name.
4. Click **Save** to overwrite the current filter with the updated search criteria. If you want discard your changes instead, click the arrow next to the save button, and select **Discard changes**.

Deleting a filter

1. Click the **My** tab on the 'Manage Filters' page.
2. Locate the filter you wish to delete, click the **cog icon > Delete**.

Cloning a filter

You can clone any filter – which is just a way of making a copy that you own – that was either created by you or shared with you.

1. Locate the filter you wish to clone and run it.
2. Update the search criteria as desired. Click the arrow next to the **Save** button, and select **Save > Save as** to create a new filter from the existing filter.

Adding a filter as a favorite

Filters that you've created or that have been shared by others can be added to your favorite filters. Favorite filters are listed in the menu under **Issues > Filters**, and in the left panel of the issue navigator.

1. Locate the filter you wish to add as a favorite.
2. Click the star icon next to the filter name to add it to your favorites.

Sharing a filter

Filters that you have created can be shared with other users via user groups, projects, and project roles. They can also be shared globally. Any filter that is shared is visible to users who have the 'JIRA Administrators' global permission. See [Managing other users' shared filters](#) below.

1. Click the **My** tab on the 'Manage Filters' page.
2. Locate the filter you wish to share, click the **cog icon > Edit**.
3. Update the **Add Shares** field by selecting the group, project, or project role that you want to share the filter with, and clicking **Add**. Note that you can only share filters with groups/roles of which you are a member.
 - ▼ [Why can't I see the filter's sharing configuration?](#)
You need the Create Shared Object global permission to configure sharing for a filter. Contact your JIRA administrator to obtain this permission.
4. Click **Save** to save your changes.

*Tip: You can also share your filter by running it, then clicking **Details > Edit Permissions**.*

Defining a filter-specific column order

You can add a defined column order to a saved filter, which displays the filter results according to the saved column order. Otherwise, the results are displayed according to your personal column order (if you have set this) or the system default.

*Tip: To display your configured column order in a filter subscription, select 'HTML' for the 'Outgoing email format' in your **User Profile**. If you receive text emails from JIRA, you won't be able to see your configured column order.*

To add a column layout to a saved filter:

1. Click the **My** tab on the 'Manage Filters' page.
2. Locate the filter you wish to update; click the filter's name to display the results. Be sure you are viewing the filter in the **List** view so that you see the columns.
3. Configure the column order as desired by clicking on the column name and dragging it to the new position. Your changes are saved and will be displayed the next time you view this filter.

To remove a filter's saved column layout:

1. Click the **My** tab on the 'Manage Filters' page.
2. Locate the filter you wish to update; click the filter's name to display the results. Be sure you are viewing the filter in the **List** view so that you see the columns.
3. Click the **Columns** option on the top right of the displayed columns, and select **Restore Defaults** in the displayed window.

Exporting column ordered issues

When the results of a saved filter are exported to Excel, the column order and choice of columns are those that were saved with the filter. Even if a user has configured a personal column order for the results on the screen, the **saved configuration** is used for the Excel export. To export using your own configuration, save a copy of the filter along with your configuration, and then export the results to Excel.

Subscribing to a filter

See [Working with search results](#).

Managing other user's shared filters

A **shared filter** is a filter whose creator has shared that filter with other users. Refer to [Sharing a filter](#) above for details. 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.

If you have the **JIRA Administrators** global permission, you can manage shared filters that were created by other users. For instructions, see [Managing shared filters](#).

Next steps

Read the following related topics:

- [Searching for issues](#)
- [Basic searching](#)
- [Advanced searching](#)
- [Working with search results](#)

Working with search results

Once you have run a search, your search results will be displayed in the issue navigator. You may want to triage the entire list of issues or maybe be looking for just one. This page will show you what you can do with your search results, from changing what you see in the issue navigator to modifying the issues.

On this page:

- Changing your view of the search results
- Working with individual issues
- Sharing your search results
- Displaying your search results in Confluence
- Displaying your search results as a chart
- Exporting your search results
- Printable views
- Subscribing to your search results
- Bulk modifying issues in your search results
- Next steps

The following screenshot provides an overview of the key features of the issue navigator.

Screenshot: Issue navigator (Detail view)

The screenshot shows a JIRA search interface. On the left, there's a filter panel with sections like 'FILTERS', 'My Open Issues', 'FAVORITE FILTERS', and 'All open issues for ...'. A note says: 'Click <> to collapse the filter panel so you can have more space in the detail view.' Another note says: 'Select an issue from this panel to see the details in the detail view window.' Below the filter panel, a note says: 'Select a filter to see all the matching issues in the panel to the immediate right.'

The main area displays search results for 'All open issues for Teams in Space'. One result is selected: 'Initial draft for review' (TIS-127). The right side shows the 'Detail' view for this issue, including fields like Type (Sub-task), Status (CLOSED), Priority (Major), and Resolution (Unresolved). It also shows the 'People' assigned to the issue (Harvey Jennings) and the 'Activity' log.

A blue arrow points from the right side of the detail view towards the top right of the screen, with the text: 'Click to switch between the detail view and list view'.

On the far right, another note says: 'Check out all the details about the selected issue in this detail view'.

Changing your view of the search results

List view or Detail view	<p>Click the  dropdown to switch between List view and Detail view for your search results.</p> <ul style="list-style-type: none"> List view: Shows your search results as a list of issues. This view is easiest to scan and is best when you only need to know a few details about each issue. Detail view: Shows your search results as a list of issues, with the right panel showing the details of the currently selected issue. This view is best when you need more information about the individual issues, or you want to quickly edit issues as you go (via inline edit for certain fields).
Change the sort order	<p>Click the column name. If you click the same column name more than once, the sort order will switch between ascending and descending. Note:</p> <ul style="list-style-type: none"> You cannot sort by the 'Images' column nor the sub-task aggregate columns (i.e. all columns beginning with ''). If you sort the search results for an advanced search, an 'ORDER BY' clause will be added/updated for your JQL query to reflect the order of issues in your search results.

<p>Columns</p> <p>—</p> <p>show/hide and move</p>	<p>You can create different column configurations for yourself and for specific filters. To switch between different column configurations, click Columns and select one of the following tabs:</p> <ul style="list-style-type: none"> • My Defaults: This is your default column configuration for search results. • Filter: This is enabled if you are viewing the search results for a filter. It will override your default column configuration. • System (shows if you are a JIRA administrator): This is the column configuration that applies to all users. It will be overridden by a user's default column configuration and filter-specific column configurations. <p>You can also modify any of these configurations. Make sure you have switched the desired configuration, then do the following:</p> <ul style="list-style-type: none"> • Show/hide columns: Click Columns, choose the desired columns, then click Done. • Move a column: Click the column name and drag it to the desired position. <p>▼ Why can't I add a column to my column configuration?</p> <p>If you cannot find a column, please make sure that you haven't run into any of the following restrictions:</p> <ul style="list-style-type: none"> • 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.
---	---

Working with individual issues

You can action individual issues in your search results, directly from the issue navigator. Note that the list of issues will remain constant even if you change an issue, so that it doesn't meet the original search criteria. The advantage of this is that you have a constant set of search results that you can work from when triaging issues.

<p>View an issue</p>	<p>Click the key or summary of the issue.</p> <ul style="list-style-type: none"> • If you are in List view, you will be redirected to the issue (leaving the search results page). • If you are in Detail view, the issue details will display in the right panel.
<p>Action an issue</p>	<p>To action an issue (e.g. edit it, transition it, log work on it, etc):</p> <ul style="list-style-type: none"> • If you are in List view, click the cog icon and select from the options. • If you are in Detail view, select the issue and action it via the details panel. <p>You can also select an issue and action it via keyboard shortcuts in either views. <i>Tip: use the 'J' and 'K' keys to select the previous/next issue in the issue navigator.</i></p>

Sharing your search results

Click **Share** in the issue navigator to email a link to a search result or shared filter.

- Recipients will receive an email with a link to the search result and the content of the **Note** field (if specified). The subject of the email will state that you (using your username) shared the issue.
- If you share the results of a filter, rather than an ad-hoc search, recipients will receive a link to the filter. Note, if the recipient does not have permission to view the filter, they will receive a link to the search results instead.

Displaying your search results in Confluence

If your JIRA applications are connected to Confluence, you can display your search results on a Confluence

page using the JIRA issues macro. For instructions, see [JIRA issues macro](#).

Displaying your search results as a chart

Click **Export > Dashboard charts**. Choose the desired chart from the dialog that is displayed, then click **Save to Dashboard**.

The chart will be added to your dashboard. For more information on what each chart shows, see [Reporting](#).

Exporting your search results

Excel	<p>Click Export > Excel (All fields) or Export > Excel (Current fields).</p> <ul style="list-style-type: none"> • Excel (All fields): this will create a spreadsheet column for every issue field (excluding comments). <i>Note: This will only show the custom fields that are available for all of the issues in the search results. For example, a field that is only available for one project when your search results has issues from multiple projects.</i> • Excel (Current fields): this will create a spreadsheet column for the issue fields that are currently displayed. <p>Note, large exports (e.g. many hundreds of issues) are not recommended. You can change the number of issues that are exported, by changing the value of the tempMax parameter in the URL.</p>
CSV	<p>Click Export > CSV (All fields) or Export > CSV (Current fields).</p> <p>The comma separated value (CSV) file will contain a header row with a value for every applicable issue field, comment and attachment in your search result.</p> <ul style="list-style-type: none"> • CSV (All fields): this will create a comma separated value for every issue field, comment and attachment. The header row may contain multiple values of "Comment" and/or "Attachment" if your issues have multiple comments and/or attachments. • CSV (Current fields): this will create a comma separated value for the issue fields that are currently displayed. <p>Note, large exports (e.g. many hundreds of issues) are not recommended. You can change the number of issues that are exported, by changing the value of the tempMax parameter in the URL.</p>
Word	<p>Click Export > Word.</p> <p>The export will include the Description, Comments, and all other issue data, not just the issue fields that are currently configured in your Issue Navigator. Note, large exports (e.g. hundreds of issues) are not recommended.</p>
XML	<p>Click Export > XML.</p> <p>You can use the URL of the XML view in a Confluence JIRA issues macro. However, you can also use the JQL or the URL of the issue search, which are easier to get.</p> <p>To restrict which issue fields are returned in the XML export, specify the <code>field</code> parameter in your URL. For example, to include only the Issue key and Summary, add <code>&field=key&field=summary</code> to the URL. If the <code>field</code> parameter is not specified, the XML output will include <i>all</i> the issue fields. Otherwise, if one or more <code>field</code> parameters are specified, the XML output will contain only the Issue key plus your chosen field(s). See the "List of fields for field parameter" below.</p>

List of fields for field parameter (XML exports):

▼ [Show me...](#)

Value	Sample XML output
<code>title</code>	<pre><title>[TEST-4] This is a test</title></pre>

link	<link> https://extranet.atlassian.com:443/j
project (or pid)	<project id="10330" key="TST">Test</project>
description	<description>This is a detailed description</description>
environment	<environment>Sydney network</environment>
key	<key id="22574">TEST-4</key>
summary	<summary>This is a test</summary>
type (or issuetype)	<type id="3" iconUrl="https://extranet.atlan...>TEST-4</type>
parent	<parent id="22620">TEST-5</parent>
priority	<priority id="4" iconUrl="https://extranet.atlassia...>TEST-4</priority>
status	<status id="5" iconUrl="https://extranet.atlassia...>TEST-4</status>
resolution	<resolution id="1">Fixed</resolution>

labels	<pre><labels> <label>focus</label> <labels></pre>
assignee	<pre><assignee username="jsmith">John Smith</assignee></pre>
reporter	<pre><assignee username="jsmith">John Smith</assignee></pre>
security	<pre><security id="10021">Private</security></pre>
created	<pre><created>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)</created></pre>
updated	<pre><updated>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)</updated></pre>
resolved (or resolutiondate)	<pre><resolved>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)</resolved></pre>
due (or duedate)	<pre><due>Mon, 1 Sep 2008 17:30:03 -0500 (CDT)</due></pre>
version (or versions)	<pre><version>2.4.7</version></pre>
fixfor (or fixVersions)	<pre><fixVersion>2.6</fixVersion></pre>
component (or components)	<pre><component>Documentation</component></pre>
votes	<pre><votes>1</votes></pre>

comments (or comment)	<pre><comments> <comment id="39270" author="jsmith" created="Mon, 9 Feb 2009 13:32:58 -0600 (CST) too familiar</comment> <comment id="39273" author="jbrown" created="Tue, 10 Feb 2009 00:30:11 -0600 (CST) too</comment> </comments></pre>
attachments (or attachment)	<pre><attachments> <attachment id="30318" name="Issue Navigator Help.pdf" size="1024" type="application/pdf" created="Mon, 9 Feb 2009 13:32:58 -0600 (CST) too</attachment> <attachment id="30323" name="Windows XP Help.pdf" size="1024" type="application/pdf" created="Tue, 10 Feb 2009 00:30:11 -0600 (CST) too</attachment> </attachments></pre>
timeoriginalestimate	<pre><timeoriginalestimate seconds="600">10 minutes</timeoriginalestimate></pre>
timeestimate	<pre><timeestimate seconds="300">5 minutes</timeestimate></pre>
timespent	<pre><timespent seconds="300">5 minutes</timespent></pre>
aggregatetimeoriginalestimate	<pre><aggregatetimeoriginalestimate seconds="3600">1 hour</aggregatetimeoriginalestimate></pre>
aggregatetimeestimate	<pre><aggregatetimeestimate remainingestimate seconds="18000">5 hours</aggregatetimeestimate></pre>
aggregatetimespent	<pre><aggregatetimespent seconds="18000">5 hours</aggregatetimespent></pre>
timetracking	<pre><timeoriginalestimate seconds="600">10 minutes</timeoriginalestimate> <timeestimate seconds="300">5 minutes</timeestimate> <timespent seconds="300">5 minutes</timespent> <aggregatetimeoriginalestimate seconds="3600">1 hour</aggregatetimeoriginalestimate> <aggregatetimeestimate remainingestimate seconds="18000">5 hours</aggregatetimeestimate> <aggregatetimespent seconds="18000">5 hours</aggregatetimespent></pre>

issuelinks	<pre><issuelinks> <issuealinktype id="10020"> <name>Duplicate</name> <inwardlinks description="is dupli< <issuelink> <issuekey id="22477">INTSY:< </issuelink> </inwardlinks> </issuealinktype> </issuelinks></pre>
subtasks (or subtask)	<pre><subtasks> <subtask id="22623">TEST-8</subtask> </subtasks></pre>
customfield_xxxxx	<pre><customfields> <customfield id="customfield_10112" key="com.atlassian.jira.plugin.system.cust< <customfieldname>Department</customf< <customfieldvalues> <customfieldvalue>Administratio< </customfieldvalues> </customfield> </customfields></pre>
allcustom	<pre><customfields> <customfield id="customfield_10112" key="com.atlassian.jira.plugin.system.cust< <customfieldname>Department</customf< <customfieldvalues> <customfieldvalue>Administratio< </customfieldvalues> </customfield> <customfield id="customfield_10111" key="com.atlassian.jira.plugin.system.cust< <customfieldname>Expenditure Type< <customfieldvalues> <customfieldvalue>Operating</ci< </customfieldvalues> </customfield> </customfields></pre>

Printable views

Printable	Click Export > Printable . Creates a view of your search results in your browser that can be printed 'Landscape'. This view only contains issue Type, Key, Summary, Assignee, Reporter, Priority, Status, Resolution, Created date, Updated date, and Due date.
Full content	Click Export > Full content . Creates a view of your search results in your browser that can be printed. This view contains all issue fields, comments, and a list of attachments (there is no preview) for every issue returned by your search.

Subscribing to your search results

A subscription provides you with a periodic notification for all issues returned by the search. If you want to be notified when a particular issue changes, you should watch the issue instead.

Email	Your search must be saved as a filter, if you want to create an email subscription for it. You can create a subscription of any frequency for yourself and/or other users. Note, only the first 200 results of a filter are sent. 1. Run the filter that you want to subscribe to, then click Details (next to filter name). 2. Fill in the 'Filter Subscription form' and click Subscribe . More information: <ul style="list-style-type: none">• If you choose 'Advanced' for your Schedule, see this page for help on constructing Cron expressions.• If you want to specify a group as a recipient:<ul style="list-style-type: none">• You must have the 'Manage Group Filter Subscriptions' global permission.• Be aware that the emailed filter results will be specific to each recipient. For example, if the filter uses the <code>currentUser()</code> function, the search results will be evaluated with the recipient as the current user. This does not apply to distribution lists (group email aliases).• Be careful about sharing a subscription with a group with many members, as it can take a long time to generate the emails to be sent, since the search needs to be executed for each user (as per the previous point).
RSS	Click Export > RSS (Issues) or Export > RSS (Comments) . The URL of the page that shows can be used in your feed reader. Tips: <ul style="list-style-type: none">• You can change the number of issues that are returned, by changing the value of the <code>tempMax</code> parameter in the URL.• If you only want to receive current comments in an RSS feed, use the Date Updated field when doing a search. For example, to only receive comments created in the last week, add the Date Update field and set it to updated within the last 1 week.• You may need to log into your JIRA applications to view restricted data in your feed. If so, you can add <code>os_authType=basic</code> to the feed URL (e.g. <code>http://mycompany.com/anypage?os_authType=basic</code>) to show a login dialog when viewing the feed.

Bulk modifying issues in your search results

Bulk operations let you action multiple issues at once. These actions include transitioning issues, deleting issues, moving issues, and watching/unwatching issues.

Click **Tools > Bulk Change: all <N> issue(s)** and follow the 'Bulk Operation' wizard.

For more information, see [Editing multiple issues at the same time](#).

Next steps

Read the following related topics:

- [Searching for issues](#)
- [Constructing cron expressions for a filter subscription](#)

Constructing cron expressions for a filter subscription

This page describes how to construct a cron expression. Cron expressions can be used when creating a subscription to a filter, as described in [Working with search results](#).

A cron expression gives you more control over the frequency, compared to the default schedules. For example, you could define a cron expression to notify you at 8:15 am on the second Friday of every month.

Constructing a cron expression

A cron expression is a string of fields separated by spaces. The following table displays the fields of a cron expression, *in the order that they must be specified (from left to right)*:

	Second	Minute	Hour	Day-of-month	Month	Day-of-week	Year (optional)
Allowed values	0–59	0–59	0–23	1–31	1–12 or JAN–DEC	1–7 or SUN–SAT	1970–2099
Allowed special characters	, - * /	, - * /	, - * /	, - * / ? L W C	, - * /	, - * / ? L C #	, - * /

Note, cron expressions are not case-sensitive.

Here is an example:

0 15 8 ? JAN MON 2014

This literally translates to 0 second, 15 minute, 8 hour, any day of the month, January, 2014.

In plain English, this represents 8:15am on every Monday during January of 2014. Note, the ? character means "no particular value". In this example, we've set the Day-of-month to no particular value. We don't need to specify it, as we've specified a Day-of-week value. Read more about special characters in the next section.

More examples of cron expressions are explained in the [Examples](#) section at the bottom of this page.

Special characters

Special character	Usage
,	Specifies a list of values. For example, in the Day-of-week field, 'MON,WED,FRI' means 'every Monday, Wednesday, and Friday'.
-	Specifies a range of values. For example, in the Day-of-week field, 'MON-FRI' means 'every Monday, Tuesday, Wednesday, Thursday and Friday'.
*	Specifies all possible values. For example, in the Hour field, '*' means 'every hour of the day'.
/	Specifies increments to the given value. For example, in the Minute field, '0/15' means 'every 15 minutes during the hour, starting at minute zero'.
?	Specifies no particular value. This is useful when you need to specify a value for one of the two fields Day-of-month or Day-of-week , but not the other.

L	Specifies the last possible value; this has different meanings depending on context. In the Day-of-week field, 'L' on its own means 'the last day of every week' (i.e. 'every Saturday'), or if used after another value, means 'the last xxx day of the month' (e.g. 'SATL' and '7L' both mean 'the last Saturday of the month'). In the Day-of-month field, 'L' on its own means 'the last day of the month', or 'LW' means 'the last weekday of the month'.
W	Specifies the weekday (Monday-Friday) nearest the given day of the month. For example, '1W' means 'the nearest weekday to the 1st of the month' (note that if the 1st is a Saturday, the email will be sent on the nearest weekday <i>within the same month</i> , i.e. on Monday 3rd). 'W' can only be used when the day-of-month is a single day, not a range or list of days.
#	Specifies the nth occurrence of a given day of the week. For example, 'TUES#2' (or '3#2') means 'the second Tuesday of the month'.

Examples

0 15 8 ? * *	Every day at 8:15 pm.
0 15 8 * * ?	Every day at 8:15 am.
0 * 14 * * ?	Every minute starting at 2:00 pm and ending at 2:59 pm, every day.
0 0/5 14 * * ?	Every 5 minutes starting at 2:00 pm and ending at 2:55 pm, every day.
0 0/5 14,18 * * ?	Every 5 minutes starting at 2:00 pm and ending at 2:55 pm, AND every 5 minutes starting at 6:00 pm and ending at 6:55 pm, every day.
0 0-5 14 * * ?	Every minute starting at 2:00 pm and ending at 2:05 pm, every day.
0 0/10 * * * ? *	Every 10 minutes, forever.
0 10,44 14 ? 3 WED	2:10 pm and 2:44 pm every Wednesday in the month of March.
0 15 8 ? * MON-FRI	8:15 am every Monday, Tuesday, Wednesday, Thursday, and Friday.
0 15 8 15 * ?	8:15 am on the 15th day of every month.
0 15 8 L * ?	8:15 am on the last day of every month.
0 15 8 LW * ?	8:15 am on the last weekday of every month.
0 15 8 ? * 6L	8:15 am on the last Friday of every month.
0 15 8 ? * 6#2	8:15 am on the second Friday of every month.
0 15 8 ? * 6#2 2007-2009	8:15 am on the second Friday of every month during the years 2007, 2008, and 2009.

Working with issues

Need help working with issues? In JIRA Software, you can create issues, estimate issues, even manage your code via issues. On this page, you'll find a quick overview for everything that you can do with an issue, such as creating issues, updating issues with estimates, creating branches in your source repository that reference the relevant issues in JIRA Software, viewing development work for issues, resolving issues, as well as links to

On this page:

- [What is an issue?](#)
- [Next steps](#)

pages with more detail. This page introduces you to the concept of an issue. You can then learn more about creating, editing, and collaborating issues in the Next steps section.

What is an issue?

Different organizations use JIRA applications to track different kinds of issues, which can represent anything from a software bug, a project task, to a leave request form.

Issues are the building blocks of any JIRA project. An issue could represent a story, a bug, a task, etc in your project. This is what an issue looks like in JIRA Software:

On a board (Scrum project — backlog)

The screenshot shows a JIRA Scrum backlog board. At the top, there's a header with filters: Product, UI, Only My Issues, and Recently Updated. Below the header, a table lists issues in Sprint 6. One issue is selected: TIS-109 Afterburner revision VI script. The right side of the screen displays the 'Issue detail view' for this selected issue, showing fields like Status (OPEN), Components (None), Labels (None), Affects Versions (1.8), Fix Versions (None), and Epic (Afterburner Plus).

Full view (e.g. via search results)

The screenshot shows the full view of an issue titled 'Draft network plan for Mars Office'. It includes sections for Details (Type: Story, Priority: Major, Status: CLOSED, Resolution: Unresolved), Description (a note about reviewing network architecture), Sub-Tasks (one task listed), Activity (a history of changes), and a Development panel. The Development panel shows 2 branches, 2 merges, and an active sprint from 07/04/15 to 03/05/15.

Note, the **issue detail view** may show different fields to the full view if it has been customized by your administrator. To see the full view, open the issue in a new tab/window (e.g. click the issue key with your middle mouse button).

Most information on an issue is self-explanatory. Here are a few things to be aware of though:

Why can't I see the development panel?

This panel only shows if JIRA is connected to your development tools. You will be able to see related commits, builds, etc, to help you evaluate the development status of your issue. See [Viewing the development information for an issue](#).

Why can't I see the Time Tracking panel?

This panel will be shown on the issue, if your administrator has set Time Tracking to 'Remaining Estimate' and 'Time Spent'. If an issue has sub-tasks, the Remaining estimate for the issue plus sub-tasks is rolled up into the parent issue.

What activity is shown in the History and Activity tabs?

The **History** tab of an issue records the following information: creator of the issue (this may be the same as the reporter, but can be distinct), changes to an issue field, attachment of a file, deletion of a comment, deletion of a worklog, creation or deletion of an issue link.

The **Activity** tab has the same information, plus additional information, such as comments. However, this may load more slowly, especially if there has been a lot of activity on the issue.

Next steps

Check out the following pages to reach issue ninja status:

- Creating issues and sub-tasks
- Attaching files and screenshots to issues
- Editing and collaborating on issues
- Logging work on issues

Creating issues and sub-tasks

The building blocks of any project are issues. Issues act as the packets of work that travel through their respective workflows within their projects, until the work is completed. An issue may also have sub-tasks that can be assigned and tracked individually, as well as issue level security to restrict the issue to select members of your team. On this page, you'll learn more about creating and converting issues and sub-tasks, and setting issue level security. If you are looking to import multiple issues (and sub-tasks) using a CSV file, you can find the import process explained in more detail [here](#).

Before you begin

You need the **Create Issue** project permission for the issue's relevant project.

On this page:

- Before you begin
- Creating an issue
- Cloning an issue
- Creating a sub-task
- Converting a sub-task to an issue
- Converting an issue to a sub-task
- Restricting access to an issue

Creating an issue

1. Click **Create** at the top of the screen to open the **Create Issue** dialog box.
 2. Select the relevant **Project** and **Issue Type** in the **Create Issue** dialog box.
 3. Type a **Summary** for the issue and complete any appropriate fields — at least the required ones that are marked by an asterisk.
- If you want to access fields that are not shown in this dialog box, or you want to hide existing fields:
- a. Click the **Configure Fields** button at the top right of the screen.
 - b. Click **Custom** and select the fields you want to show or hide by selecting or clearing the relevant check boxes respectively, or click **All** to show all fields.
- When you next create an issue, these selected fields will be displayed.
4. Optional: To create a series of similar issues – with the same **Project** and **Issue Type** – select the **Create another** checkbox at the bottom of the dialog. Depending on your configuration and the values you may have specified when creating previous issues, some of the fields in the new Create Issue dialog box may be pre-populated. Make sure you check they're all correct before creating the next issue.
 5. When you are satisfied with the content of your issue, click the **Create** button.

Tips:

- Your issue will be created at the top of the backlog, unless;
 - you have an issue selected in the backlog — your issue will be created right below the selected issue
 - you have specified a sprint when creating the issue — your issue will be created at the bottom of the sprint
- If you are using a Scrum board, you can quickly create issues using the inline issue create in the Backlog (backlog and future sprints only). Just click **+ Create issue**.

The screenshot shows a JIRA backlog interface. At the top, it says "Backlog 1 of 9 issues visible Clear all filters". Below this is a list of issues. The first issue is partially visible: "SSP-5 As a team, I'd like to commit to a set of stories to be completed in a sprint". At the bottom of the backlog list, there is a button labeled "+ Create issue".

Note, if your board's filter specifies more than one project, you will still need to complete the full 'Create issue' dialog.

- If you are using epics, you may want to click **Configure Fields** and add the **Epic Link** field to the screen, so that you can add issues to epics as you create them.
- Note, you can also create an issue that automatically belongs to a particular epic. See [Working with](#)

epics.

Cloning an issue

Cloning an issue lets you quickly create a duplicate of an issue within the same project. The cloned issue contains most of the same details stored in the original issue — e.g. Summary, Affects Versions, Components, etc. Other details are not cloned — e.g. Work Log, Comments, Issue history, and Links to Confluence pages. The issue status also returns to the first step of the corresponding workflow, and the resolutions are cleared. The cloned issue can be linked to the original issue, but does not have to be.

1. Open the issue you wish to clone.
2. Select **More > Clone**. The **Clone Issue** screen will appear.
3. You can edit the clone issue's **Summary** if you want.
4. If applicable to the issue you are cloning, you can also select from these options:
 - **Clone sub-tasks** to copy existing sub-tasks
 - **Clone attachments** to add any existing attachments
 - **Clone links** to add any existing linked issues
 - **Clone sprint values** to copy across the issue's current and closed sprint values
5. Click **Create**.

Creating a sub-task

A sub-task can be created for an issue to either split the issue into smaller chunks, or to allow various aspects of an issue to be assigned to different people. An issue cannot be resolved until all its sub-tasks are completed and resolved. If you find a sub-task is holding up the resolution of an issue, you can convert the sub-task to an issue, to allow it to be worked on independently. If you find an issue is really just a sub-task of a bigger issue, you can also convert an issue to a sub-task.

You can only create sub-tasks if your administrator has enabled sub-tasks, and has added the sub-task issue type to the project's issue type scheme.

1. Navigate to the issue you would like to be the parent issue of the sub-task you are about to create.
2. Select **More > Create Sub-Task**. You will see the **Create sub-task** screen.
3. Fill in the details as needed, and then click **Create** at the bottom of the page.

Note that when you create a sub-task, the following values are inherited from the parent task:

- project
- issue security level
- sprint value, if any (only for JIRA Software issues)

Tip: You can customize the **Create sub-task** screen to show fields you use most often. To do this, click **Configure Fields** at the top right corner of the dialog, and use the **All** and **Custom** links to switch between the default screen and your custom settings. Your changes are saved for future use.

Converting a sub-task to an issue

1. Navigate to the sub-task issue you would like convert.
2. Select **More > Convert to Issue**.
3. In the **Step 1. Select Issue Type** screen, select a new issue type (i.e. a standard issue type) and click **Next**.
4. If the sub-task's current status is not an allowed status for the new issue type, the **Step 2. Select New Status** screen is displayed. Select a new status and click **Next**.
5. In the **Step 3. Update Fields** screen, you will be prompted to enter any additional fields if they are required. Otherwise, you will see the message 'All fields will be updated automatically'. Click **Next**.
6. The **Step 4. Confirmation** screen is displayed. If you are satisfied with the new details for the issue, click **Finish**.
7. The issue will be displayed. You will see that it is no longer a sub-task, that is, there is no longer a parent issue number displayed at the top of the screen.

Converting an issue to a sub-task

1. Navigate to the issue you would like to convert.
2. Select **More > Convert to Sub-Task**.
3. In the **Step 1. Select Parent Issue and Sub-Task Type** screen, type or select the appropriate parent issue type and the new issue type (i.e. a sub-task issue type). Click **Next**.
4. If the issue's current status is not an allowed status for the new issue type, the **Step 2. Select New Status** screen is displayed. Select a new status and click **Next**.
5. In the **Step 3. Update Fields** screen, you will be prompted to enter any additional fields if they are required. Otherwise, you will see the message 'All fields will be updated automatically'. Click **Next**.
6. The **Step 4. Confirmation** screen is displayed. If you are satisfied with the new details for the issue, click **Finish**.
7. The issue will be displayed. You will see that it is now a sub-task, that is, its parent's issue number is now displayed at the top of the screen.

Note: You will not be able to convert an issue to a sub-task if the issue has sub-tasks of its own. You first need to convert the issue's sub-tasks to standalone issues; you can then convert them to sub-tasks of another issue if you wish. Sub-tasks cannot be moved directly from one issue to another — you will need to convert them to standard issues, then to sub-tasks of their new parent issue.

Restricting access to an issue

When creating (or editing) an issue, you can restrict access to that issue to members of your team who are part of a chosen security level. To be able to set the security level for an issue, your administrator must add you to the appropriate issue security level, and also grant you the 'Set Issue Security' permission for the appropriate projects.

1. Create/edit the relevant issue.
 2. In the **Security Level** drop-down field, select the desired security level for the issue. You will only see the security levels you belong to.
 3. Save the issue. It is now only accessible to members of the specified security level.
- Users who are not members of this security level will not be able to access that issue, or see it in any filters, queries, or statistics.

Creating issues using the CSV importer

If you have the **Create Issue** project permission and the **Bulk Change** global permission for the relevant projects, you can create issues in bulk using a comma-separated value (CSV) file. CSV files are text files that represent tabulated data, and are supported by most systems that handle tabulated data, such as spreadsheets (MS Excel, Numbers) and databases.

The CSV importer allows you to import data from external systems that can export their data in a tabulated format. It also allows you to create your own CSV file to perform bulk issue creation and updates.

Your administrator has access to more import options designed specifically for other systems, such as Github, Fogbugz, and Bugzilla. If you are planning on importing from an external system a large amount of issues, administrators have access to advanced import functionalities by following: [Migrating from other issue trackers](#), including [Importing data from CSV](#).

On this page:

- Preparing your CSV file
- Running the CSV file import wizard
- Tips for importing CSV data into issue fields

There are two steps to using the CSV importer, and an optional third step:

1. Preparing your CSV file
2. Running the CSV import wizard
3. Saving your configuration for future use

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 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 your project's issues.

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, 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 issue fields

You can import multiple values into an issue 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 issue 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 issue fields upon import.

 Be aware that only a limited number of issue fields support multiple values. The CSV importer will not allow you to import aggregated data into issue fields that 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 instance *must* be able to access.

Importing issues into multiple projects

You can import issues from your CSV file into different 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 projects to which they will be imported.
-  The project name and key data is the *minimum project data* required for importing issues from a CSV file into specific 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** issue fields, respectively during the CSV file import wizard.

Importing work log entries

Your CSV file can contain work log 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](#)

).

Running the CSV file import wizard

Before you begin: If your JIRA installation has existing data, you should [back it up](#).

1. Select **Issues > Import Issues from CSV** to open the **Bulk Create Setup** page. (If you do not have the option **Import issues from CSV**, your JIRA Admin must update the JIRA Importers plugin to version 6.2.3 or above.)
2. On the **Setup** page, select your **CSV Source File**. 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 installation.
 - 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 ask you if you want to create a configuration file that you can use for subsequent CSV imports.
3. Click the **Next** button to proceed to the **Settings** step of the CSV file import wizard. Complete the required fields.
 - If your CSV file uses a different separator character other than a comma, specify that character in the **CSV Delimiter** field. If the separator is a 'Tab', this can be entered using the format '**\t**'.
4. Click the **Next** button to proceed to the **Map fields** step of the CSV file import wizard. Here, you can map the column headers of your CSV file to the fields in your selected project. If you want to select specific JIRA field values to map specific CSV values to, tick the checkbox for **Map field value**.
 - **Note:** You must map a CSV field to the issue's summary field. This ensures the issues created have a summary.
5. Click the **Next** button to proceed to the **Map values** step of the CSV file import wizard. On this step of the import wizard, you can select which specific CSV field values you want to map to which specific issue field value. For example, your issue types you may have a CSV field value of "Feature Request",

which you may want to map to the issue type field value "New Feature".

i 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'.
- If you are importing a username-based CSV field (e.g. **Reporter** or **Assignee**) and you do not select the **Map field value** checkbox for this field in the previous step of the CSV file import wizard, then the importer will automatically map imported usernames from the CSV file to (lowercase) JIRA usernames.

i Regardless of whether or not you select the **Map field value** checkbox, JIRA will automatically create usernames based on the data in your CSV file if they have not already been defined in JIRA.

6. 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.
7. If you're confident your import is correctly set up, click the **Begin Import** button. Your import will begin and once complete you will be informed of any errors. If you'd like to check your import first, click the **Validate** button and JIRA will validate your import and inform you of any expected errors or warnings. You can then go back and correct these before running your full import.

i Note:

- If you experience problems with the import (or you are curious), click the **download a detailed log** link to reveal detailed information about the CSV file import process.
- If you need to import another CSV file with the same (or similar) settings to what you used through this procedure, click the **save the configuration** link to download a CSV configuration file, which you can use at the first step of the CSV file import wizard.

Congratulations, you have successfully imported your CSV data into JIRA! If you have any questions or encounter any problems, please contact [Atlassian support](#).

Tips for importing CSV data into issue fields

Below are some helpful tips when importing data from your CSV file into specific issue fields:

Issue Field	Import Notes
Project	CSV data is imported on a per-project basis. You can either specify an existing 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.
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.
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 instance. For more information, see Defining issue type field values . You can also create new 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 instance. For more information, see Defining priority field values . You can also create new values on-the-fly during the import process.
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	<p>You can choose to have the importer automatically create JIRA users for any values of the Assignee or Reporter field.</p> <ul style="list-style-type: none"> Users will be created as active accounts in JIRA. Users will need to get their passwords emailed to them the first time they log into JIRA. Users with no real name will get the portion of their email address (login name) before the "@" character as their Full Name in JIRA. If you are using External User Management, the import process will not be able to create 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 exist yet 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.

Editing and collaborating on issues

Work on your issues more efficiently with these tips and tricks for editing and collaborating on issues.

In addition to learning about the basics of editing and commenting on an issue, you can refer to this page for help with:

- Using the wiki toolbar to make your comments and descriptions pop
- Sharing issues with your team and mentioning other people
- Keeping track of issues with labels and issue watchers

On this page:

- Attaching files and screenshots
- Collaborating on issues
- Editing issue details
- Commenting on issues
- Formatting text with wiki markdown
- Tracking issues with labels
- Watching and voting for issues

Attaching files and screenshots

If your administrator has enabled file attachments, you and your customers can attach files and screenshots to issues you're working on. See [Attaching files and screenshots to issues](#) for more information.

Collaborating on issues

You can easily keep your team informed by using the



button to share an issue with other JIRA users. If your administrator has enabled anonymous access, you can also share issues by entering the email address of a non-JIRA user.

If you want to invite members of your team to help you work on an issue, you can mention them by typing @ and their username in the issue description or comment. People already involved in the issue, like the reporter or a commenter, will be listed first in the user list so you can select them faster. Note that the users you mention will be notified once you save the issue description or comment.

Editing issue details

What permissions do you need?

To edit an issue, you need the **Edit Issue** project permission for the issue's relevant project. If you do not have this permission, please contact your administrator.

To edit an existing issue, select **Edit** to open the Edit Issue dialog box and modify the issue details. If you want to change the fields you need to edit, select **Configure Fields > Custom** and choose the fields you want to show or hide. Select **Update** to save your changes.

Commenting on issues

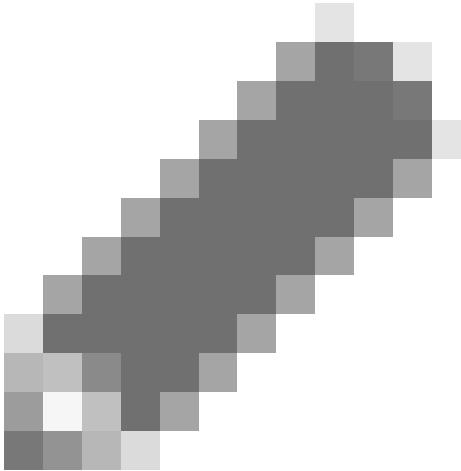
What permissions do you need?

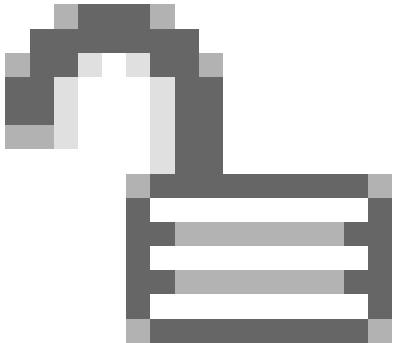
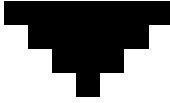
To add comments to an issue, i.e. to see the **Comment** button, you must have both of the following

project permissions for the issue's relevant project:

- **Browse Project** permission to view the issue to be commented on
- **Add Comments** permission to add a comment to the issue.

Note that you automatically become a watcher of the issues that you comment on. You can disable this via the **Preferences > Autowatch** option in your profile.

What	How
Add a comment	Simply click Comment and type your comment for the issue.
Delete a comment	On the comment you wish to delete, select the trashcan icon located on the comment. Confirm that you want to remove this comment from the issue by selecting Delete when prompted.
Edit a comment	<p>Select</p>  <p>located on the comment, and edit the text or restrictions (Viewable by...) as needed. When you save your revised comment, you'll see 'edited' displayed to indicate that the comment has been edited:</p> <div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <p> Susan Griffin added a comment - 15/Mar/13 2:36 PM - edited   </p> <p>I think this nerd is looking better. Thanks for fixing it up!</p> </div> <p>You can hover over 'edited' to see who edited the comment and when.</p>
Link to a comment	<p>Right-click on the Permalink icon on the comment, then copy the permanent link to the comment. Paste the copied permanent link into your email or chat message.</p> <p>Clicking the permanent link takes you to that particular comment in the JIRA issue. If your JIRA issue contains an extensive list of comments, the issue page will automatically be scrolled down so that the linked comment is visible.</p>

Restrict a comment	<p>Apply viewing restrictions to a comment by selecting the open padlock icon</p>   <p>(or  Restricted to Users if restrictions already apply).</p>
--------------------	--

Formatting text with wiki markdown

JIRA application [Text Formatting Notation](#) allows you to use rich-text features, such as:

- Italic, bold, underlined text
- Multiple levels of headings
- Bullets, numbered lists, tables, and quotations
- Images
- Macros

When you edit an issue description, comment, or any rich-text field, you can expand the simple wiki editor toolbar to format your text and select **preview** to see how your formatted text will appear. Note that your JIRA administrator can enable, disable and configure the which allows you to use wiki markdown, so your options may vary slightly. Note that if you're administrator has enabled the rich text editor, you'll still be able to format your content using wiki markdown, but if you select the [visual editor](#), you'll see the markdown applied directly.

Tracking issues with labels

Labeling helps you categorize and search for an issue. When viewing an issue, select **More > Labels** to add or remove labels, which will appear in the Details section:

Details			
Type:	 Documentation SubTask	Status:	 Open (View Workflow)
Priority:	 Minor	Resolution:	Unresolved
Affects Version/s:	6.0	Fix Version/s:	6.0-OD10
Component/s:	None		
Labels:	doc		

You can click a label (e.g. **doc** in the above screenshot) to jump to the Issue Navigator and see a list of all issues that have this label. You can also add the [Labels Gadget](#) to your dashboard to quickly find issues with labels relevant to you and your team.

Watching and voting for issues

What permissions do you need?

To view other users watching or voting for an issue, you need the **View Voters and Watchers** and **Manage Watcher List** project permissions.

If your administrator has set up the needed notification scheme, you can select **Start watching this issue** to be automatically notified of issue updates. You can also click the number of watchers on the issue to add other JIRA users as watchers.

If your administrator has enabled the voting on issues, you can select **Vote for this issue** to encourage the

responsible team to resolve or complete the issue.

Linking issues

Issue linking allows you to create an association between two existing issues on either the same or different JIRA servers. For example:

- An issue may *relate* to another.
 - An issue may *duplicate* another.
 - An issue may *block* another.

Issue linking also allows you to:

- Create a new linked issue from an existing issue in a service desk or business project.
 - Create an association between an issue and a Confluence page.
 - Link an issue to any other web page.

Your JIRA administrator can customize the types of links that you can create.

On this page:

- Creating a link to another issue on the same JIRA site
 - Creating a link to an issue on another JIRA site
 - Create a new linked issue from an existing issue in a service desk or business project
 - Creating a link to a Confluence page
 - Creating a link to any web page URL
 - Deleting a link
 - Searching for linked issues

Issue links within an issue look like this:

Screenshot: the 'Issue Links' section within an issue

Issue Links	
belongs to Epic	JRADEV-16605 [1] May 2013 - Master documentation issue for JIRA 6.0.0
clones	JRADEV-16620 3 Dec - JIRA 6.0-OD1 documentation issue
is cloned by	JRADEV-17453 14 Jan - JIRA 6.0-OD4 documentation issue
relates to	JRADEV-15643 Newly onboarded plugin developers of development plans JRADEV-16065 Update final 5.2 docs to include webhooks "Exclude issue details" flag JRADEV-16150 Webhooks, note that JIRA supports a "responsive" and not "initiative"

Note: Resolved issues (i.e. issues with a Resolution set) are displayed in strike-through font, e.g. DEMO-1.

To create links on issues, you need to have the Link Issues permission in the project(s) to which the issues belong.

Creating a link to another issue on the same JIRA site

1. Open the issue you wish to link to another issue in the same JIRA site.
 2. Select **More > Link** to display the **Link** dialog box.

Select a JIRA issue to link this issue to

Server

This issue

Issue

or search for an issue
Begin typing to find recently viewed issues

Create reciprocal link
JIRA will create a link from the issue on the remote JIRA instance back to this issue

Comment

3. Ensure that the **JIRA Issue** item is selected at the left of the dialog box and then choose the type of link to be created from the **This issue** drop-down list.

i If your JIRA system administrator has configured *fully reciprocal application links* between your JIRA site and another one, a **Server** drop-down list may appear above the **This issue** list. If this is the case, ensure your JIRA site appears or has been selected from the **Server** list.
4. In the **Issues** field, specify the issue(s) to be linked to your currently viewed/selected issue. There are two ways to do this:
 - Type the full issue key (e.g. **ABC-123**) — or to link to multiple issues, press the 'Enter' key between each typed issue key.

i If you have previously browsed an issue, you can quickly find the issue by typing the first few letters of the issue key (or part of the Summary), which will appear in an 'autocomplete' drop-down list for selection:

OR:

 - Click the **search for an issue** link to use the **Find JIRA issues** popup, which allows you to perform either a simple **text search** or an **advanced search** for issues.
5. Optional: Add a **Comment** to describe why you are linking these issues.
6. Click the **Link** button at the bottom of the dialog.

Creating a link to an issue on another JIRA site

! To create this type of link, your JIRA system administrator should have configured *fully reciprocal application links* between your JIRA site and the other JIRA site containing the issue(s) you want to link to.

1. Open the issue you wish to link to another issue.
2. Select **More > Link** to display the **Link** dialog box.
3. Ensure that the **JIRA Issue** item is selected at the left of the dialog box.

i Note:

- This option will not be available if your JIRA system administrator has not configured an application link between your JIRA site and the remote JIRA site.
- If, after selecting this option, you are prompted for authorization, you may be required to log in to the remote JIRA site, which will allow your JIRA site to access the remote JIRA site *on behalf of your account on the remote JIRA site*.
- i** This behavior means the application links configured between your JIRA site and the remote JIRA site use OAuth authentication.

4. If your JIRA site is connected to multiple remote JIRA sites, choose the relevant JIRA site from the **Server** drop-down list.
5. Choose the type of link to be created from the **This issue** drop-down list.
6. Type the **Issue** key of the issue on the remote JIRA site that you want to link to. Alternatively, you can search for issues on the remote JIRA site by clicking the **search for an issue** link, which opens the **Find JIRA issues** popup.
 You can link to any issue on the remote JIRA site to which you have access on that site.
7. Select the **Create reciprocal link** checkbox to create the complementary link on the remote issue you are linking to, back to your issue. For example, if you create a **blocks** link type to a remote issue, the reciprocal link generated on the remote issue will be a **is blocked by** link type back to your local issue.
8. Optional: Add a **Comment** to describe why you are linking these issues.
9. Click the **Link** button at the bottom of the dialog.

Troubleshooting

 **Problem:** If you selected the **Create reciprocal link** checkbox, but after clicking the **Link** button, you discover that a reciprocal link from the remote issue back to your issue has not been created, then your JIRA system administrator has most likely created only a one-way link from your JIRA site to the remote JIRA site.

 **Solution:** Ask your JIRA system administrator to configure *fully reciprocal application links* between your JIRA site and the remote JIRA site.

 **Problem:** If you attempted to create a reciprocal link but received the following message:

'A reciprocal link from issue 'XYZ-123' back to this issue was not created as the remote JIRA server returned the following error: No Link Issue Permission for issue 'XYZ-123'.' (where 'XYZ-123' is the issue key on the remote JIRA site),

then a reciprocal link on the remote JIRA site will not have been created, because the user account through which you authenticated on the remote JIRA site (at step 3 above) does not have the Link Issues project permission.

 **Solution:**

- Ask the JIRA project administrator(s) on the remote JIRA site to grant your user account the Link Issues project permission for the relevant project(s) to which you need to create issue links.
- Alternatively, if the application link between your JIRA site and the remote JIRA site use OAuth authentication and you suspect you may have authenticated on the remote site with another user account that does not have the Link Issues project permission, repeat the procedure above but during the authorization step (at step 3), authenticate on the remote site with a user account which has this permission.
 If you are not prompted for authentication during authorization, try clearing your browser's cookies first and repeat the procedure again.

Create a new linked issue from an existing issue in a service desk or business project

To create a linked issue, you need to have Create issue and Linked Issues permissions in the destination project(s).

To create a linked issue:

1. Open the issue from which you wish to create the linked JIRA issue.
2. In the Issue screen, select **More > Create linked issue** to display the **Create Linked Issue** dialog box.
3.  **Keyboard Shortcut:** '!' + start typing **Create linked issue**. The newly created linked issue contains the same Project, Issue Type, and Summary information stored in the original issue. It is also linked to the service desk issue, in this case CCF-3.

Create linked issue

Project

Issue Type Problem

Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

Created issue

Linked issues

Search for issues to link to from the one you're creating.

Summary

Description

Copy attachments
 Copy links

4. Select the destination **Project** in which the new linked issue is to be created.
5. Select the correct Issue Type for the new linked issue.
6. In the **Linked issues** field, specify issue(s) to be linked to your new linked issue.
7. Edit the linked issue **Summary**.
8. Edit the **Description** and describe why you are linking these issues.
9. Select the **Copy attachments** checkbox to include any attachments from the original issue.
10. Select the **Copy links** checkbox to include any URLs from the original issue.
11. Click the **Create** button at the bottom of the dialog.

Your linked issue has now been created.

Creating a link to a Confluence page

This feature is only supported in Confluence versions 4.0 or later.

! To create this type of link, your JIRA system administrator needs to have configured an *application link* between your JIRA site and the Confluence site containing the pages you want to link to.

1. Open the issue you wish to link to another issue.
2. Select **More > Link** to display the **Link** dialog box.
3. Click the **Confluence Page** option at the left of the dialog box.

i This option is not available if your JIRA system administrator has not configured an application link between your JIRA site and Confluence site.
4. If more than one application link has been configured between your JIRA site and other Confluence sites, then choose the appropriate Confluence site from the **Server** drop-down list.
5. Specify the Confluence page to be linked to your currently viewed issue. There are two ways to do this:
 - In the **Page URL** field, enter the URL of a page on the Confluence site you want to link to. For example:

http://<confluence-server>/display/ds/Welcome+to+the+Confluence+Demonstration+Space

- Click the **search for a page** link. The **Link** dialog box is replaced by the **Find a Confluence page** dialog box.
i If you are prompted for authorization, you may be required to log in to the Confluence site, which will allow your JIRA site to access the Confluence site *on behalf of your account on the Confluence site*.
This behavior means the application links configured between your JIRA site and the remote Confluence site use OAuth authentication.
 - a. In the first **Search** field, specify one or more search terms that appear in the page you want to link to. This field is mandatory.
 - b. Optional: In the second **Search** field, select the Confluence space to further narrow down the search.
 - c. Click the **Search** button and then the title of the page you want to link to.
- 6. Optional: Add a **Comment** to describe why you are linking these issues.
- 7. Click the **Link** button at the bottom of the dialog.

Troubleshooting

x Problem: If Confluence page links you create show **Failed to load** on the issue or if you attempted to search for a Confluence page but received the following message:

'Content on the Confluence site could not be accessed because the Confluence server's 'Remote API' feature is disabled. The Confluence system administrator must enable this 'Remote API' feature for JIRA to successfully access this content.'

then JIRA was unable to communicate with the Confluence server to either:

- retrieve information about the link or
- conduct a Confluence page search in the **Find a Confluence page** dialog box.

✓ Solution:

Ask the Confluence system administrator to enable 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.

Creating a link to any web page URL

1. Open the issue you wish to link to another issue.
2. Select **More > Link** to display the **Link** dialog box.
3. Click the **Web Link** option at the left of the dialog box.
4. Specify the **URL** of the web page you want to link to.
5. Specify the **Link Text** that will appear in the **Issue Links** section of the 'view issue' page and will be hyperlinked to your URL.
6. Optional: Add a **Comment** to describe why you are linking these issues.
7. Click the **Link** button at the bottom of the dialog.

Deleting a link

1. Go to an issue that contains links, and locate the **Issue Links** section (see screenshot above).
2. Hover your mouse over the link you wish to delete, and click the **Delete** (trashcan) icon that appears.

Searching for linked issues

You can search for issues that are linked to a particular issue. See [Advanced searching](#) for more information.

i Be aware that this functionality does not extend to issues on a remote JIRA server.

Editing multiple issues at the same time

At some point, you may need to change multiple issues at the same time.
You can do this by performing a bulk operation.

There are restrictions placed on some of the bulk operations. For example, if

you select multiple issues with different workflows, you can only transition them in groups with the same workflow, and one group at a time. The restrictions are explained further in the relevant sections.

On this page:

- Before you begin
- Transition multiple issues
- Delete multiple issues
- Move multiple issues
- Edit multiple issues
- Watch / stop watching multiple issues

Before you begin

Required permissions - To perform a bulk operation, you'll need the appropriate project-specific permission and the global Bulk Change permission. For example, you would need to have both the **Move Issue** and **Bulk Change** permissions to perform the **Bulk Move** operation.

Disabling Mail Notification for Bulk Operations - You can disable mail notifications for a particular bulk operation by deselecting the **Send Notification** checkbox in the bulk operation wizard. For this option to be available, you must be an administrator or project administrator of all the projects associated with your selected issues.

Using the bulk change wizard - The bulk change wizard will progress you through your bulk change. To step back at any step of the operation, select the relevant step in the menu on the left-hand side. Selecting **Cancel** will cancel the entire process.

Transition multiple issues

This bulk operation allows you to transition multiple issues through a workflow at the same time. You can only perform one transition bulk operation at a time. You will also need to provide any values required to complete the transition. For example, to close multiple issues, you will need to provide a value for the Resolution field, such as Done, Fixed, or Won't Fix.

▼ How to transition multiple issues

1. Perform a search with the required filters to produce a list of issues.
2. Select **Tools > Bulk Change**.
3. Select the issues you'd like to perform the bulk operation on, and select **Next**.
4. Select **Transition Issues**, and select **Next**.
5. Select the available workflow action. The actions available are dependent on the issues (and their associated workflows) that you have selected. Select **Next**.
6. Select a value for any required fields for this transition, and if available, decide whether you'd like to send email notifications. Select **Next**.
7. Review your bulk operation, and select **Confirm** when you are happy with the operation.

Delete multiple issues

This bulk operation allows you to delete multiple issues at the same time.

▼ How to delete multiple issues

1. Perform a search with the required filters to produce a list of issues.
2. Select **Tools > Bulk Change**.

3. Select the issues you'd like to perform the bulk operation on, and select **Next**.
4. Select **Delete Issues**, and select **Next**.
5. If available, decide whether you'd like to send email notifications. Select **Next**.
6. Review your bulk operation, and select **Confirm** when you are happy with the operation.

Move multiple issues

This bulk operation allows you to move multiple issues at the same time. The issues you're moving need to be mapped to both a project and an issue type, and in doing this, you may need to also map the status and fields of the issues. Subtasks need to be mapped, too.

▼ How to move multiple issues

1. Perform a search with the required filters to produce a list of issues.
2. Select **Tools > Bulk Change**.
3. Select the issues you'd like to perform the bulk operation on, and select **Next**.

▼ More information...

The bulk move operation can be performed on both standard issues and sub-task issues.

Standard issues can be moved to another project and issue type, whereas a sub-task can only have its issue type changed. (Note that it is possible to convert a sub-task to an issue, and vice versa.)

It is **not** possible to select *both* a sub-task and its parent to bulk move. This is so as to adhere to the parent/sub-task relationship (i.e. the sub-task is always located in the same project as the parent issue). Any sub-tasks of selected parent issues that were also selected will be automatically discarded from the move.

For example, you have issue B being a sub-task of issue A and you try to bulk move both A and B simultaneously. You will see a warning message (see below) and will be prompted to select a target project and issue type for issue A. If you select a new project for A, you will be prompted to move the sub-task to a new issue type based on issue A's new project. If you *don't* change the project for issue A, the sub-task will not be required to be moved.

4. Select **Move Issues**, and select **Next**.

The bulk move operation may require additional information dependent on which issues you have selected to move. This information is requested as follows:

- a. Select Projects and/or Issue Types

▼ More information...

The first step of the Bulk Move wizard is to choose which projects and issue types you will move your issues to. The target project and issue type will determine whether extra steps will be required to migrate statuses and fields.

Selected issues are grouped by their current project and issue type. You can either select a new project and issue type for each one or choose to move all standard issues to a single project and issue type.

i Note: This *does not apply to sub-tasks* since they cannot be moved to a standard issue type.

- b. Select Projects and/or Issue Types for Sub-Tasks

▼ More information...

If you are moving issues with sub-tasks to another project, you will also need to move the sub-tasks to the new project. You can also elect to change the issue types of the sub-tasks being moved if you need to.

- c. Select status migration mappings for invalid statuses

▼ More information...

As multiple workflows can be active simultaneously, some statuses associated with the collection of selected issues may not be valid in the target workflow. In this case, you should map invalid statuses to valid statuses in your new workflow.

- d. Select values for required fields and fields with invalid values

▼ More information...

In order to adhere to the field configuration scheme associated with the target project

and issue type, it may be necessary to update/populate required fields (e.g. fields that are required in the target project, but may not have been in the original project).

For each field that needs to be populated, you will be prompted to supply a value. This value will be applied to all issues that are being *Bulk Moved* together.

For the following fields, you can select from a list of possible values provided for you:

- Component
- Affects Version
- Fix Version
- Custom fields of type 'Version-Picker'

Note that versions which have been archived in the target project cannot be selected as the target when performing a bulk move. If you need to move issues into an archived version, you will need to first unarchive the version in the target project.

It is possible to retain original field values that are valid in the target destination by checking the **Retain** checkbox associated with the field. For example, some issues may already include a valid custom field value — these values can be retained, while issues that require an update will adopt the value specified on the **Field Update** screen.

- **Checked:** the original value is retained where possible¹. The field will not be updated with the specified new value.
- **Unchecked:** all fields will be updated with the specified new value.

Note that the 'Retain' checkbox is not available for the following fields, since an explicit mapping is required:

- Component
- Affects Version
- Fix Version
- Custom fields of type 'Version-Picker'

2. Confirm changes to be made and complete the operation

More information...

When all move parameters — e.g. target project, status mappings and field updates — have been specified for all issues, you will be presented with a confirmation screen displaying all changes that will be made to the issues being moved. The following details are displayed as applicable:

- **Issue Targets:** the target project and issue type
- **Workflow:** the target workflow and invalid status mappings
- **Updated Fields:** new values for fields that require updating
- **Removed Fields:** values to be removed in fields that are not valid in the target

The issues will only be moved once the **Confirm** button is clicked from the confirmation page. If the operation is exited anytime before this step, no changes will be made to the issues.

Note that steps C and D above will occur once for each different target project and issue type combination.

Edit multiple issues

This bulk operation allows you to edit multiple issues at the same time. The bulk edit operations available depend on the issues selected and the nature of the field/s you want to change.

How to edit multiple issues

1. Perform a search with the required filters to produce a list of issues.
2. Select **Tools > Bulk Change**.
3. Select the issues you'd like to perform the bulk operation on, and select **Next**.
4. Select **Edit Issues**, and select **Next**.
5. Select the bulk edit operation from the list of available operations (expand more information for a full list of available and unavailable operations, and their conditions).

More information...

Available Operations	Conditions
Change Affects Version/s	<ul style="list-style-type: none"> Selected issues belong to one project, and that project has version/s This field is not hidden in any field configurations the selected issues belong to Current user has 'edit issue' permission for all the selected issues
Change Assign To	<ul style="list-style-type: none"> This field is not hidden in any field configurations the selected issues belong to Current user has 'assign issue' permission for all the selected issues
Change Comment	<ul style="list-style-type: none"> This field is not hidden in any field configurations the selected issues belong to Current user has 'comment issue' permission for all the selected issues
Change Component/s	<ul style="list-style-type: none"> Selected issues belong to one project, and that project has component/s This field is not hidden in any field configurations the selected issues belong to Current user has 'edit issue' permission for all the selected issues
Change Due Date	<ul style="list-style-type: none"> This field is not hidden in any field configurations the selected issues belong to Current user has 'edit issue' permission for all the selected issues Current user has 'schedule issue' permission for all the selected issues
Change Fix For Version/s	<ul style="list-style-type: none"> Selected issues belong to one project, and that project has version/s This field is not hidden in any field configurations the selected issues belong to Current user has 'edit issue' permission for all the selected issues
Change Issue Type	<ul style="list-style-type: none"> Current user has 'edit issue' permission for all the selected issues
Change Priority	<ul style="list-style-type: none"> This field is not hidden in any field configurations the selected issues belong to Current user has 'edit issue' permission for all the selected issues
Change Reporter	<ul style="list-style-type: none"> This field is not hidden in any field configurations the selected issues belong to Current user has 'edit issue' permission for all the selected issues Current user has 'modify reporter' permission for all the selected issues
Change Security Level	<ul style="list-style-type: none"> This field is not hidden in any field configurations the selected issues belong to All the selected projects are assigned the same issue level security scheme Current user has 'edit issue' permission for all the selected issues Current user has 'set issue security' permission for all the selected issues
Change Custom Fields	<p>The 'Change Custom Fields' operation is available only if:</p> <ul style="list-style-type: none"> a global custom field exists OR an issue type custom field exists and the issues are all of this specific issue type OR a project custom field exists and the issues are all of the same project
Edit a Closed Issue	<ul style="list-style-type: none"> Your workflow must allow editing of closed issues

Change Sprint	<p>You need to specify the sprint ID.</p> <ul style="list-style-type: none"> • This operation only affects active and future sprints, i.e. closed/completed sprints are not included when bulk editing the Sprint field.
---------------	---

Unavailable Operations

The fields listed in this section have no operations for bulk editing. This is because there is an alternative method or it is not logical to perform bulk edit on them.

The following system fields are unavailable for bulk editing:

- Attachments
- Summary
- Description
- Environment
- Project — Please use 'Bulk Move' to move issues between projects
- Resolution — Please use 'Bulk Workflow Transitions' to modify the resolution of issues
- Time Tracking fields — Original Estimate, Remaining Estimate, Time Spent

The following custom field types are unavailable for bulk editing:

- Import Id
- Read Only Text

6. Select a value for any required fields for this operation, and if available, decide whether you'd like to send email notifications. Select **Next**.
7. Review your bulk operation, and select **Confirm** when you are happy with the operation.

Watch / stop watching multiple issues

These bulk operations allows you to start watching or stop watching multiple issues at the same time.

How to watch multiple issues

1. Perform a search with the required filters to produce a list of issues.
2. Select **Tools > Bulk Change**.
3. Select the issues you'd like to perform the bulk operation on, and select **Next**.
4. Select **Watch Issues**, and select **Next**.
5. Review your bulk operation, and select **Confirm** when you are happy with the operation.

How to stop watching multiple issues

1. Perform a search with the required filters to produce a list of issues.
2. Select **Tools > Bulk Change**.
3. Select the issues you'd like to perform the bulk operation on, and select **Next**.
4. Select **Stop Watching Issues**, and select **Next**.
5. Review your bulk operation, and select **Confirm** when you are happy with the operation.

Scheduling an issue

You can schedule issue due dates in JIRA Software to track and review, and inform teams about issue dates. The powerful scheduling feature allows you to perform fixed and relative date searches based on specific due dates, as well as arbitrary search periods. You can also perform advanced searches using JIRA Query Language.

Scheduling an issue

To schedule an issue, populate its **Due** date field. This can be done either when creating an issue, or at a later stage by editing the issue.

To enable Issue Scheduling, at least one group or project role must be given the Schedule Issues permission by your JIRA administrator. Only users with the Schedule Issues permission can populate the **Due** date field.

Searching by due date

You can use either [basic search](#) or [advanced search](#) to search for issues by their Due Date.

Using simple search

You can search for issues using the search form in Issue Navigator (see [Searching for issues](#)). There are two ways to search for issues based on the **Due** date field. The first way is using fixed date values, the second is using periods that are relative to the current date.

Fixed date searches

There are two text fields in the search form that allow searching based on the **Due** date field.

- To search for all issues that are due after a certain date, enter the date in the Due After text field. For example, to find all issues that are due after 1st June 2010, enter 1-6-2010 in the Due After field. You can also use the Calendar popup to select a date by clicking the calendar icon to the right of the field.
- To search for issues that are due before a certain date, enter the date in the Due Before text field. For example, to find all issues that are due before 1st July 2010, enter 1-7-2010 in the Due Before field.

To search for issues that are due between two dates, populate both the Due After and the Due Before fields.

Relative period search

It is possible to perform a search that is relative to the time when it is run. For example, it is possible to do a search for issues that are due seven days from now. To do this, enter 7d in the Due Date To text field of the Issue Navigator. If the search is saved and run the next day, the issues that are due in seven days from the time that the search is run will be retrieved. Thus, this search will find all issues that are due within a week every time it is run.

The values that are entered in the Due Date From and Due Date To fields have to conform to a special syntax (described below). However, it is also possible to use the Due Date popup by clicking the icon to the right of the Due Date To text field to specify the search period.

Due Date Popup

Use the Due Date popup to do the following:

- To search for issues that are overdue at the time of the search, select the first radio button, and click **OK**.
- To search for issues that are overdue by more than a certain number of days, populate the text field in the second row, and click **OK**.
- To search for issues that are due in the next certain amount of days, and are not overdue at the time of the search, populate the text field in the third row with the number of days, and choose **and not** from the select box in the third row. Select the third radio button, and click **OK**.
- To search for issues that are due in the next certain amount of days, and are overdue at the time of the search, populate the text field in the third row with the number of days, and choose **and** from the select box in the third row. Select the third radio button, and click **OK**.
- The fourth row of the popup is used for arbitrary period searches. Use the **to** text field to specify the upper bound of the search, and the **from** text field to specify the lower bound of the search. A blank text field means no bound. Populating the text fields in the fourth row actually has the same effect as populating the Due Date From and Due Date To text boxes. The syntax is described below.

Relative Period Search Syntax

The Due Date From and Due Date To fields use a special syntax to denote time period bounds. The syntax uses numbers and abbreviations that follow the numbers to represent what the numbers actually mean. The abbreviations are "w" for weeks, "d" for days, "h" for hours, and "m" for minutes. For example, to specify 10 days in the future, use "10d" or "1w and 3d". To specify a period bound in the past, prefix the value with the "-" sign. For example, to specify 2 days, 4 hours, and 3 minutes ago, use "-2d 4h 3m".

Using advanced search

You can also use JIRA Query Language (JQL) to search for issues by due date — see [Advanced searching](#), and

particularly the documentation on the Due field.

Moving an issue

Sometimes, an issue may belong to a different project, and you may want to move this issue to another project. You can easily do this by using the **Move Issue** wizard.

Before you begin:

- You must have the Move Issues permission for the project that has the issue that you want to move.
- You must have the Create Issues permission for the project that you wish to move your issue to.

If you do not have either of this permissions, please contact your JIRA administrator to have these added to your user profile.

If you wish to move multiple issues between projects at the same time, please refer to the documentation on [bulk moving issues](#).

Moving an issue

The **Move Issue** wizard allows you to specify another project in your JIRA instance to move your selected issue to. As there may be significant differences in the configuration of your original project and target project, the **Move Issue** wizard allows you to change certain attributes of the issue. These include:

- **Issue Type** — If your issue is a custom issue type that does not exist in your target project, you must select a new issue type. You can also choose to arbitrarily change the issue type.
- **Issue Status** — You may have set up custom issue statuses as part of a workflow. If you have assigned a custom status to your issue, and it does not exist in your target project, you must select a new issue status for your issue. You cannot arbitrarily change the issue status, i.e. the option to change the issue status will only appear if you are required to change it.
- **Custom Fields** — If you have defined **required** custom fields for your issue that do not exist in your target project, you must set values for them. You will only be prompted to enter the values for **required custom fields** in the target project that are missing values. If the custom fields of your original project also exist in your target project, and these custom fields are not required in the target project, you may need to set values for them, to move the issue successfully. If you wish to change the existing values for other fields on your issue, you can do this after the move is complete.

To move an issue:

1. View the issue that you wish to move.
2. Select **More > Move**.
3. The first page of the **Move Issue** wizard is displayed. Complete the steps required.
4. The confirmation page will display with all of your changes. If you wish to revise any of your changes, you can click the appropriate step in the left-hand menu to return to that page of the wizard. Once you are happy with your changes, click **Move** to move the issue to the target project.
5. Your issue will be moved to the target project and displayed on screen. You can now edit the issue to make further changes, if you wish.

Moving related issues

- If your issue has sub-tasks, the 'Move Issue' wizard will also move the sub-tasks to the target project.
- If you are moving an epic, the 'Move Issue' wizard will not move the issues in the epic. The epic and the issues in the epic will still be linked to each other, but the issues in the epic will remain in the original project. You will need to move them separately.

Troubleshooting

- Restricted comments appear to be removed after moving the issue. See this article: [Restricted comments disappear after moving an issue to a new project](#).

Approving a service desk request

JIRA Service Desk projects have an option to include an approval step, and assign approvers to their service desk issues. You may be asked to approve a service desk request if you've been assigned as an approver.

You'll receive an email to notify you that your approval is required, and a link to the service desk customer portal where you'll be able to view the request. When in the customer portal, you can also view any outstanding approvals or requests you may have.

The screenshot shows a JIRA Service Desk approval request titled "Please upgrade db.test.stg to Postgres 9.4". The status is "AWAITING APPROVAL". The "Your approval" section has "Approve" and "Decline" buttons. A comment input field says "Comment on this request...". The "Activity" section shows a message from "IT-4" at 10:34 AM: "Request requires approval. 1 approval needed. Today 10:34 AM LATEST". The "Details" section includes a description: "We need to upgrade our test staging DB to complete testing for our new environments." It also asks "Who is your manager?" with "Frank Smith" listed. On the right, under "You can", are "Approve", "Decline", "Add a comment", and "Add attachment". Under "People involved" is "Elaine Gould Creator". Under "Awaiting approval" is "Pending" and "Frank Smith".

Approvers view of a request in the customer portal requiring their approval

Approving and declining requests

1. Navigate to the service desk customer portal by either selecting the link in your email, or entering the URL.
2. View the approval request and review the supporting information.
3. Select **Approve** or **Decline**, and add an optional comment if you want to (you don't need to add a comment, but if you're declining a request it's helpful to let the person who submitted the request know why you declined it). The customer won't receive a response when you approve or decline a request, but they will if you add a comment.

If you're the only approver required on the request, and you approve it, the request will be moved to the status defined in the workflow for the approve transition. If there are more than one approval required, the status will remain the same until all approvers have responded, and your approval will be noted on the request.

If you decline a request (or any of the approvers decline it), it's automatically moved to the status as defined in the workflow for the decline transition, and your response is noted on the request.

Visual editing

Visual editing is part of a Labs feature in JIRA platform 7.2. Labs features can be turned off by your administrator, so if you don't have access to visual editing, that's probably why.

Formatting content in Visual mode gives you a What You See Is What You Get (WYSIWYG) experience. Formatting appears as you apply it, and you no longer have to flip to a Preview to see what your content will look like when saved. You still have the option to view the wiki markdown by selecting the Text tab. You'll know you have access to the visual editor because you'll see the Visual and Text tabs.

Visual	<p>The new visual editor is AWESOME!!</p> <p>I can apply formatting by using the toolbar or typing using wiki markdown, and the formatting appears as I apply it! Things like:</p> <ul style="list-style-type: none"> • colour • underline • emoticons 😊 <p>and I even get</p> <p style="border: 1px solid #ccc; padding: 2px;">Tables</p> <p>that look like tables!</p> <p style="text-align: right;">Text Visual Viewable by All Users Add Cancel</p>
Text	<p>The new visual editor is _AWESOME_!!</p> <p>I can apply formatting using the toolbar or typing wiki markdown, and it appears as I apply it! Things like:</p> <pre>* {color:#d04437}colour{color} * +underline+ * emoticons :)</pre> <p>and I even get</p> <p> Tables </p> <p>that look like tables!</p> <p style="text-align: right;">Text Visual Viewable by All Users Add Cancel</p>

In Visual mode, you can still enter wiki markdown syntax as you add your content, and it'll be rendered exactly as it'll display when you save. You can even flip between modes to view the formatted content, and the wiki markdown syntax. You can also use the toolbar to format and style your content.

As Visual editing is really a preview of what we're working on, there's a few things that may not work quite as you'd expect them:

- Formatting content in a complex way can affect its ability to be rendered, things like tables in the cells of other tables, and adding images to table cells won't work.
- Pasting content may not work as expected, as the source content may really be formatted using a method we don't support. So pasting tables may work, and it may not, depending on the source. Pasting plain text is absolutely fine.

Attaching files and screenshots to issues

To share information with your team, you can attach documents, images, and screenshots to your JIRA application issues.

Multiple files can be attached to an issue. Click an image thumbnail to open a preview, and if there is more than one image in the gallery, navigate to the next image preview by clicking the right arrow in the preview.

On this page:

- Before you begin
- Adding attachments
- Sorting and managing attachments
- Accessing ZIP file contents
- Capturing and attaching screenshots

Before you begin

A JIRA administrator must enable specific user permissions so that you can add attachments and screenshots into issues. The most common permissions are briefly described below. For more information, your administrator should refer to [Configuring file attachments](#).

▼ JIRA administrator set permissions

- You can attach files and screenshots if your JIRA administrator has file attachments enabled.
- You need the **Create Attachments** permission in the appropriate projects.
- The screenshot feature only works with Windows or Mac client. If you use another operating system, you can attach a screenshot using the file attachment feature. For Linux users, please see [our article](#) for enabling this feature.
- If your JIRA admin has disabled thumbnails in JIRA's attachment settings, the image files will appear as a list.
- If your JIRA admin has disabled ZIP support in JIRA's attachment settings, the attachments feature will not be available. You must download the zip file to your computer before accessing its individual files.
- To remove attachments from an issue, you need one of the following the project permissions in that issue's project:
 - **Delete Own Attachments** — to delete files that you have added to the issue.
 - **Delete All Attachments** — to delete files that anyone has added to the issue.

▼ Browser capabilities

- If you're using Google Chrome, Mozilla Firefox, or Internet Explorer 11, attaching screenshots relies on HTML5 compatibility. Safari is not supported.

Adding attachments

You can add file and image attachments to any issue. To add an attachment when you first create an issue, copy a file from your computer and paste it directly in the **Create Issue** dialog. To add attachments to an existing issue, open the issue and follow these steps:

1. Click **More > Attach files**.
2. Add a file(s).
3. Click **Attach** or **Open**.

You can also drag and drop files onto an issue to attach them.

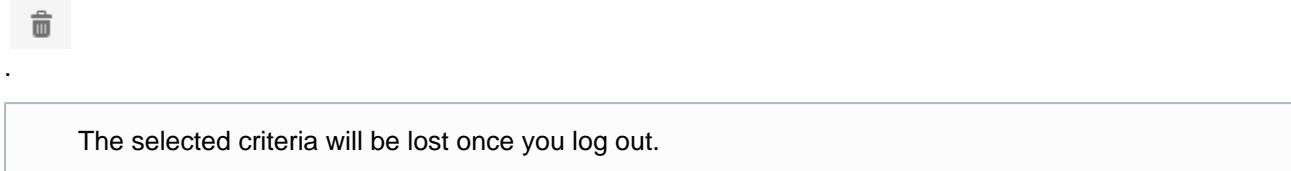
▼ Acceptable file formats, characters, and sizes

- File formats: GIFs, JPGs, PNGs
- A valid file name cannot contain any of these characters: '\', '\"', '%', ':', '\$',

- '?', '*'.
- By default, the maximum size of any one file is 10MB, although this limit can be customized by your JIRA admin.

Sorting and managing attachments

The attachments section of the issue displays a list of options to sort, manage, and download attachments. Select the down-arrow to the right of the attachments section to open the menu. You can reorder the attachments according to a selected criteria. This criteria will be applied to all issues in your project. To remove attachments from the issue, select **Manage Attachments** or hover over the attachment and select



Accessing ZIP file contents

You can view the contents of a zip file (including '.zip' or '.jar' file name extensions) in the attachments section. Click the down-arrow and select **List**. In list view, click the arrow icon in front of the zipped file's name to view and download its individual files. If a file is located within a subdirectory of the zipped file, the path to that file is indicated in the content of the zipped file. To download the entire zip file, click **Download Zip**.

Capturing and attaching screenshots

You can capture a screenshot to the system clipboard and paste it directly into an issue.

- Capture a screenshot using your system keyboard shortcut.
- Paste the image from your clipboard onto the issue using your system keyboard shortcut or right-click menu. The **Attach screenshot** dialog will display.
- Enter a filename.
- Select **Upload**.

Logging work on issues

JIRA Software provides the flexibility to set your estimation and tracking statistics differently, depending on your team's needs. Time Tracking features project schedule planning and time expectations management. Thanks to its reports, you can see the original and current time estimates for

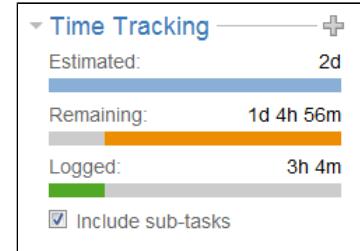
all the issues, and whether they are ahead or behind the original schedule. Product teams often need to be able to estimate how long a product will take to deliver from sprint to sprint; as they work through the stories, the team will develop a cadence of completing <x> units of work they had estimated. This is also called velocity. The team has to track the amount of estimation units they have completed from sprint to sprint to know how much they can fit into each future sprint, and have conviction that all the work will be completed. You will be able to easily monitor the tracking time info by just watching the issue time tracking color bars.

On this page:

- Before you begin
- Setting a time estimate for an issue
- Logging work on an issue
- Editing a work log entry
- Deleting a work log entry
- Customize d JIRA installations

Here's how time tracking appears on an issue:

- The Estimated field displays the amount of time originally anticipated to resolve the issue
- The Remaining field displays the amount of time currently anticipated to resolve the issue
- The Logged field displays the amount of time logged working on the issue so far
- Choosing to include sub-tasks displays the aggregated time of an issue and all its sub-tasks



When you log time for the first time, the time spent is subtracted from the original estimate, and the resulting value is automatically presented in the remaining estimate. When subsequent work is logged, any time spent is subtracted from the remaining estimate.

Before you begin

- Make sure your JIRA administrator has enabled the [Time Tracking](#) feature.
- Make sure you have the Work on Issues, Delete Work Logs, and Edit Work Logs project permissions.

Note that anyone with the Browse Project permission can view time tracking information on an issue.

Setting a time estimate for an issue

Teams can set a time estimate for an issue in order to calculate how long it will take to solve the issue.

1. Open the issue and select **Edit**.
2. Scroll down the Edit issue window to fill in the following time tracking fields:

Field	Description
Original Estimate	Amount of time you believe is required to solve the issue. If you want to change original estimate values once they have logged work time, ask your JIRA administrator to disable legacy mode on time tracking.

Remaining Estimate	Amount of time you believe is required to solve the issue in its current state.
--------------------	---

If the JIRA time tracking feature is in legacy mode, you will only see the original estimate field if work has not been logged. Once work time has been logged, you will only see the remaining estimate field.

Tips:

- You can specify additional time units after a time value 'X', such as Xw,Xd,Xh, or Xm, to represent weeks (w), days (d), hours (h), and minutes (m), respectively. If you type a number without specifying a time unit (e.g. if you type '2' instead of '2h'), the default time unit that your JIRA administrator specified will apply.
- Default conversion rates are 1w = 5d and 1d = 8h.

3. Select **Update**.

When work is first logged against the issue, the **Time Spent** is subtracted from the **Original Estimate**, and the resulting value is automatically presented in the **Remaining Estimate**. When subsequent work is logged, any **Time Spent** is subtracted from the **Remaining Estimate**.

Additionally, once work has been logged on an issue, various reports based on the time tracking information become available.

Logging work on an issue

Once you have started to work on a specific issue, you can log your work by following these steps:

1. Select the issue you want to log time on.
2. Go to **More > Log Work**.
3. Fill in the following **Log Work** fields, and select **Log**:

Log Work field	Description
Time spent	The amount of time spent on the issue. This is the aggregate amount of time that has been logged against this issue.
Date started	Date and time when you started this unit of work.
Remaining estimated	Amount of time anticipated to resolve the issue after completing this unit of work. You can adjust this value using the following options: <ul style="list-style-type: none"> • Adjust Automatically - Adjust the remaining estimate value by subtracting the amount of work logged in the Time Spent field from the remaining estimate current value. • Leave Estimate unset - This option is displayed only if no time estimate has been specified on the issue. You can use this option when you want to keep track of work, but you don't necessarily have a time estimate for an issue. • Use Existing Estimate of - Select this option if you do not want to change the issue remaining estimate value. • Set to - You can adjust the remaining estimate value to the amount of time you specify in this field. • Reduce by - Select this option to manually adjust the remaining estimate value by subtracting the amount of time you specify in this field.

<p>Work description</p> <p>Type a description related to the achieved work.</p> <p>Comments are copied to the Workflow Description by default, but your JIRA administrator can change this option in the 'Copy Comment to Workflow Descriptions' settings. If this setting is disabled:</p> <ul style="list-style-type: none"> • The work log entry may be visible to anyone. If this is a concern, you need to edit this work log entry after creating it to modify its visibility. • You have to manually copy comments to a workflow description once you have logged work.

You can also log work while resolving or closing an issue by closing it and editing the log work fields. Select the padlock icon to set the work logged to be viewable only by members of a particular project role or group.

Editing a work log entry

You can edit your own work log entries if you have been granted the Edit Own Work Logs permission. You can also edit other people's work log entries if you have been granted the Edit All Work Logs permission.

Deleting a work log entry

You can delete your own work log entries if you have been granted the Delete Own Work Logs permission. You can also delete other people's work log entries if you have been granted the Delete All Work Logs permission.

1. Go to the desired issue, and open the **Work Log** tab.
2. Hover over the work log entry to display the actions for the entry on the right side.
3. Select the entry you want to delete, and click the trash can icon. You will be prompted to choose how the Remaining Estimate is affected by deleting the work log:

Option field	Description
Auto adjust	Choose this option to automatically add the time spent value to the current remaining estimate value.
Leave existing estimate	Select this option if you do not want to change the issue remaining estimate value.
Set estimated time remaining	Choose this option to manually set the issue's remaining estimate value to the specified amount.
Increase estimated time remaining	Select this option to increase the estimated remaining.

4. Click **Delete**.

Customized JIRA installations

JIRA applications can be customized by your JIRA administrator by adding the Log Work and Time Tracking fields to the customized screens. This way, you can log work and specify time estimates on the same JIRA screen when performing any JIRA operation, such as editing, creating an issue, or transitioning an issue to another status.

If you want to work *and/or* specify time estimates on the same JIRA screen:

1. Navigate to the issue and view its details.
2. Perform the customized JIRA operation that allows you to log work *and* specify time estimates on the same JIRA screen. For example, assuming that your JIRA administrator has added the **Time Tracking** fields to the **Resolve Issue Screen**, and assuming this screen also retains the default **Log Work** fields, select **Workflow > Resolve Issue** at the top of the issue.

- If your JIRA administrator has configured the Log Work fields as optional, then you can choose whether or not to log work by checking the Log Work checkbox.
- If your JIRA administrator has made logging work mandatory, you will not see the Log Work checkbox, and will instead need to log work when transitioning an issue.

Estimating an issue

Before you begin

Estimating stories in your backlog helps you predict how long it would take you to deliver certain portions of the backlog. Note that this discussion refers to the best practices we've implemented as the main path in JIRA Software — you can choose not to use this approach if you feel it's really not suitable for your team.

 This page only applies to Scrum boards.

On this page:

- Before you begin
- Estimate an issue
- Concepts about estimation

Related pages:

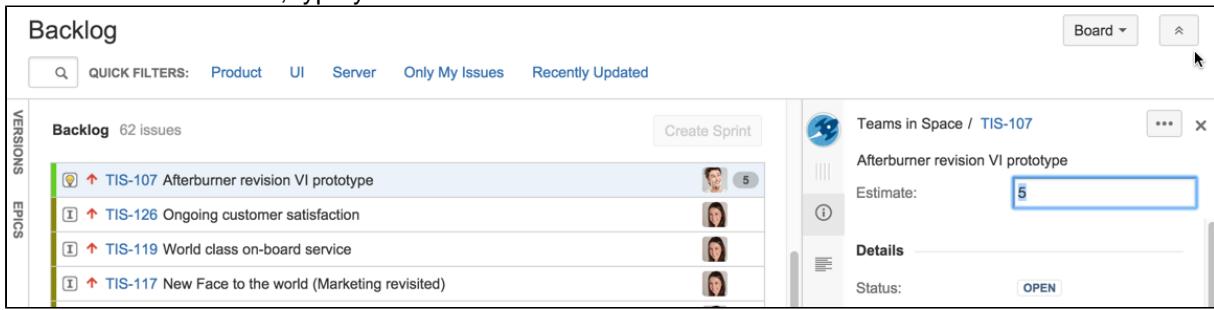
- Configuring estimation and tracking
- Using the backlog
- Using active sprints

Estimate an issue

Before a sprint starts, you need to enter the Original Estimates of your issues. And as you work on the issues during the sprint, you may need to adjust the Remaining Estimates as necessary.

To enter the Original Estimate, do the following for each issue:

1. Navigate to the **Backlog** of your desired board.
2. Click the issue that you want to set the Original Estimate for.
3. In the Issue Detail View, type your estimate in the **Estimate** field.

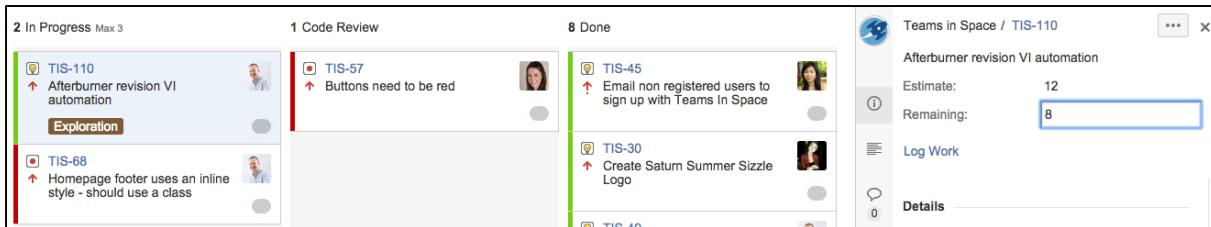


The screenshot shows the JIRA Backlog interface. On the left, there are navigation tabs for 'VERSIONS' and 'EPICS'. The main area displays a list of 62 issues under the heading 'Backlog'. One issue, 'TIS-107 Afterburner revision VI prototype', is selected and shown in the details view on the right. The details view includes fields for 'Estimate' (set to 5), 'Details', and 'Status' (set to OPEN). A sidebar on the right shows 'Teams in Space / TIS-107' and lists other issues: 'Afterburner revision VI prototype', 'Estimate: 5', 'Details', and 'Status: OPEN'.

 The type of units used by the 'Estimate' field (e.g. hours) is affected by your Estimation Statistic — see [Configuring estimation and tracking](#).

To adjust the Remaining Estimate:

1. Navigate to the **Active sprints** of your desired board.
2. Click the issue that you want to adjust the Remaining Estimate for.
3. In the Issue Detail View, type your estimate in the **Remaining** field.



- i** The type of units used by the 'Remaining' field (e.g. hours) is affected by your Tracking Statistic — see [Configuring estimation and tracking](#).

Concepts about estimation

Here are some concepts to consider when estimating issues in JIRA Software.

Estimation is different from tracking

In Scrum, there is a distinction between estimation and tracking. *Estimation* is typically performed against Primary Backlog Items (PBIs, usually stories), and is used to work out how long portions of the backlog might take to be delivered. *Tracking* refers to monitoring the progress of a sprint, to be sure that all stories included in the sprint will be delivered.

Tracking is often performed by breaking down stories into tasks, and applying hour estimates to them during sprint planning, then monitoring the remaining time in a burndown during the sprint.

How is estimation done in traditional development environments?

In traditional development environments, estimation is done this way:

1. A team estimates items in 'man-hours' – and these estimates are assumed to be accurate.
2. The team then calculates the total number of man-hours for the backlog of a project.
3. The team then divides the total number of man-hours by the number of people on the team, and the man-hours in a week. This becomes the forecast date for the project.

These estimates are often inaccurate because they don't consider the following:

- The natural estimation characteristics of the team – meaning, over- and under-estimation are not considered
- Unexpected interruptions during the man-hours allocated to the items
- The performance of the team members themselves over time

When the estimates become inaccurate, the team then exerts time and effort in trying to 'force' the estimates to be accurate. This makes the man-hour approach difficult — if not impossible.

Estimation in Scrum world is all about velocity

In the Scrum world, teams don't try to achieve estimation accuracy. Instead, they aim to achieve 'reliable velocity'.

Velocity is a measure of the number of estimation units that a team tends to complete from sprint to sprint. After their first few sprints, most teams will achieve a reasonably consistent velocity. Armed with velocity and estimates on the PBIs in the backlog, teams can predict more accurately how long portions of the backlog will take to complete.

The key is, the estimation unit doesn't matter – as long as it becomes reasonably predictable from sprint to sprint. For example, teams can use 'ideal hour' estimates, but it's neither necessary or expected that those hours will have any relationship to elapsed time. If a team has a 'man-hour' capacity of 120h in each sprint but a velocity of 60h, that makes no difference because you can still use the 60h velocity to estimate the number of sprints that portions of the backlog will take to complete — and therefore, the elapsed time.

Many people then start wondering where 'the other 60 hours' went, thereby implying that there is something wrong with team productivity. But that's usually got nothing to do with it: a team's estimates merely represent their view of how hard items will be, taking into consideration the team's natural behavior (such as over- and under-estimation), as well as organizational overhead, etc. The velocity is all that matters from a planning perspective.

Since the units are not related to time, most teams now choose to use story points as their estimation unit. A story point is an arbitrary number that measures the complexity of one story relative to others. In

effect, story points clearly break the mental link with time.

▼ Inaccurate estimates are good, as long as they are equally inaccurate

For a team's velocity to reach a stable state, the team must estimate each backlog item with the same level of accuracy. At the risk of repeating the obvious, the goal of velocity is to be able to look at a backlog of not particularly well-understood stories, and understand how many sprints it will take to complete. This requires a similar level of uncertainty for all of the estimates in the backlog.

There is a counter-intuitive implication here — that teams should estimate each item once, and not change that estimate even if they discover new information about the item that makes them feel their Original Estimate was wrong. If the team were to go ahead and update estimates, this 'discovery of new information' will happen regularly. This leads to the backlog having some items that have higher accuracy, but most that don't. This would pollute velocity because sprints with a larger percentage of high accuracy estimates will complete a different number of units compared to those with a lower percentage of high accuracy estimates. As a result, the velocity could not be used for its primary purpose — that is, for estimating the number of sprints it will take for a team to complete a set of not-well-understood stories in the backlog. Therefore, it's critical to use the first estimates so that the team's velocity realistically represents their ability to *complete* a certain number of units of not-well-understood work far ahead into the future.

▼ But what about when teams realize they've gotten it wrong?

Consider the following scenario:

- Issue X has an Original Estimate of 5 days.
- Before the next sprint is planned, the team realizes that the Original Estimate was too optimistic, and that the issue actually takes 15 days.

Some people would argue that using the Original Estimate will endanger the sprint's success, because the team will take in what they think is 5 days of work into the next sprint when it's actually 15 days of work.

However, the inaccurate estimate of 5 days is unlikely to be an isolated occurrence. In fact, the estimates are always going to be wrong (some very little, some wildly so). This will often be discovered after the sprint has started rather than before. As long as the team estimates the same way across the whole backlog, this will work itself out over time. For example, if they always underestimate, they may find that for a 10-day sprint with 4 team members, they can only really commit to 20 days of their estimation unit. If they have established a stable velocity, then this has no effect. From a planning perspective, we can still reliably estimate how much work we'll get done in upcoming Sprints.

▼ But doesn't that break sprint commitment?

When the team is about to start a sprint, they can use the velocity as an indication of items from the backlog that they can realistically complete. The velocity here is based on the number of items they have successfully completed in the past. However, some people may question how this can be right when the Original Estimates won't include information about work that may have already been done, or information about how hard a particular item of work is.

As an example, consider the following scenario:

- An issue has an Original Estimate of 10 days.
- The team works 5 days on the issue in the current sprint.
- The team discovers a bad bug somewhere else in the project, and they decide that fixing that bug in the current sprint is far more important than completing issue X as planned.
- The sprint gets finished, and the issue returns to the backlog.

In the next sprint, the team would be tempted to update the estimate for the issue to 5 days, and use that to make their decision whether or not to include it in the sprint. The implication is that they might not include enough work in the next sprint if they used the issue's Original Estimate of 10 days. However, the reason that the task was not completed previously is because of unplanned work — and it's unrealistic to assume that this won't happen again in the future, perhaps even in the next sprint. Thus, the 10-day estimate is a realistic number to use in the absence of certainty. As a result, the cost of the unplanned work that may happen is eventually accounted for in the Original Estimate. Even if the work does turn out to be insufficient for the next sprint, the team will correct that by dragging more work into the sprint.

In the same example, consider if this were the only issue in that sprint and will be the only issue in the next. If the issue is completed in the second sprint, and we use the Remaining Estimate, then the velocity

will be $(0d + 5d) / 2 = 2.5d$. However, the team can clearly complete more work than that in future sprints. If we use the Original Estimate, then the velocity will be $(0d + 10d) / 2 = 5d$. The use of the Original Estimate accounts for the fact that the team cannot commit to 10d in every sprint because unplanned work will likely make that impossible. It also realistically accounts for the fact that unplanned work will not happen in every sprint.

▼ Why not estimate on sub-tasks and roll that up for Velocity and Commitment?

Many teams break down stories into sub-tasks shortly before the sprint begins so they can use the stories for tracking. This raises the possibility of using the sum of the estimates on the sub-tasks as a way to decide which issues to commit to in the sprint (and potentially for velocity).

As described above, tracking is really a separate process from estimation and velocity. The estimates that are applied to sub-tasks clearly have higher accuracy than those that were originally applied to the story. Using them for velocity would cause the velocity to have both high and low accuracy estimates, making it unusable for looking further out in the backlog where stories have only low accuracy estimates.

In addition, only items at the top of the backlog are likely to have been broken into tasks. Using task estimates for velocity means that the velocity value could only predict the time to complete the backlog up to the last story that has been broken into tasks.

Lastly, using sub-task roll-up to decide sprint commitment is risky because, unlike velocity value, it doesn't consider the overhead of unplanned work and interruptions.

▼ Story points are highly recommended — but use what works for your team

More and more industry leaders are moving away from hour estimates, and are now using the story point approach. This makes sense because in a sprint, the main questions to be answered are:

- How much work can we realistically commit to completing this sprint?
- How long will this part of the backlog take to deliver?

The story point approach based on original estimates can deliver the answers to these questions without the anxiety around 'accuracy' that teams feel when asked to estimate in hours.

The JIRA Software team itself uses the approach described in this article, and has established a reliable velocity that we use to plan work months in advance — even when new work has been encountered during those months. We recommend this approach because while it is sometimes counter-intuitive, it is also powerful, fast, and simple.

All of that said, one of the key precepts of agile is finding the way that works for you. So JIRA Software does support the alternatives described above, including the use of remaining estimates for sprint commitment, hours for estimation, and hour estimates on sub-tasks.

Flagging an issue

You can flag an issue to indicate that it's important. The card of a flagged issue is displayed in yellow in both Backlog and Active sprints, with the 'flag' icon replacing the Priority icon.

Sprint 6 11 issues
19/Jun/15 10:05 PM • 03/Jul/15 10:05 PM
Linked pages
3 6 7
...
TIS-8 Requesting available flights is now taking > 5 seconds 2.0 SeeSpaceEZ Plus
TIS-56 Add pointer to main css file to instruct users to create 2.0 Large Team Support
TIS-45 Email non registered users to sign up with Teams In 3.0 Large Team Support
TIS-49 Draft network plan for Mars Office 2.1 Local Mars Office
TIS-68 Homepage footer uses an inline style - should use a class

Related pages:

- Using the backlog
- Using active sprints
- Editing and collaborating on issues

Flagging or unflagging an issue

1. Click **Boards** (in header) > select your desired board.
2. Click either **Backlog**, **Active sprints**, or **Kanban board**.
3. Click the issue that you want to flag or unflag.

4. In the Issue Detail View, select **Add flag** or **Remove flag** from the 'cog' drop-down. You can also right-click on the issue > **Add flag** or **Remove flag**.

You can also add a comment when you're adding a flag to or removing a flag from an issue. You may want to do this to indicate your reason for adding or removing the flag.

In the Issue Detail View, select **Add flag and comment** or **Remove flag and add comment** from the 'cog' drop-down. You can also right-click on the issue > **Add flag and comment** or **Remove flag and add comment**. After adding your comment, it will appear in the Comments section of the issue, with an indication that a flag was added or removed accordingly.

Searching for flagged issues

The flag for an issue is stored in a custom checkbox field named "Flagged", which has only one value: Impediment.

This means that you can use this JQL query to find flagged issues: `Flagged = Impediment`

Ranking an issue

Rank your issues to organize the tasks in a sprint more effectively. By ranking issues, you actually arrange issues according to their relative importance or urgency. For example, you have two issues that are of 'High' priority. With JIRA Software ranking, you can choose which of these two issues have a higher priority or ranking than the other.

When you rank an issue, you also change its relative priority in its current column and swimlane.

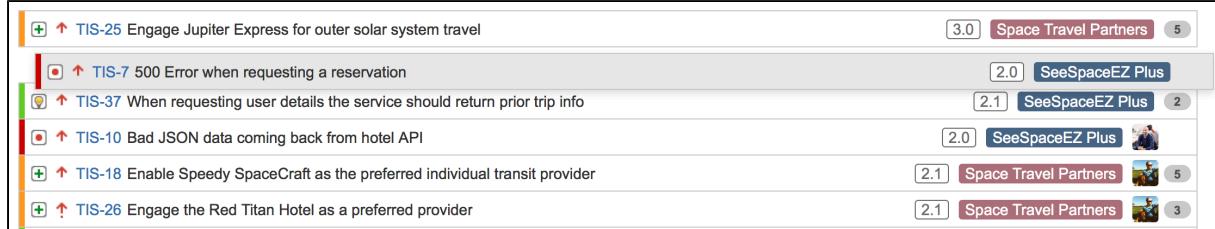
Related pages:

- Sorting by rank does not show expected values
- Enabling ranking
- Using active sprints

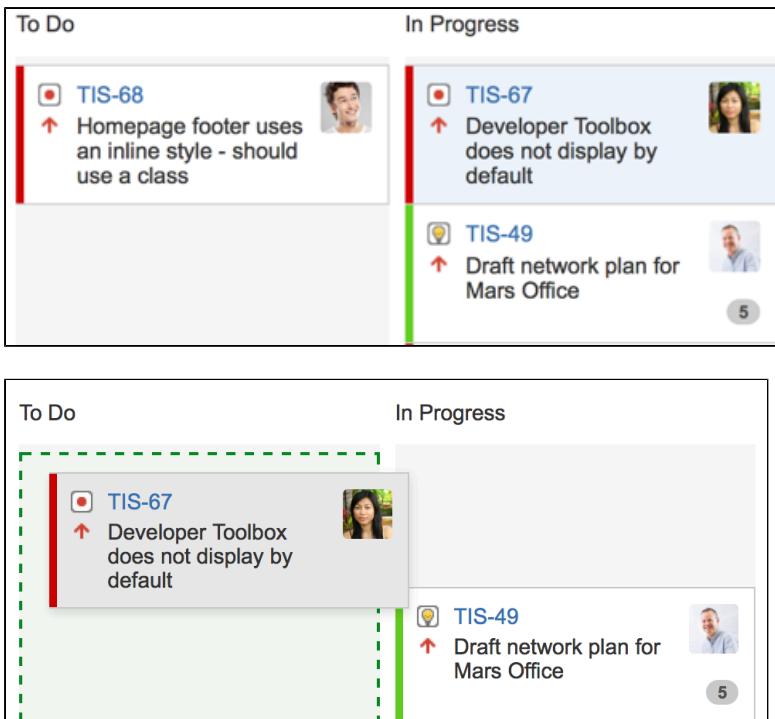
To rank an issue:

1. Click **Boards** (in header) > select your desired board.
2. Click **Backlog** or **Active sprints**.
3. Rank an issue by dragging and dropping it to a higher or lower position in the Backlog, or to a column in the Active sprints.

Screenshot: dragging and dropping an issue in the Backlog



Screenshot: dragging and dropping an issue to a column in the Active sprints



Tips and notes:

- You can use Keyboard Shortcuts '**s**' + '**b**' to move an issue to the bottom of its current column, or '**s**' + '**t**' to move it to the top.
- You can send multiple issues to the top or bottom of the backlog. Select the issues while pressing and holding down the 'Shift' or 'Ctrl' key, and then right-click > **Top of Backlog** or **Bottom of Backlog**.
- When you rank an issue that has sub-tasks, all of the sub-tasks are automatically moved with the issue.
- Sub-task issues can only be ranked in relation to their 'sibling' issues.

i Please note:

- You can only rank issues if ranking has been enabled — see [Enabling ranking](#).
- You can only rank issues if you have the 'Schedule Issue' and 'Edit Issue' permissions for the issue you want to move higher or lower on your board.

Transitioning an issue

Before you begin

Before you can transition issues to columns on a board, your JIRA administrator or board administrator will have already mapped [workflow status](#) to your board's columns. To know more about configuring these mappings, see [Configuring columns](#). Also, you can only transition an issue if you have [project permission](#) to move that issue to the status in the JIRA workflow to which the column is mapped.

Related pages:

- [Using active sprints](#)
- [Configuring columns](#)
- [Using the Simplified Workflow](#)

JIRA Software will respect your JIRA workflow configuration, and will trigger all your JIRA workflow validations, conditions, functions, and notifications when you transition issues (See [Workflows](#) for more information.) However, note that boards currently don't auto-assign issues to you.

To transition an issue to a different JIRA workflow status on a board:

1. Click **Boards** (in header) > select your desired board.
2. In the **Active sprints** (*Scrum board*) or **Kanban board** (*Kanban board*), transition an issue(s) by dragging and dropping the issue(s) from one column to another.
 - You can select multiple issues by using the **Shift** or **Ctrl** key while clicking the issues. However, you can't transition multiple issues by dragging them to another column. You can only move multiple issues within the same column.

- When you start dragging an issue, you can only transition the issue to columns whose backgrounds turn blue. When the issue is hovered over the target column, the blue background turns green.
- 3. If the target column is associated with more than one status, when you hover the issue over that column, each status will appear as a separate section with a dashed outline. Complete the transition to the appropriate status by dropping the issue onto the section of the column.
- 4. If a dialog box opens for your workflow transition status, you must complete all mandatory fields (indicated by a red asterisk) before submitting the form.

Note, if an issue has sub-tasks, and all sub-tasks have been completed, you need to resolve the issue itself. When you move the last sub-task to the 'Done' column, JIRA Software will prompt you to move the parent issue as well. If you resolved the sub-tasks in native JIRA Software instead, a button will be displayed on the parent issue the next time you visit the Active sprints page.

Printing issue cards

Whether you're planning work or working on issues on your Scrum or Kanban board, it may also be good to print out these issue cards. You and your team can use the printed cards on a physical board, which can be a replication of your board on JIRA Software. You can print a single issue card or multiple issue cards, if you want. You can also print all issue cards in your current board.

The printed issue cards include the following issue details:

- Summary
- Issue type
- Issue key
- Issue priority
- Estimate
- Assignee
- Epic (optional in Active sprints/Kanban board)
- Version (when printing from the Backlog)
- Up to 3 extra fields, depending on your card layout configuration

The printed issue cards fit on A4-, A3-, or Letter-sized pages in both portrait and landscape modes.

To print an issue card:

1. Click **Boards** (in header) > select your desired board.
2. Click **Backlog**, **Active sprints**, or **Kanban board**.
3. Right-click the issue that you want to print > **Print selected card**. The Print window will be displayed.
4. Select the card size for printing from the 'Card size' drop-down.
5. Click the **Print** button, and use the print functionality for your browser.

Tips:

- To print multiple issue cards, select the issues while pressing and holding down the 'Shift' or 'Ctrl' key, and then right-click > **Print selected cards**. The Print window displays the number of issue cards for printing.
- To print all issues in your current board, click **Board** > **Print cards**.
- When filters are active, only the visible issues will be printed.
- When printing from the Backlog of a Scrum board, you can filter the cards for printing by selecting one of the sprints from the 'Select' drop-down.

Next steps

 **Need help?** If you can't find the answer you need in our documentation, we have other resources available to help you. See [Getting help](#).

- Read [Customizing cards](#) for more information on configuring issue cards in JIRA Software.

Viewing the development information for an issue

If your administrator has connected JIRA Software to a compatible development tool, a **Development** panel will be displayed on the View Issue screen. Depending on which tools JIRA Software is connected to, the Development panel provides the following functionalities:

- Bitbucket Cloud or Bitbucket Server: view and create branches, view commits, and view and create pull requests
- FishEye/Crucible (Git/Subversion/Perforce/CVS): view branches (not create), view commits, and view and create reviews
- Bamboo: view the status of builds and deployments
- GitHub or GitHub Enterprise: view branches (not create), view commits, and view and create pull requests

On this page:

- Before you begin
- Make your development information available in JIRA Software
- View the Development panel
- Investigate and action the information

Before you begin

- Your administrator needs to have set up JIRA Software and your development tools correctly. Refer your administrator to [Integrating with development tools](#).
- You must have the 'View development tools' permission to be able to see the Development panel.

Make your development information available in JIRA Software

You must reference the issue key in your commit, branch, pull request, etc if you want it to display as a link in the Development panel. For information on how to reference issue keys correctly, see [Referencing issues in your development work](#).

View the Development panel

If everything has been set up correctly, you will see a Development panel on each of your issues, similar to the screenshot below. The Development panel provides you with just enough information to evaluate the status of an issue's development at a glance.

The screenshot shows a JIRA issue view for 'TIS-4'. The main content area displays the issue details, including its type (Story), priority (Major), and various versions and components. The 'Development' panel is visible on the right side, containing sections for People, Dates, Time Tracking, and Deployment. The 'People' section lists the assignee (Kevin Campbell) and reporter (Jennifer Evans). The 'Dates' section shows the issue was created on 25/May/14 at 11:01 AM and updated yesterday. The 'Time Tracking' section shows the estimated time as 'Not Specified', remaining time as '0m', and logged time as '3d 4h'. The 'Deployment' section shows 4 commits, 1 pull request (MERGED), and 1 build (green circle icon). A note at the bottom right of the Development panel says 'Development panel'.

Notes:▼ [What does the status lozenge next to the pull requests mean?](#)

The pull request(s) status in the Development panel is:

OPEN if there is at least one open pull request.

MERGED if there are no open pull requests, and at least one pull request has been merged.

DECLINED if there are no open or merged pull requests, and at least one pull request has been declined.

▼ [What does the status lozenge next to the reviews mean?](#)

The review(s) status in the Development panel is:

REVIEW if there is at least one review in 'Review' status. (yellow with black writing)

APPROVAL if there are no reviews in 'Review' status, and at least one review is in 'Approval' status. (black)

SUMMARIZE if there are no reviews in 'Review' or 'Approval' statuses, and at least one review is in 'Summarize' status. (black)

REJECTED if there are no reviews in 'Review', 'Approval', or 'Summarize' statuses, and at least one review is in 'Rejected' status. (red)

APPROVAL all reviews are in 'Closed' status.

Note, 'Draft' and 'Abandoned' reviews are not shown.

▼ [What does the status lozenge next to the builds mean?](#)

The build(s) status in the Development panel is:



if all the different builds (for example, unit tests, functional tests, deploy to staging) have passed.



if at least one run failed for any build by any linked build server.

Investigate and action the information

If you want to investigate something in the Development panel, you can click the item to display a dialog with more details. The dialog provides links for you to action or drill into. Note, you may be prompted to authenticate with the linked application first.

For example, say that the Development panel was showing this: [3 builds](#)



You could click [3 builds](#) to open a dialog showing which of the three builds are failing. If you wanted to investigate further, then you could click the build plan or build result to view it in Bamboo.

Here are the actions you can do via the Development panel:

▼ [Create feature branches](#)

BITBUCKET

Click **Create branch** in the Development panel to open your connected SCM, and start the process for creating a branch. If you have multiple applications connected, then you can choose where you'd like to create the branch. The key for the JIRA issue will be automatically added to the name of the branch.

▼ [See repository branches](#)

BITBUCKET

FISHEYEE

GITHUB

Click **n branches** in the Development panel to open a dialog in JIRA Software that shows the branches for the linked SCM. If JIRA Software has been linked to more than one SCM, a tab will show for each SCM product (e.g. Bitbucket). The branches will be grouped under each SCM in these tabs.

TIS-45: 2 branches				
Repository		Branch	Pull request	Action
 Apollo UI		feature/TIS-45-email-non-registered-users-to-sign	MERGED	Create pull request

- Click a repository or branch to open the linked SCM at the relevant repository/branch.
- Hover over a **Pull request** status (e.g. **MERGED**) to show a popup displaying a link to the pull request.
- Click **Create pull request** to create a pull request for the branch (to merge it back to master).

▼ See commits to repositories

[BITBUCKET](#) [FISHEYE](#) [GITHUB](#)

Click **n commits** in the Development panel to open a dialog in JIRA Software that shows the commits and related files for the linked SCM. If JIRA Software has been linked to more than one SCM, a tab will show for each SCM product (e.g. Bitbucket). The commits will be grouped under each SCM in these tabs. If a commit is greyed out, it has been merged through a pull request.

TIS-73: 2 unique commits (and 2 duplicates)					
Show all files					
Author Commit Message			Date	Files	
 3047fd41d44	TIS-73	Fix the highlighting	22/Aug/14	1 file	
 9d5d94c1e26	TIS-73	License update and minor fix	22/Aug/14	2 files	

- Click a repository or commit to open the SCM at the relevant repository/branch.
- If JIRA Software is linked to more than one SCM, the dialog may show duplicate commits across tabs. For example, you may have pointed FishEye and Bitbucket to the same repository.
- (*Bitbucket only*) If the commits belong to a fork of a repository, they will be grouped under the fork. The fork will also have a link to the original repository, "**Fork of <repository name>**".

▼ Create pull requests

[BITBUCKET](#) [GITHUB](#)

See the 'See repository branches' section above.

▼ See the status of pull requests

[BITBUCKET](#) [GITHUB](#)

Click **n pull requests** in the Development panel to open a dialog on JIRA Software that shows the pull requests for the linked SCM(s). If JIRA Software has been linked to more than one SCM, a tab will show for each SCM product (e.g. Bitbucket). The pull requests will be grouped under each SCM in these tabs.

TIS-57: 1 pull request					
ID	Title	Status	Author	Reviewer	Updated
#1	Bugfix/TIS-57 buttons need to be red	OPEN		 	12/Aug/14

- Click a pull request to open it in the linked SCM.
- Hover over a user icon to show the user's name.

▼ See the status of reviews

[CRUCIBLE](#)

Click **n reviews** in the Development panel to open a dialog in JIRA Software that shows the reviews.

TIS-57: 1 review		Status	Author	Reviewers	Date
ID	Title				
CR-4	TIS-57 Changed button background to red for toolbar	OPEN		 	Due 24/Sep/13 10:00 AM

- Click a review to open it in Crucible.

▼ See the status of builds

BAMBOO

Click **n builds** in the Development panel to open a dialog in JIRA Software that shows the builds.

TIS-68: 3 builds			
Teams In Space			
Plan	Latest build	Result	Completed
Developer Toolbox Functional Tests > bugfix-TIS-68-homepage-f...	#2	Passed in under 1 second	31/Jan/14
Developer Toolbox Checkstyle > bugfix-TIS-68-homepage-f...	#2	Failed in under 1 second	31/Jan/14
Developer Toolbox Browser Tests > bugfix-TIS-68-homepage-f...	#6	Failed in under 1 second	31/Jan/14

- Click a plan or build result to view it in Bamboo.

▼ See the status of deployments

BAMBOO

Click **Deployed (to environment)** in the Development panel to open a dialog in JIRA Software that shows the deployment.

TIS-59: 2 deployments			
Teams In Space - Apollo UI Commits were in release 1.3_RC7			
Environment	Status	Latest release	Last deployed
Production		1.3.2	16/Sep/13
QA		1.3.3	Yesterday

- Click an environment or release to view it in Bamboo.

Referencing issues in your development work

If your administrator has connected JIRA Software to your development tools, you can easily synchronize your development work with your issues — just reference an issue key(s) in your commits, branches, pull requests, etc, and you will enable the following:

- Show links to your development work on the issue in a **Development** panel
- Automatically transition the issue to a new status, if your administrator has set up **workflow triggers**
- Collate your issues and development work on the **Version details** page, which is used to track and release the version

Note, when you reference your issue key(s), your connected development tools will also have links back to the relevant issues.

The table below describes how to reference an issue key in a commit, branch, pull request, or review. In all cases, **the issue key must conform to the default JIRA key format** – that is, two or more uppercase letters ([A–Z] [A–Z] +), followed by a hyphen and the issue number. For example, ABC-123.

Event	Dev Tool	Instructions
-------	----------	--------------

Create commit	Bitbucket, GitHub, GitHub Enterprise, FishEye	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	Bitbucket, GitHub, GitHub Enterprise, FishEye	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	Bitbucket, GitHub, GitHub Enterprise	Do at least one of the following: <ul style="list-style-type: none"> • Include a commit in the pull request that has the issue key in the commit message. Note, the commit cannot be a merge commit. • Include the issue key in the pull request title. • Ensure that the source branch name includes the issue key. 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	Crucible	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.
Create build plan	Bamboo	A build is automatically linked to an issue if one of the build's commits includes the issue key in its commit message. The issue key must be included in the commit to activate this feature.
Create deployment project	Bamboo	A deployment to an environment, such as Production or testing, is linked to an issue if a commit associated with the deploy contains the issue key in its commit message. The issue key must be included in the commit to activate this feature.

Processing issues with 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, called Smart Commits, in your commit messages.

You can:

- comment on issues
- record time tracking information against issues
- transition issues to any status defined in the JIRA Software project's workflow.

There are other commands available if you use Crucible for software reviews. See [Using Smart Commits](#) in the Crucible documentation.

A single Smart Commit command cannot span more than one line (i.e. you cannot use carriage returns in the commit message), but you can add multiple commands to the same line, or multiple commands on separate lines. See [this example](#) below.

On this page:

- Smart Commit commands
- Advanced examples
- Get Smart Commits working
- Notes

Smart Commit commands

The basic command line syntax for a Smart Commit message is:

```
<ignored text> <ISSUE_KEY> <ignored text> #<COMMAND> <optional  
COMMAND_ARGUMENTS>
```

Any text between the issue key and the Smart Commit command is ignored.

There are three Smart Commit commands you can use in your commit messages:

- [comment](#)
- [time](#)
- [transition](#)

Comment

Description	Adds a comment to a JIRA Software issue.
Syntax	<ignored text> ISSUE_KEY <ignored text> #comment <comment_string>
Example	JRA-34 #comment corrected indent issue
Notes	<ul style="list-style-type: none"> • The committer's email address must match the email address of a single JIRA Software user with permission to comment on issues in that particular project.

Time

Description	Records time tracking information against an issue.
Syntax	<ignored text> ISSUE_KEY <ignored text> #time <value>w <value>d <value>h <value>m <comment_string>
Example	JRA-34 #time 1w 2d 4h 30m Total work logged
Notes	<p>This example records 1 week, 2 days, 4 hours and 30 minutes against the issue, and adds the comment 'Total work logged' in the Work Log tab of the issue.</p> <ul style="list-style-type: none"> • Each value for w, d, h and m can be a decimal number. • The committer's email address must match the email address of a single JIRA Software user with permission to log work on an issue. • Your system administrator must have enabled time tracking on your JIRA Software instance.

Workflow transitions

Description	Transitions a JIRA Software issue to a particular workflow state.
Syntax	<ignored text> ISSUE_KEY <ignored text> #<transition_name> <comment_string>
Example	JRA-090 #close Fixed this today

Notes	<p>This example executes the close issue workflow transition for the issue and adds the comment 'Fixed this today' to the issue.</p> <p>You can see the custom commands available for use with Smart Commits by visiting the JIRA Software issue and seeing its available workflow transitions:</p> <ol style="list-style-type: none"> 1. Open an issue in the project. 2. Click View Workflow (near the issue's Status). <p>The Smart Commit only considers the part of a transition name before the first space. So, for a transition name such as <code>finish work</code>, then specifying <code>#finish</code> is sufficient. You must use hyphens to replace spaces when ambiguity can arise over transition names, for example: <code>#finish-work</code>.</p> <p>If a workflow has two valid transitions, such as:</p> <ul style="list-style-type: none"> • Start Progress • Start Review <p>A Smart Commit with the action <code>#start</code> is ambiguous because it could mean either of the two transitions. To specify one of these two transitions, fully qualify the transition you want by using either <code>#start-review</code> or <code>#start-progress</code>.</p> <ul style="list-style-type: none"> • When you resolve an issue with the <code>#resolve</code> command, you cannot set the Resolution field with Smart Commits. • If you want to add a comment during the transition, the transition must have a screen associated with it. • The committer's email address must match the email address of a single JIRA Software user with the appropriate project permissions to transition issues.
--------------	--

Advanced examples

Multiple commands over multiple lines on a single issue

Syntax	<code><ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS></code>
Commit message	JRA-123 #comment Imagine that this is a really, and I mean really, long comment #time 2d 5h
Result	Adds the comment 'This is a really, and I' (but drops the rest of the comment) and logs 2 days and 5 hours of work against issue JRA-123.

Multiple commands on a single issue

Syntax	<code><ISSUE_KEY> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS></code>
Commit message	JRA-123 #time 2d 5h #comment Task completed ahead of schedule #resolve
Result	Logs 2 days and 5 hours of work against issue JRA-123, adds the comment 'Task completed ahead of schedule', and resolves the issue.

A single command on multiple issues

Syntax	<ISSUE_KEY1> <ISSUE_KEY2> <ISSUE_KEY3> #<COMMAND> <optional COMMAND_ARGUMENTS> etc
Commit message	JRA-123 JRA-234 JRA-345 #resolve
Result	Resolves issues JRA-123, JRA-234 and JRA-345. Multiple issue keys must be separated by whitespace or commas.

Multiple commands on multiple issues

Syntax	<ISSUE_KEY1> <ISSUE_KEY2> ... <ISSUE_KEYn> #<COMMAND_1> <optional COMMAND_1_ARGUMENTS> #<COMMAND_2> <optional COMMAND_2_ARGUMENTS> ... #<COMMAND_n> <optional COMMAND_n_ARGUMENTS>
Commit message	JRA-123 JRA-234 JRA-345 #resolve #time 2d 5h #comment Task completed ahead of schedule
Result	Logs 2 days and 5 hours of work against issues JRA-123, JRA-234 and JRA-345, adds the comment 'Task completed ahead of schedule' to all three issues, and resolves all three issues. Multiple issue keys must be separated by whitespace or commas.

Get Smart Commits working

It's easy to get Smart Commits working for your instance of JIRA Software:

Tool	Connection instructions
Bitbucket Server	Create an application link between JIRA Software and Bitbucket Server. See Linking Bitbucket Server to JIRA . Then, enable Smart Commits in JIRA Software. See Enabling DVCS Smart Commits .
FishEye	Create an application link between JIRA Software and FishEye. See Linking to another application . Then, enable Smart Commits in JIRA Software. See Enabling DVCS Smart Commits .
Crucible	Create an application link between JIRA Software and Crucible. See Linking to another application . Then, enable Smart Commits in JIRA Software. See Enabling DVCS Smart Commits .
Bitbucket Cloud	First, link your JIRA Software and Bitbucket accounts. See Linking Bitbucket Cloud and GitHub accounts to JIRA Software . Then, enable Smart Commits in JIRA Software. See Enabling DVCS Smart Commits .
GitHub	First, link your JIRA Software and GitHub accounts. See Linking Bitbucket Cloud and GitHub accounts to JIRA Software . Then, enable Smart Commits in JIRA Software. See Enabling DVCS Smart Commits .

Notes

- Smart Commits only support the default JIRA Software issue key format. This format is two or more uppercase letters, followed by a hyphen and the issue number, for example JRA-123.
- A DVCS such as Git includes a user's email address in the commit data. Users configure this email address in their local system. Smart Commits requires that this email address match *exactly one* email address in the JIRA Software user base. If the email address matches to multiple users in JIRA Software, or the user does not have permissions for the requested action, the Smart Commit action will fail. The commit itself will succeed however, and will show on the issue. Mismatched email addresses is a common reason why Smart Commits fail to work as expected. If a Smart Commit fails, JIRA Software sends an email notification to either the JIRA Software user, or to the DVCS user (if a JIRA Software user can't be identified). In rare cases, JIRA Software doesn't have either of these email addresses, and the Smart Commit fails silently.
- Smart Commits relies on the [JIRA DVCS Connector Plugin](#) when your repositories are hosted in Bitbucket Cloud or GitHub. The plugin is bundled with JIRA Software, but if necessary, a JIRA administrator can install it directly from within the JIRA administration area. Go to **Add-ons > Find new Add-ons**. See [Installing add-ons](#).

Configuring dashboards

Your dashboard is the main display you see when you log in to your project. You can create multiple dashboards for different projects, or multiple dashboards for one big project. Each project has a default dashboard, or you can create a personal dashboard and add gadgets to keep track of assignments and issues you're working on. Dashboards are designed to display gadgets that help you organize your projects, assignments, and achievements in different charts.

You can see all dashboards by selecting the **Dashboards** drop-down from your JIRA application header.

On this page:

- About the default dashboard
- Creating a dashboard
- Managing dashboard s and permission s
- Sharing and editing your dashboard
- Adding favorite dashboard s
- Note on dashboard permission s
- Setting up a Wallboard

About the default dashboard

The gadgets on the default dashboard can be reordered and switched between the left and right columns. Additional gadgets can also be added, while 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 dashboard. 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.

Creating a dashboard

You can easily create and customize your own dashboard to display the information you need. Note that only administrators can customize the default dashboard for your project.

1. At the top right of the Dashboard, click the **Tools** menu.

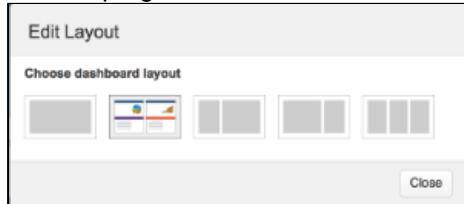
2. Select either **Create Dashboard** to create a blank dashboard, or **Copy Dashboard** to create a copy of the dashboard you are currently viewing.
3. Name and describe your dashboard.
4. Fill out the rest of the fields as applicable.
5. Click **Add**.

By default, sharing is set to private if you have not specified a personal preference. You can adjust this setting in the sharing preferences in your [user profile](#), and change dashboard permissions at any time in the Manage Dashboards page.

Choosing a dashboard layout

To choose a different layout for your dashboard page (e.g. three columns instead of two):

1. At the top right of the Dashboard, click the '**Edit Layout**' link. A selection of layouts will be displayed:



2. Select your preferred layout.

Managing gadgets

To get the most out of your dashboard, including adding, rearranging, removing, and configuring gadgets, see [Adding and customizing gadgets](#).

Managing dashboards and permissions

You can edit, delete, copy, mark favorites, and share your dashboards from the Manage Dashboards page.

1. Select **Dashboards > Manage Dashboards**.
2. Choose the dashboard.

Sharing and editing your dashboard

You can edit the details for your dashboard, and restrict or share with other users according to the permissions that are set. In addition, you can see all the dashboards you've created, any public dashboards, and any shared dashboards.

1. Click  > **Edit/Share > Add sharing permissions**.
2. Edit the settings.

Adding favorite dashboards

If you find a dashboard you like, click the star icon next to its name to add it to your favorite dashboards list. You can also add the default dashboard to your favorites list so it's easily available to you.

Name	Owner	Shared With
IT Support	Lily Williams (lwilliams)	Private Dashboard

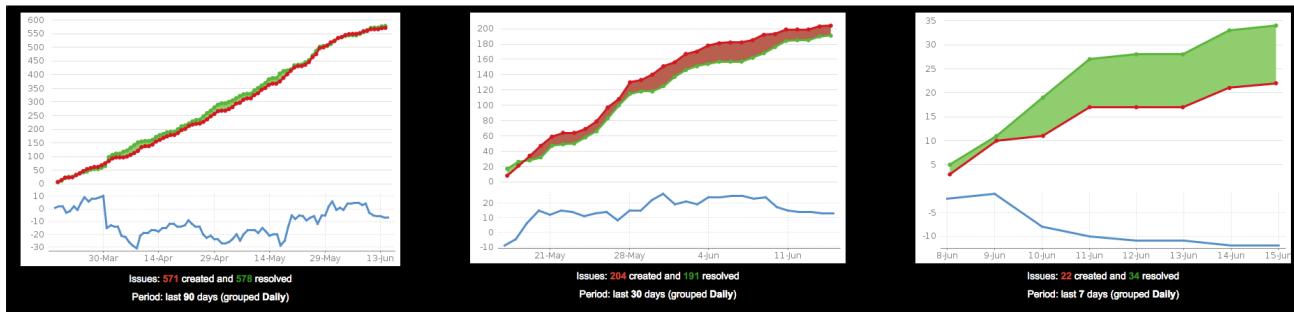
Note on dashboard permissions

JIRA administrators, as set in global permissions, can manage their users' shared dashboards in the **Shared dashboards** menu. Administrators can also change the ownership of a dashboard if the creator is unable to maintain the dashboard or its gadgets. See [Managing shared dashboards](#) for more information.

Setting up a Wallboard

Turn any JIRA application dashboard into a wallboard by plugging your computer into a TV monitor. The Wallboard is a dashboard **gadget** that acts as an information radiator to provide instant visual insight into project progress and team accomplishments. With your favorite dashboard selected, click **Tools > View as Wallboard**. The dashboard will appear against a black background, and will rotate gadgets if the user enables the slideshow option.

The Wallboard below shows the same **Created vs. Resolved Issues** gadgets and data above.



Adding and customizing gadgets

Adding a gadget to a dashboard

You can add gadgets to your own personal dashboard(s). To add a gadget to the default dashboard for your JIRA application, you must be a JIRA admin.

Some applications allow dashboards that are shared by groups of people. If you have permission to update a shared dashboard, the other people sharing the dashboard will see your changes, too.

1. Go to the dashboard by selecting the **Dashboard** link in the header.
2. On the dashboard, Click **Add Gadget**.
3. Use the gadget wizard to choose the gadgets you want to add. You can see a list of these gadgets in [Gadgets for JIRA applications](#).

For more information about managing dashboards, see [Configuring dashboards](#).

Customizing how gadgets look

There are a few ways you can customize the view of gadgets in a dashboard:

To	Do this
Expand or collapse gadgets	Use the  button in the gadget header.

Expand a gadget to take up the entire dashboard	Use the  button in the gadget header. ▼ Notes... This view often provides more functionality than is available in the standard view of the gadget. Only some gadgets provide the maximized or canvas view. The canvas view setting is stored in a cookie, and is not saved to the dashboard server.
Rearrange gadgets	Use the  button in the gadget header.
Customize the gadget frames	Use the  button in the gadget header.
Delete a gadget	

Custom gadgets

You need administrator privileges to add a gadget to the list of available gadgets. 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. This functionality is only available for the Server version of applications; if you would like to add an Atlassian gadget to a directory in your Cloud site, please contact Atlassian Support.

Gadgets for JIRA applications

Gadgets let you customize the information that appears on dashboards in JIRA applications (or on your wallboards, if you use dashboards for that purpose). This page lists all of the gadgets available for JIRA applications and which ones they're available for.

Gadget	JIRA Core	JIRA Software	JIRA Service Desk	Use it to
Activity Stream	✓	✓	✓	See the activity in your instance: it's like a Facebook feed for your instance!
Sprint Burndown Gadget		✓		See the burndown for a given sprint in a handy line chart. ▼ Notes... <ul style="list-style-type: none"> • The vertical axis represents your configured estimation statistic. • The gadget will only display sprints that have not been completed.

Sprint Health Gadget				<p>Seeing a summary of the issues in a sprint in a handy color-coded bar graph.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> The colors in this gadget match the colors in your column configuration. The work completed is calculated based on the estimation statistic used for your board. This is reflected by the green part of the progress bar. For example, if you have 50 story points in a sprint and you have 3 issues with 10 story points that have been resolved, the 'Work complete' will be 20% (i.e. 10 out of 50 story points). The gadget won't reflect the progress from work logged in the 'Remaining Estimate' and 'Time Spent' fields in JIRA. Adding or removing an issue from a sprint, after it has started is considered a change of scope. The percentage is calculated using the statistic that is configured for the board. For example, if you started a sprint with 50 story points and add an issue with 5 story points, the Sprint Health gadget would show a 10% scope change. If you add/remove issues that don't have estimates, the scope change will not be altered. If you're using Time Tracking, Scope Change will not be shown. The "blocker" field counts all blockers that are in 'To Do' or 'In Progress'. <p>See Configuring estimation and tracking for more information.</p>
Version Report				Track the projected release date for a version. This helps you monitor whether the version will release on time, so you can take action if work is falling behind.
Agile Wallboard Gadget				Know how you're tracking with an agile board displayed on your wallboard (or dashboard).
Assigned to Me				Quickly see all the unresolved issues assigned to you.
Average Age Chart				<p>Want to know the average age of unresolved issues? This gadget tells you just that.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> The report is based on your choice of project or issue filter, and your chosen units of time (i.e. hours, days, weeks, months, quarters or years). For the purposes of this gadget, an issue is defined as unresolved if it has no value in the system resolution field. The age of an issue is the difference between the current date and the created date of the issue.
Average Number of Times in Status				Displays the average number of times issues have been in a status.
Average Time in Status				Displays the average number of days issues have spent in status.

Bamboo Charts				<p>Checking out Bamboo plan stats in your dashboard.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> Your JIRA administrator must have configured the Bamboo plugin on your JIRA server, if you want to add the Bamboo Charts gadget to your dashboard. If you have added multiple Bamboo servers in JIRA there will be one Bamboo Charts gadget available per server, e.g. 'Bamboo Charts Gadget from http://172.20.5.83:8085', 'Bamboo Charts Gadget from http://172.19.6.93:8085', etc. When you add this gadget to your JIRA dashboard, you may see a message similar to this: <div style="border-left: 1px solid black; padding-left: 10px;"> <i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i> </div> <p>To fix this problem, you will need to configure your Bamboo site to allow JIRA to draw information from it via gadgets on the JIRA dashboard. To do this, your JIRA administrator first needs to define your JIRA site as an OAuth consumer in Bamboo. You will then be required to perform a once-off authentication before your gadget will display correctly.</p>
Bamboo Plan Summary Chart				<p>Seeing a graphical summary of a Bamboo build plan.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> Your JIRA administrator must have configured the Bamboo plugin on your JIRA server, if you want to add the Bamboo Charts gadget to your dashboard. If you have added multiple Bamboo servers in JIRA there will be one Bamboo Charts gadget available per server, e.g. 'Bamboo Charts Gadget from http://172.20.5.83:8085', 'Bamboo Charts Gadget from http://172.19.6.93:8085', etc. When you add this gadget to your JIRA dashboard, you may see a message similar to this: <div style="border-left: 1px solid black; padding-left: 10px;"> <i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i> </div> <p>To fix this problem, you will need to configure your Bamboo site to allow JIRA to draw information from it via gadgets on the JIRA dashboard. To do this, your JIRA administrator first needs to define your JIRA site as an OAuth consumer in Bamboo. You will then be required to perform a once-off authentication before your gadget will display correctly.</p>

Bamboo Plans				<p>Seeing a list of all plans on a particular Bamboo server and each plan's current status.</p> <p> Notes...</p> <ul style="list-style-type: none"> Your JIRA administrator must have configured the Bamboo plugin on your JIRA server, if you want to add the Bamboo Charts gadget to your dashboard. If you have added multiple Bamboo servers in JIRA there will be one Bamboo Charts gadget available per server, e.g. 'Bamboo Charts Gadget from http://172.20.5.83:8085', 'Bamboo Charts Gadget from http://172.19.6.93:8085', etc. When you add this gadget to your JIRA dashboard, you may see a message similar to this: <div style="border-left: 1px solid #ccc; padding-left: 10px;"> <i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i> </div> <p>To fix this problem, you will need to configure your Bamboo site to allow JIRA to draw information from it via gadgets on the JIRA dashboard. To do this, your JIRA administrator first needs to define your JIRA site as an OAuth consumer in Bamboo. You will then be required to perform a once-off authentication before your gadget will display correctly.</p>
Bubble Chart				<p>Visually track the correlation of issues in a project or filter during a configured period, based on the following details:</p> <ul style="list-style-type: none"> number of days the issues have been open number of comments the issues have number of participants or votes the issues have <p> Notes...</p> <ul style="list-style-type: none"> The horizontal axis represents the number of days the issues have stayed open, while the vertical axis represents the number of comments the issues have. The bubble colors also indicate the correlation between days open and number of comments – with the color green indicating low values and the color red indicating high values. Only the first 200 matching open issues are displayed on the Bubble Chart. You can configure the following settings for the Bubble Chart: <ul style="list-style-type: none"> The period during which the issue comments are considered recent The basis of the bubble size, either participants or votes Automatic refresh of Bubble Chart data every 15 minutes Relative coloring to distinguish issues receiving more comments from issues receiving fewer comments Logarithmic scale (default is linear scale) to distribute the bubbles from each other accordingly. We recommend that you use the logarithmic scale if your Bubble Chart contains a large range of data.

Clover Coverage				<p>Seeing the Clover coverage of plans from a particular Bamboo server.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> Your JIRA administrator must have configured the Bamboo plugin on your JIRA server, if you want to add the Bamboo Charts gadget to your dashboard. If you have added multiple Bamboo servers in JIRA there will be one Bamboo Charts gadget available per server, e.g. 'Bamboo Charts Gadget from http://172.20.5.83:8085', 'Bamboo Charts Gadget from http://172.19.6.93:8085', etc. When you add this gadget to your JIRA dashboard, you may see a message similar to this: <i>The website (container) you have placed this gadget on is unauthorized. Please contact your system administrator to have it approved.</i> <p>To fix this problem, you will need to configure your Bamboo site to allow JIRA to draw information from it via gadgets on the JIRA dashboard. To do this, your JIRA administrator first needs to define your JIRA site as an OAuth consumer in Bamboo. You will then be required to perform a once-off authentication before your gadget will display correctly.</p>
Created vs. Resolved Chart				<p>Checking your progress by seeing the number of issues created vs number of issues resolved over a given period of time.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> The chart is based on your choice of project or issue filter, and the chart can either be cumulative or not. An issue is marked as resolved in a period if it has a resolution date in that period. The resolution date is the last date that the Resolution field was set to any non-empty value.
Crucible Charts				<p>Seeing statistical summaries of your code reviews.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> Your JIRA administrator must have configured the FishEye plugin on your JIRA server, if you want to add the Crucible Charts gadget to your dashboard (not applicable to JIRA Cloud).
Days Remaining in Sprint Gadget				Countdown! See how many working days you have before the current sprint ends.
Favorite Filters				See a list of all the issue filters that have currently been added by you as a favorite filter.
Filter Results				Seeing the results of a specified issue filter on the dashboard.
FishEye Charts				Chart LOC data from a FishEye repository.

FishEye Recent Changesets	✓	✓	✓	<p>Get two charts about your repo in one: lines of code and commit activity.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> Your JIRA administrator must have configured the FishEye plugin on your JIRA server, if you want to add the Crucible Charts gadget to your dashboard (not applicable to JIRA Cloud).
Introduction	✓	✓	✓	<p>Say hello to users with a configurable message on the dashboard.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> The text/html displayed in the introduction gadget is configured by your JIRA administrator, through the JIRA configuration page.
Issue Statistics	✓	✓	✓	See the issues returned from a specified project or saved filter (grouped by a specified field).
Issues In Progress	✓	✓	✓	Time to work! See all issues that are currently in progress and assigned to you.
JIRA Issues Calendar	✓	✓	✓	<p>Generating a calendar-based view of due dates for issues and versions</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> The JIRA Calendar plugin is required for this gadget to be available.
JIRA Road Map	✓	✓	✓	See which versions are due for release in a given period, as well as a summary of the progress made towards completing the issues in the versions.
Labels Gadget	✓	✓	✓	Use this gadget to see a list of all the labels used in a given project.
Pie Chart	✓	✓	✓	See the issues returned from a specified project or issue filter, grouped by a specified field.
Quick Links	✓	✓	✓	Link to frequently-used searches and operations.
Recently Created Chart	✓	✓	✓	See the rate at which issues are being created, as well as how many of those created issues are resolved - all in a bar chart.
Resolution Time	✓	✓	✓	<p>Check trends in the average time taken to resolve issues.</p> <p>▼ Notes...</p> <ul style="list-style-type: none"> The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years). The 'Resolution Time' is the difference between an issue's Resolution Date and Created date. If a Resolution Date is not set, the issue won't be counted in this gadget. The Resolution Date is the last date that the system Resolution field was set to any non-empty value.

Text				<p>Display your specified HTML text on the dashboard.</p> <p> Notes...</p> <ul style="list-style-type: none"> • This gadget is only available if your JIRA administrator has enabled it. It is disabled by default because it is a potential security risk, as it can contain arbitrary HTML which could potentially make your JIRA system vulnerable to XSS attacks. • To enable the text gadget: 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 site. Enable the Text module in the Atlassian JIRA - Plugins - Gadgets Plugin (You need to select the System add-ons from the drop-down). • If you cannot enable the text gadget, please contact Atlassian Support for assistance.
Test Sessions				View a list of test sessions.
Time Since Chart				<p>See a bar chart showing the number of issues for which your chosen field (e.g. 'Created', 'Updated', 'Due', 'Resolved', or a custom field) was set on a given date.</p> <p> Notes...</p> <ul style="list-style-type: none"> • 'Resolved' here is the system Resolution Date field, which is the last date that the system Resolution field was set to any non-empty value. • The report is based on your choice of project or issue filter, and your chosen units of time (ie. hours, days, weeks, months, quarters or years).
Time to First Response				Displays the number of hours taken to respond to issues for a project or filter.
Two Dimensional Filter Statistics				See data based on a specified issue filter (For example, you could create a filter to retrieve all open issues in a particular project. You can then configure the gadget to display the statistical data on this collection of issues, in a table with configurable axes.
Voted Issues				See all the issues you've voted for.
Watched Issues				Seeing all the issues you're watching.
Workload Pie Chart				Displays the matching issues for a project or filter as a pie chart.

Managing your user profile

You can manage your JIRA settings (e.g. your password, email address, or the format in which you would like to receive email notifications) in your user profile. Your user profile also displays recent work in the Activity Stream, and contains useful shortcuts to issues you have been working on or reported.

To manage your user profile:

Choose **your user name** at top right of the screen, then choose **Profile**.

On this page:

- Editing your user details
- Changing your avatar
- Choosing your homepage
- Managing email notification s
- Managing your user preference s
- Managing service desk preference s
- Managing your OAuth and login tokens

Editing your user details

In the **Details** section on the **Summary** page, click the edit icon



at the top-right of the section to edit your display name, email address, and password. If your JIRA administrator has configured the user directory with external password management, the **Change Password** link will not be available.

Changing your avatar

Select



or your current avatar to change the image that appears next to your name in JIRA. If your administrator has [enabled Gravatar for user avatars](#), your Gravatar (i.e. the Gravatar associated with the email address in your user profile) will automatically be set as your user avatar. If Gravatar has been enabled, you will not be able to choose JIRA -specific user avatars and vice versa. using [Gravatar.com](#). If Gravatar has been disabled, you can choose your user avatar from the ones pre-packaged with JIRA or upload your own.

- Your cropped image is resized to 48x48 pixels before it is saved as your new custom user avatar.
- A separate 16x16 pixel version of your custom user avatar will be generated for use in comments.
- Custom user avatars can only be selected by the user who uploaded them.

Choosing your homepage

Your JIRA home page is the JIRA page you are presented with immediately after you log in.

You can configure the following JIRA pages as your JIRA home page:

- The Dashboard
 - The Issue Navigator
 - The Rapid Board (available if you're using JIRA Software)
1. Click on your **profile** icon at the top right of the screen.
 2. Select the appropriate home page option within the **My JIRA Home** section:
 - Dashboard
 - Issue Navigator
 - Rapid Board (available if you're using JIRA Software)
 3. **(Optional)** To verify that your JIRA home page has been reset, log out and log back in to JIRA again. You should be taken directly to the JIRA home page you selected in the previous step.
- i** Your page will be reloaded the JIRA home page you selected.

Managing email notifications

In the **Preferences** section on the **Summary** page, click the **edit** icon



at the top-right of the section to open the **Updated User Preferences** dialog box. You can then manage the following:

- Change the **Email Type** to change the format (plain text or HTML) in which JIRA sends its outgoing email notifications.
- In **My 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**).

Managing your user preferences

The global defaults for most of the user preferences below can be set by your JIRA administrator; however, you can override these default settings by changing the following:

- The **Page Size**, or number of issues displayed on each Issue Navigator page
- Your preferred **language** from the drop-down list. If you don't see your preferred language in the list, see [Translating JIRA](#) for more information.
- Your **time zone** specified in your profile doesn't match the time zone of the computer you are working on, JIRA will ask if you want to update this selected time zone setting. All time fields in JIRA will now be displayed in your preferred time zone.
- Choose the default **Sharing** setting for when you create new filters and dashboards, which can be either shared with all other users (**Public**) or restricted.
- Choose to enable or disable JIRA's keyboard shortcuts feature.
- Choose between allowing JIRA to make you an **autowatcher** of any issue that you create or comment on.

Managing service desk preferences

Service desk agents can enable or disable the **Pre-populated commenting** field by editing their user profiles. This setting can help save time by pre-filling conversation greeting text when agents comment on customer issues. When enabled the following text appears in the comment field and in the email notification sent to customers:

The screenshot shows the JIRA service desk comment editor. At the top, there are tabs for 'All', 'Comments', 'Work Log', 'History', and 'Activity'. Below the tabs, a message says 'There are no comments yet on this issue.' A large text area contains the pre-populated text: 'Hi <Reporter_name>, | - <Agent_name>'. Below this text area are 'Send response' and 'Cancel' buttons. At the bottom right, there are links for 'preview' and 'syntax help'.

Managing your OAuth and login tokens

An OAuth access token is issued by JIRA to give [gadgets](#) access to restricted data on an external, OAuth-compliant web application or website (also known as a "consumer"). Check out [Allowing OAuth access](#) for recommendations on when to issue or revoke OAuth access tokens.

If you are accessing your JIRA applications in a public environment, you can clear your login tokens by clicking the **Clear all Tokens** link in the Details section of your Profile.

Allowing OAuth access

About OAuth access tokens

OAuth access tokens allow you to:

- Use a JIRA gadget on an external, OAuth-compliant web application or website (also known as a 'consumer')
- Grant this gadget access to JIRA data which is restricted or privy to your JIRA user account.

Before you begin

Your JIRA administrator must establish an OAuth relationship with this external web application or instance by approving it as an OAuth consumer. For example, if you want to add a JIRA gadget to your Bamboo homepage and allow this gadget to access your restricted JIRA data, then your JIRA administrator must first approve Bamboo as an OAuth consumer.

On this page:

- [About OAuth access tokens](#)
- [Before you begin](#)
- [Issuing OAuth access tokens](#)
- [Revoking OAuth access tokens](#)

The JIRA gadget on the 'consumer' is granted access to your JIRA data via an 'OAuth access token', which acts as a type of 'key'. As long as the consumer is in possession of this access token, the JIRA gadget will be able to access JIRA data that is both publicly available and privy to your JIRA user account. You can revoke this access token at any time from your JIRA user account, otherwise, all access tokens expire after seven days. Once the access token is revoked or has expired, the JIRA gadget will only have access to publicly available data on your JIRA instance.

An OAuth access token will only appear in your user profile if the following conditions have been met:

1. Your JIRA Administrator has established an application link using OAuth between your JIRA instance and the consumer. JIRA Administrators should refer to [Using AppLinks to link to other applications](#).
2. You have accessed a JIRA gadget on a consumer and have allowed this gadget access to your JIRA data. See [Issuing OAuth access tokens](#) below for details on this process.

Screenshot: Viewing your OAuth Access Tokens

OAuth Access Tokens

You have allowed the following gadgets/applications to access JIRA data using your account:

Consumer	Consumer Description	Issued on	Expires on	Actions
Activity Stream	Atlassian RefImpl at http://localhost:8080/dashboards	02/10/2009	09/10/2009	Revoke OAuth Access Token
Created vs Resolved Chart	Atlassian RefImpl at http://localhost:8080/dashboards	02/10/2009	09/10/2009	Revoke OAuth Access Token

Issuing OAuth access tokens

An OAuth access token is issued by JIRA to provide one of its gadgets on a consumer, access to your JIRA data (that is, data which is restricted to your JIRA user account).

1. When you are using a JIRA gadget on a consumer (such as Bamboo) and this gadget requires access to your JIRA data, you will first be prompted to log in to JIRA (if you have not already done so).
2. Once you have logged in to JIRA, you will be prompted with a '**Request for Access**' message:

Screenshot: Request for Access Message

Request for Access

The application Bamboo would like to access your Atlassian JIRA account on your behalf. If you trust this application and would like to allow it access, click the 'Approve Access' button. An example of such access is a gadget running on another server.

By approving this request for access, you are allowing the application to **read** and **update** data using your username. The application will not have access to your password.

You can revoke this access at any time by going to the OAuth Access Tokens section of your user profile. [Learn more.](#)

[Approve Access](#)

[Deny Access](#)

At this point, JIRA is preparing to issue the JIRA gadget (on the consumer) with an OAuth access token.

3. To grant the gadget access to your JIRA data, click the '**Approve Access**' button. The consumer application will receive the OAuth access token from your JIRA instance. This access token is specific to this gadget and as long as the token resides with the gadget, your gadget will have access to your JIRA data.

Revoking OAuth access tokens

You can revoke an OAuth access token to deny a JIRA gadget on a consumer access to JIRA data which is restricted to your JIRA user account. You can only revoke OAuth access tokens that you have allowed JIRA to issue previously.

1. Choose **your user name** at top right of the screen, then choose **Profile**.
2. Click the '**Tools**' menu and select the '**View OAuth Access Tokens**' menu item.
3. The '**OAuth Access Tokens**' page will be displayed.

Screenshot: Viewing your OAuth Access Tokens

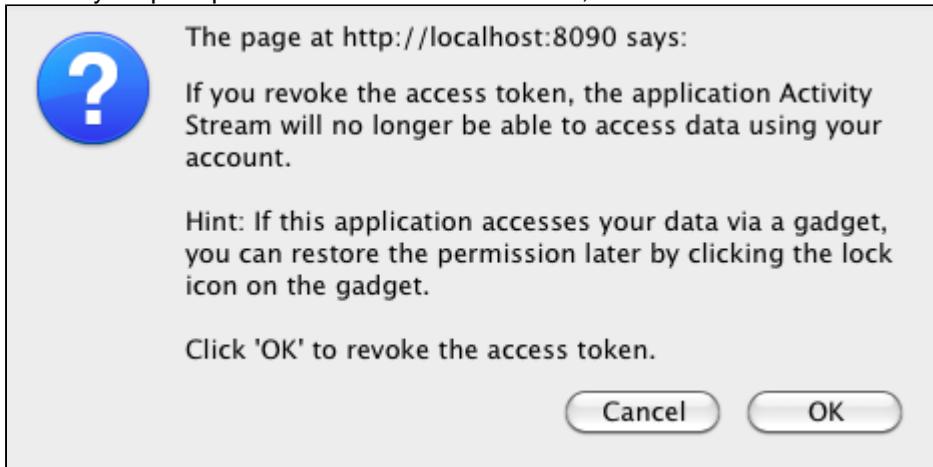
OAuth Access Tokens

You have allowed the following gadgets/applications to access JIRA data using your account:

Consumer	Consumer Description	Issued on	Expires on	Actions
Activity Stream	Atlassian RefImpl at http://localhost:8080/dashboards	02/10/2009	09/10/2009	Revoke OAuth Access Token
Created vs Resolved Chart	Atlassian RefImpl at http://localhost:8080/dashboards	02/10/2009	09/10/2009	Revoke OAuth Access Token

Your list of OAuth access tokens is presented in a tabular format, with each access token presented in separate rows and each property of these tokens presented in a separate columns. Refer to the **OAuth access token table details** section below for more information about this table.

4. Locate the JIRA gadget and its associated consumer application whose OAuth access token you wish to revoke and click its 'Revoke OAuth Access Token' link in the 'Actions' column.
5. You may be prompted to confirm this action. If so, click the 'OK' button.



The gadget's access token is revoked and the JIRA gadget on the consumer will only have access to publicly available JIRA data.

OAuth access token table details

Column name	Description
Consumer	The name of the JIRA gadget that was added on the consumer.
Consumer Description	A description of this consumer application. This information would have been obtained from the consumer's own OAuth settings when an OAuth relationship was established between JIRA and that consumer.  If the consumer is another Atlassian application, this information is obtained from the Consumer Info tab's 'Description' field of the OAuth Administration settings. The application's administrator can customize this Consumer Info detail.
Issued On	The date on which the OAuth access token was issued to the consumer by JIRA. This would have occurred immediately after you approved this gadget access to your JIRA data (privy to your JIRA user account).
Expires On	The date when the OAuth access token expires. This is seven days after the 'Issued On' date. When this date is reached, the access token will be automatically removed from this list.
Actions	The functionality for revoking the access token.

Requesting add-ons

The [Atlassian Marketplace](#) website offers hundreds of add-ons that administrators can install to enhance and extend your JIRA applications. If the add-on request feature is enabled for your instance, you can submit requests for Marketplace add-ons directly to your administrator.

The 'Atlassian Marketplace for JIRA' page presents an integrated view of the Marketplace website from within the JIRA user interface. The page offers the same features as the Marketplace website, such as add-on search and category filtering, but tailors the browsing experience to JIRA application users.

This in-product view of the Marketplace gives day-to-day users of the Atlassian applications, not just administrators, an easy way to discover the add-ons that can help them work. When you find an add-on of interest, you can submit a request with just a few clicks.

Submitting an add-on request

1. From anywhere in the application, open your profile menu and choose **Atlassian Marketplace**.
2. In the Atlassian Marketplace page, use the search box to find add-ons or use the category menus to browse or filter by add-ons by type, popularity, price or other criteria. You can see what your fellow users have requested by choosing the **Most Requested** filter.
3. When you find an add-on that interests you, click **Request** to generate a request for your administrator.
4. Optionally, type a personal message to your administrators in the text box. This message is visible to administrators in the details view for the add-on.
5. When ready, click **Submit Request**.
6. Click **Close** to dismiss the 'Success!' message dialog box.

At this point, a notification appears in the interface your administrators use to administer add-ons. Also your request message will appear in the add-on details view, visible from the administrator's 'Find New Add-ons' page. From there, your administrator can purchase the add-on, try it out or dismiss requests.

Updating an add-on request

After submitting the request, you can update your message at any time. Click the **Update Request** button next to the listing in the 'Atlassian Marketplace' page to modify the message to your administrator.

The administrator is not notified of the update. However, your updated message will appear, as you have modified it in the details view for the add-on immediately.

Using keyboard shortcuts

Keyboard shortcuts are a great way for you to speed up editing, navigating, and for performing actions without having to take your fingers off the keyboard.

Some keyboard shortcuts require additional permissions or applications, and depend on how your JIRA administrator(s) have configured permissions for your user account and which applications are installed.

On this page:

- [View keyboard shortcuts](#)
- [Enabling and disabling keyboard shortcuts](#)

View keyboard shortcuts

- Choose  at top right of the screen, then choose **Keyboard Shortcuts**.
- When viewing a page, press **Shift + /**.

The Keyboard Shortcuts dialog is displayed and shows commands for the operating system and browser that you are using. The dialog is divided into sections for the following information:

- **Global shortcuts** - shortcuts that can be used when you are in any part of JIRA
- **Navigating issues** - shortcuts for navigating through issues
- **Issue actions** - shortcuts for working with issues
- **App specific** - any application-specific shortcuts. These shortcuts only work in the listed application.

More about the Keyboard Shortcuts dialog...

If you have other JIRA applications installed, you may have additional keyboard shortcuts available. For example, if you have JIRA Software installed, you will see a series of additional keyboard shortcuts in the lower-right of this dialog box (and some additional **Global** keyboard shortcuts specific to JIRA Software in the upper-left section). However, the keyboard shortcuts in the **Agile Shortcuts** section only function in JIRA Software, and not in a JIRA context.

Enabling and disabling keyboard shortcuts

Keyboard shortcuts are enabled by default. However, you can disable them on a per-user basis in the Keyboard Shortcuts dialog box.

1. Ensure you are logged in and open the Keyboard Shortcuts dialog box (see [above](#)).
2. At the bottom of the Keyboard Shortcuts dialog box, click **Disable Keyboard Shortcuts** or **Enable**

Keyboard Shortcuts.

You can also disable or re-enable keyboard shortcuts by editing the Preferences section of your user profile. See [Managing your user profile](#) for more information.

Modifier keys

Some keyboard shortcuts require modifier keys to be pressed simultaneously, along with a single 'action' key. Modifier keys may differ, depending on your combination of operating system and web browser. The following table identifies the modifier keys for some supported web browsers and operating systems:

Web Browser	Mac OS X	Windows	Linux/Solaris	Notes
Firefox	Ctrl	Alt + Shift	Alt + Shift	In Firefox, it is possible to customize 'Modifier key shortcuts'. Please read Mozilla's documentation for more information.
Internet Explorer		Alt		Typing a 'Modifier key shortcut' that leads to a link requires you to press the 'Enter' to complete the action.
Safari	Ctrl + Alt/Option	Ctrl		
Chrome	Ctrl + Alt/Option	Alt + Shift	Alt + Shift	

Administering JIRA Software

All administrative functions of JIRA Software require you to be a user with the **JIRA Administrators** or **JIRA System Administrators** global permission.

As JIRA Software is based on the JIRA platform, many of the functions are documented in the JIRA Admin documentation.

Installing JIRA Software and upgrading JIRA Software

If you want to install a JIRA application, or add a JIRA application to your existing JIRA installation, see [Installing JIRA applications \(JIRA Admin documentation\)](#). If you want to upgrade your current version of JIRA, see [Upgrading JIRA applications \(JIRA Admin documentation\)](#).

Backing up JIRA

We recommend that you back up your JIRA instance regularly. You should also back up before undertaking any activities that could result in unexpected data loss/changes, such as upgrading JIRA to a new version or splitting your instance across multiple servers.

Learn more: [Backing up data \(JIRA Admin documentation\)](#)

Supported platforms

We thoroughly test and provide support for our supported platforms (browsers, databases, Java platforms, etc). We strongly recommend that you use the supported platforms for your version of JIRA.

Learn more: [JIRA supported platforms \(JIRA Admin documentation\)](#)

JIRA system administration

JIRA server administration covers server administration (e.g. search indexing, audit logs, etc), global settings (e.g. time tracking, managing shared filters and dashboards, etc), and server optimization (e.g. performance testing, security, etc).

Learn more: [JIRA system administration \(JIRA Admin documentation\)](#)

Permissions overview

JIRA Software has different types of permissions that cover a range of functionalities. This gives you the flexibility to restrict parts of the application to certain users or groups of users. For example, you could allow one user to assign issues in a project, but prevent another user from doing the same.

[Learn more: Permissions overview](#)**Configuring projects**

This page provides information on how you configure projects in JIRA Software, including defining projects, managing versions and components, configuring issues and project permissions, managing project notifications, project schemes, and screens, using the issue collector, working with workflows, etc.

[Learn more: Configuring projects \(JIRA Admin documentation\)](#)**Layout and design**

You can configure the layout and design of JIRA Software to suit your organization's needs and preferences. For example, you can change JIRA's color scheme and logo, set a default dashboard, show an announcement banner, and more.

[Learn more: Layout and design \(JIRA Admin documentation\)](#)

Permissions overview

This page describes the different types of permissions and access rights that can be set up in JIRA applications.

What are permissions?

Permissions are settings within JIRA applications that control what users within those applications can see and do. All JIRA applications allow a variety of permissions: from whether users can create new projects to whether a user can see a specific type of comment on an issue. These permissions can differ between applications.

Permissions are different from application access, which is controlled by groups that have **Use** access for an application. For more information about setting application access, see [Managing user access to JIRA applications](#).

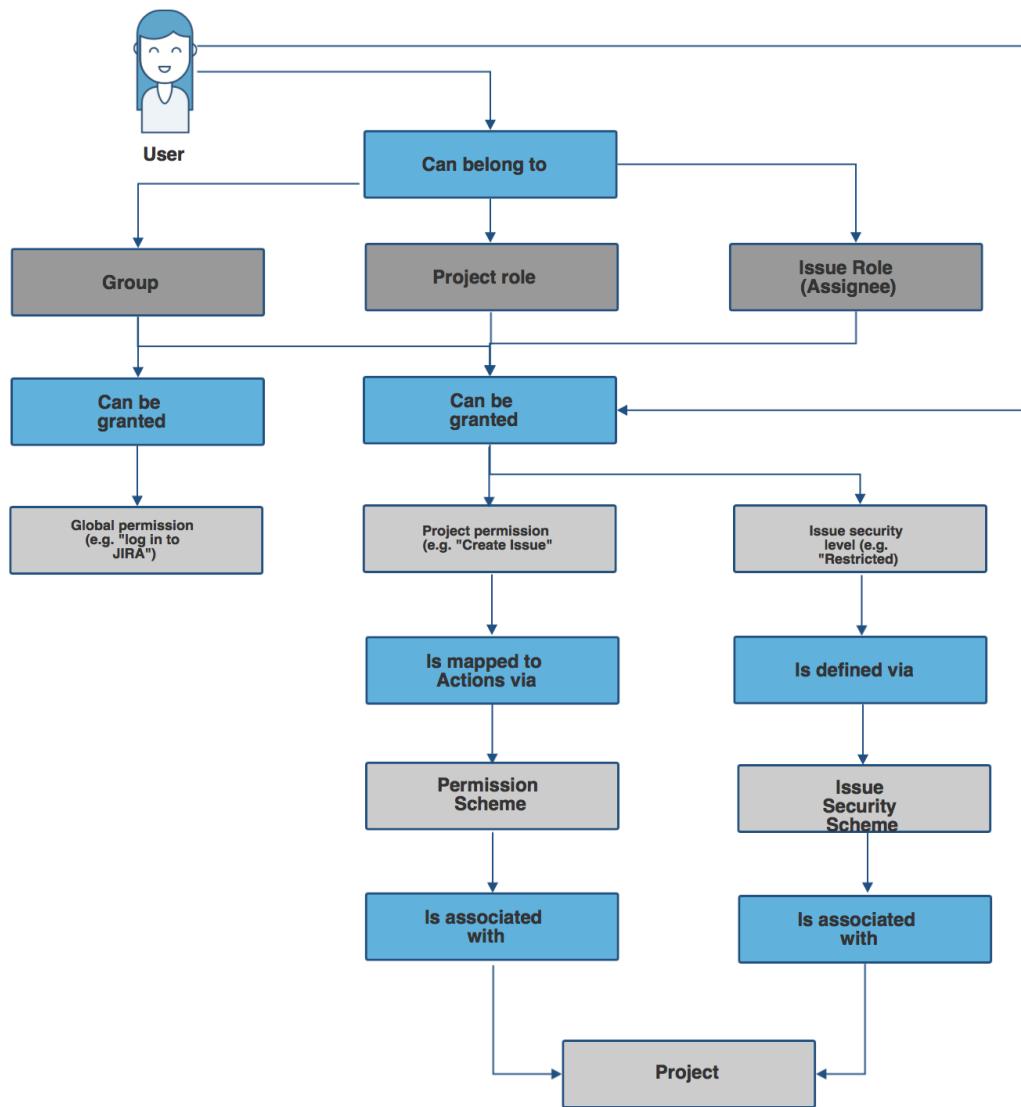
Types of permissions

There are three types of permissions in JIRA applications, and they range from the high-level to granular:

- **Global permissions** - These apply to applications as a whole, not individual projects (for example, whether users can see the other users in the application).
- **Project permissions** - Organized into permission schemes, these apply to projects (e.g. who can see the project's issues, create, edit and assign them). While project admins can assign users to a project, they can't customize the permission schemes for a project. There are lots of project-level permissions you can set to control what users can do within a project.
- **Issue security permissions** - Organized into security schemes, these allow the visibility of individual issues to be adjusted (within the bounds of the project's permissions). For example, issue security permissions can let you set up types of issues that can only be seen by project admins or users in specific groups.

How do permissions get assigned?

Permissions can be assigned to groups or to project roles/and or issue roles. This diagram illustrates how permissions are assigned to users:



Who can set permissions?

Permission	Can be set by	For more info, see...
Global permission	A user with the JIRA System administrator permission A user in a group with Admin access	Managing global permissions

Project permission	A user with the JIRA System administrator permission A user in a group with Admin access	Managing project permissions
Issue security permission	A user with the JIRA System administrator permission A user in a group with Admin access A project admin	Configuring issue-level security

Board permissions can be divided into two parts — board administration permissions and board usage permissions.

Board administration permissions cover functionality for changing the configuration of a board. For example, changing columns, customizing cards, etc. Board administration can be assigned to groups or users.

Learn more: [Configuring a board](#)

Board usage permissions cover functionality for the usage of a board. For example, creating sprints, ranking issues, etc. Board usage permissions are derived from project permissions. This is described in more detail below.

JIRA Software functions by permission

In the JIRA Software documentation, most configuration options are described as being restricted to either JIRA administrators, project administrators, or board administrators.

- A JIRA administrator is a user with the **JIRA administrators** global permission.
- A project administrator is a user with the **Administer projects** project permission for a particular project.

By default, the 'Administer projects' permission is assigned to the 'administrators' group (via the Administrators role) for projects.

Additionally, to perform sprint-related actions, users need the 'Manage sprints' permission for all projects in the origin board — the origin board being the board in which the sprint was originally created.

- A board administrator is a user that has been added to the **Administrators** for a particular board.
By default, the administrator of a board includes the person who created it.

Board administration

Function / Functional area	JIRA admin	Project admin	Board admin	Notes

Create board				<p>If you create a board via Boards (in header) > Manage Board, you will not be able to share it, unless you have the 'Create Shared Objects' global permission.</p> <p>If you create the board via the methods below, you do not need the 'Create Shared Objects' global permission to share the board:</p> <ul style="list-style-type: none"> • Creating a project (where a board is created for the project by default) • Setting up JIRA Software for the first time (where you're prompted to create a project, which also creates a board for the project) • Copying a board (the copied board will be shared with the same users as the original board)
Simplify workflow				The board must meet other criteria as well (see Simplified Workflow)
Add status				<p>Project must be currently using an Agile Simplified Workflow.</p> <p>You need to be a JIRA admin or board admin (to view the board configuration), <i>In addition</i>, you need to be a project admin for the one project that is on the board.</p>
Remove status				<p>Project must be currently using an Agile Simplified Workflow.</p> <p>You need to be a JIRA admin or board admin (to view the board configuration), <i>In addition</i>, you need to be a project admin for the one project that is on the board.</p>
All other board configuration functions				

Board usage

Board usage permissions are derived from project permissions.

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](#).

Function	Permission Level	Notes
Backlog — Sprints		
Move sprint footer	Manage Sprints permission (for all projects in the board) Schedule Issues permission and Edit Issues permission	
Move issue (reorder/rank)	Schedule Issues permission and Edit Issues permission	Not required if you only move issues across the sprint footer without changing the order of the issues

Start sprint	Manage Sprints permission (for all projects in the board)	Similar permission to creating a Version. Board ownership does not play a role here.
Create sprint	Manage Sprints permission (for all projects in the board)	This permission applies even if the sprint (that is to be started) does not include issues from all projects queried by the board.
Edit sprint information	Manage Sprints permission (for all projects in the board)	Can only be done in the backlog
Reorder sprint	Manage Sprints permission (for all projects in the board)	
Delete sprint	Manage Sprints permission (for all projects in the board)	
Add issue to sprint	Schedule Issues permission and Edit Issues permission	

Active sprints — Sprints

Add issue to sprint	Schedule Issues permission and Edit Issues permission	
Complete sprint	Manage Sprints permission (for all projects in the board)	Can only be done in Active sprints
Remove issue from sprint	Schedule Issues permission and Edit Issues permission	

Backlog — Epics

Create epic	Create Issues permission	
Rename epic	Edit Issues permission	
Rank epic	Schedule Issues permission	
Add issue to epic	Edit Issues permission	
Remove issue from epic	Edit Issues permission	

Backlog — Versions

Create version	Project Admin permission, or JIRA Admin permission	
Edit version	Project Admin permission, or JIRA Admin permission	

Add issue to version	Edit Issues permission	
Remove issue from version	Edit Issues permission	

Managing project role memberships

You can use project roles to easily associate users and groups with a particular project. For example, you may want to send notifications to a specific set of users associated with your project, and by adding them all to a project role, you can then use that project role to control who receives the notifications. You can also use project roles to restrict how much access certain users or groups have. Unlike groups, which have the same membership throughout your application, project roles have specific members for each project.

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

Viewing and editing project role members

1. Log in as a project administrator and open your project.
2. Select  **> Users and roles.**
3. You'll see all users and groups associated with each project role.
4. To add users or groups to a project role, select **Add users to a role** in the top right corner. Enter the users or groups and select the project role you wish to add them to.
5. To remove a user or group from a project role, hover over the user or group row, and select 

Since 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.

Getting help

How can we help you?

We have a number of [help resources](#) available. You can get your problem resolved faster by using the appropriate resource(s).

I don't know how to do something

Something isn't working

I don't like the way something works

Something else?

I don't know how to do something

1. Search [Atlassian Answers](#).
2. Raise a [support request](#)*

Something isn't working

1. Check the JIRA Software knowledge base at <https://confluence.atlassian.com/display/JIRAKB>.
2. Search [Atlassian Answers](#).
3. Raise a [support request*](#).

If you've identified a bug but don't need further assistance, raise a [bug report](#). (<https://jira.atlassian.com>)

I don't like the way something works

Raise a [suggestion](#). (<https://jira.atlassian.com>)

Something else?

If you need help with something else, raise a [support request*](#).

* *Tip: If you are the **JIRA system administrator**, you have a number of additional support tools available. See [Raising support requests as an administrator](#) for details.*

About our help resources

Atlassian Support

Our support team handles support requests that are raised in our [support system](#). You need to log in using your [Atlassian account](#) before you can raise support requests.

For information on our general support policies, including support availability, SLAs, bugfixes, and more, see [Atlassian Support Offerings](#). Note, you'll find anything security-related at [Security @ Atlassian](#).

Atlassian Answers

[Atlassian Answers](#) is our official application forum. Atlassian staff and Atlassian users contribute questions and answers to this site.

You may be able to find an answer immediately on Atlassian Answers, instead of having to raise a support request. This is also your best avenue for help if:

- you are using an unsupported JIRA instance or an unsupported JIRA platform,
- you are trying to perform an unsupported operation, or
- you are developing an add-on for JIRA Software.

You can also have a look at the [most popular JIRA answers](#).

JIRA Software knowledge base

If there are known issues with a version after it has been released, the problems will be documented as articles in our [knowledge base](#).

Atlassian issue tracker

Our official [issue tracker](#) records our backlog of bugs, suggestions, and other changes. This is open for the public to see. If you log in with your Atlassian account, you will be able to create issues, comment on issues, vote on issues, watch issues, and more.

Tip: Before you create an issue, search the existing issues to see if a similar issue has already been created.